Malware Analysis

Ajit Gaddam

TABLE OF CONTENTS

ajit@ospreysecurity.com

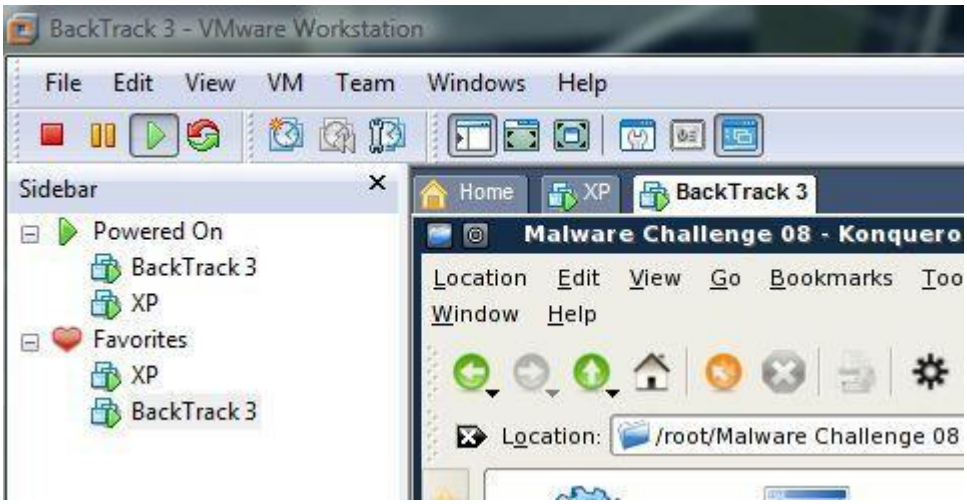
1 LAB ENVIRONMENT

Always perform the malware analysis in a controlled environment and a virtual machine gives us the perfect opportunity to do something like this. In order to analyze the malware specimen **malware.exe**, a virtual malware analysis environment was setup using VMware Workstation 6.0.

The host operating system is Windows Vista Ultimate SP1 with the latest patches installed. Two Virtual Machine (VM) images were then prepared and used to perform the analysis of malware.exe. The first one is Windows XP Pro SP3 with all the latest patches installed. The second virtual machine is an installation of BackTrack 3[1], a Linux distribution that resulted from the merger of WHAX[2] and the Auditor Security Collection. In order to ensure that the virtual network is truly isolated from the physical interfaces of the host machine, ZoneAlarm Pro was configured on the hosting machine. In addition, VMware Tools was not installed on both the virtual machines to prevent any form of interaction between the guest operating systems and the host machine.

The table below is a summary of the virtual network environment that was used to perform the malware analysis.

Station	Operating System	Disk Space	Memory	IP Address	Netmask
HOST	Windows Vista SP1	1TB	4GB	192.168.1.100	/24
XP	Windows XP Pro SP3	25GB	500MB	192.168.1.101	/24
BackTrack 3	Linux	5GB	256MB	192.168.1.102	/24



[1] http://www.remote-exploit.org/backtrack.html
[2] http://distrowatch.com/table.php?distribution=Whoppix

2 STATIC ANALYSIS

2.1 Downloading the malware

First, a zipped copy of the malware file was downloaded from the Information Security 2008 Malware Challenge website[3]. The md5 hash of malware.zip is **31d2ec3b312d0fd27940aae5c89e3787** that matches the one provided by the Information Security 2008 Malware challenge website. The zipped file was password protected with the following password: **infected**. Extracting the file gives us **malware.exe**. The next step was to generate the md5 hash of malware.exe. This gave an md5 hash value of **59A95F668E1BD00F30FE8C99AF675691**. I also Googled the md5 hashes of both malware.exe and malware.zip to see if there are any results generated. While there was none for malware.exe, a search for the md5 hash of malware.zip pointed to this online analysis[4]

2.2 Virus Scan

Traditionally, I would have scanned the malware using an Anti-Virus (AV) solution on the virtual machine. However, I wanted to give the malware free reign to do what it wants to do and did not wish to restrict the malware in any way by installing an AV product. Using an online malware analyzer provided by virscan.org, generated the following analysis:

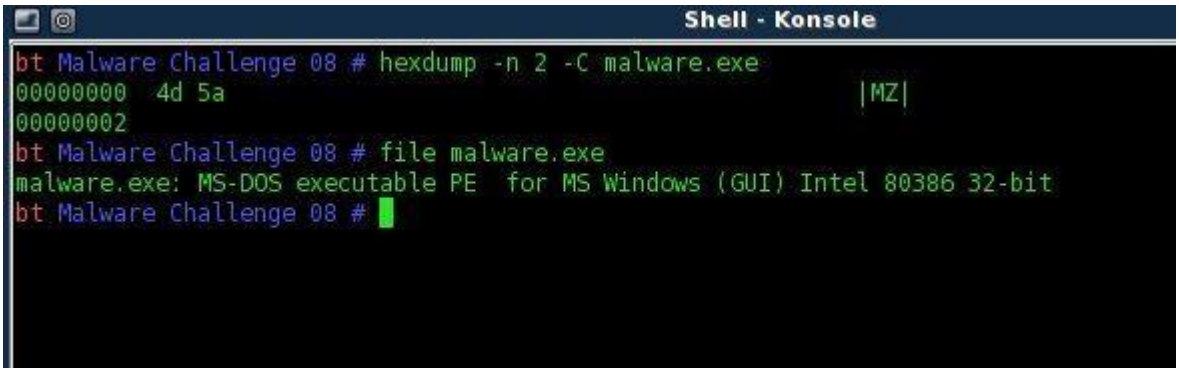
```
Backdoor.Win32.Rbot.bzf [Kaspersky Lab]

W32.IRCBot [Symantec]

W32/Sdbot.worm [McAfee]

WORM_RBOT.GEN-1 [Trend Micro]

W32/Rbot-Fam, Mal/Behav-024, Mal/IRCBot-B [Sophos]

Backdoor:Win32/Rbot.gen [Microsoft]
```

2.3 File Information

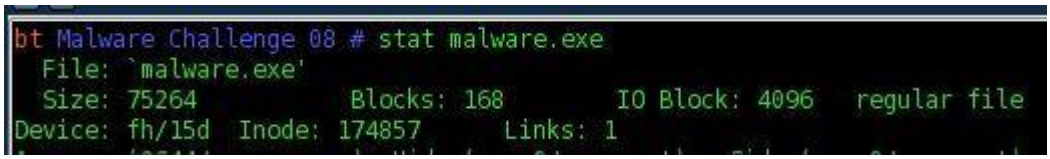
I wanted to verify the malware.exe is indeed an executable. Using **hexdump command**, I generate the first two bytes of the malware file which equals 0x4d5a. Converting this to ASCII gives us MZ which

[3] http://www.malwarechallenge.info/challenge.html
[4] http://virscan.org/report/4d8c9a5fe9b112db137feef1cc2d1ba7.html

means that malware.exe is most likely an executable file. To further verify this assessment, the **file command** was used which tells us that malware.exe is an MS-DOS executable PE file.



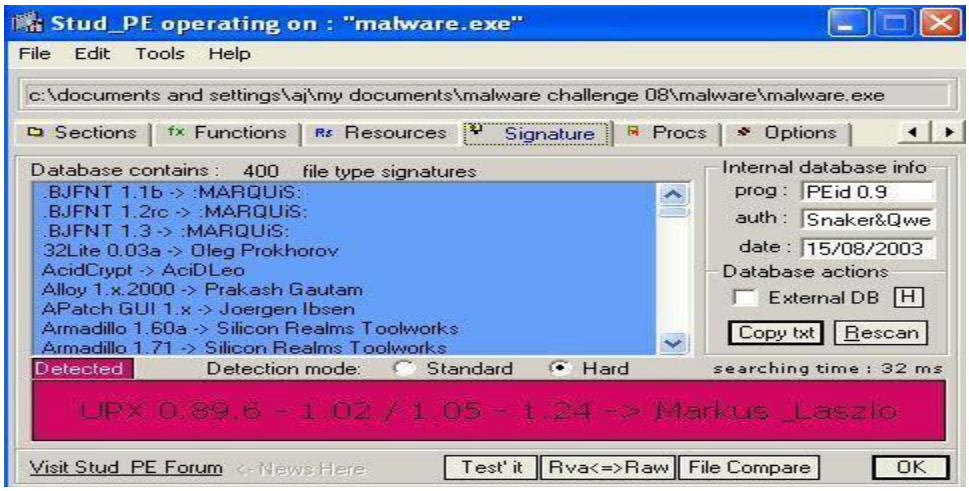
I also note the file size of malware.exe of 75264 byes, and other malware.exe file property information.



The next thing I did was perform a visual inspection of the file and its properties. Taking a look at file properties like file name, size and other file information can give us more information on the malware file. I also used WinRAR to provide me additional file information on the malware.

2.4 PE and UPX Analysis

After confirming that malware.exe is an executable PE file, I switch to the Windows XP VM. Using a tool named Stud_PE, I validated that the file was packed with UPX. Stud_PE is an excellent tool using a database of known packing signatures, it can detect what tools/versions were used the malware file.



After Stud_PE determined that the binary used the UPX packing method, an attempt the unpack the malware gave an error that it could not unpack the binary and generated the following error.

```
CantUnpackException: file is modified/hacked/protected; take care!!!
```

```
bt Malware Challenge 08 # upx -d malware.exe
                       Ultimate Packer for eXecutables
    Copyright (C) 1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006
UPX 2.03        Markus Oberhumer, Laszlo Molnar & John Reiser    Nov 7th 2006

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
upx: malware.exe: CantUnpackException: file is modified/hacked/protected; take care!!!

Unpacked 0 files.
bt Malware Challenge 08 #
```

This could mean two things, either Stud_PE made a mistake determining the packing method or more likely in this case, the binary has been mangled with to stop it from being unpacked this way.

The authors of this malware changed the UPX version number to prevent unpacking. The UPX version was changed using an HEX editor from 1.24 to 3.03, and the UPX strings denoting the binary sections, were modified 3 times, changing each instance of UPX with ABC. Replacing ABC with UPX using a Hex Editor and changing the version number from 3.03 to 9.99 makes it possible to uncompress the binary using the standard UPX tool.

The malware is now expanded from 75264 bytes to 169472 bytes. I also generate the md5 hash of the unpacked malware which is **c0e5bd6d5e3579e659841b98e2cca59d**

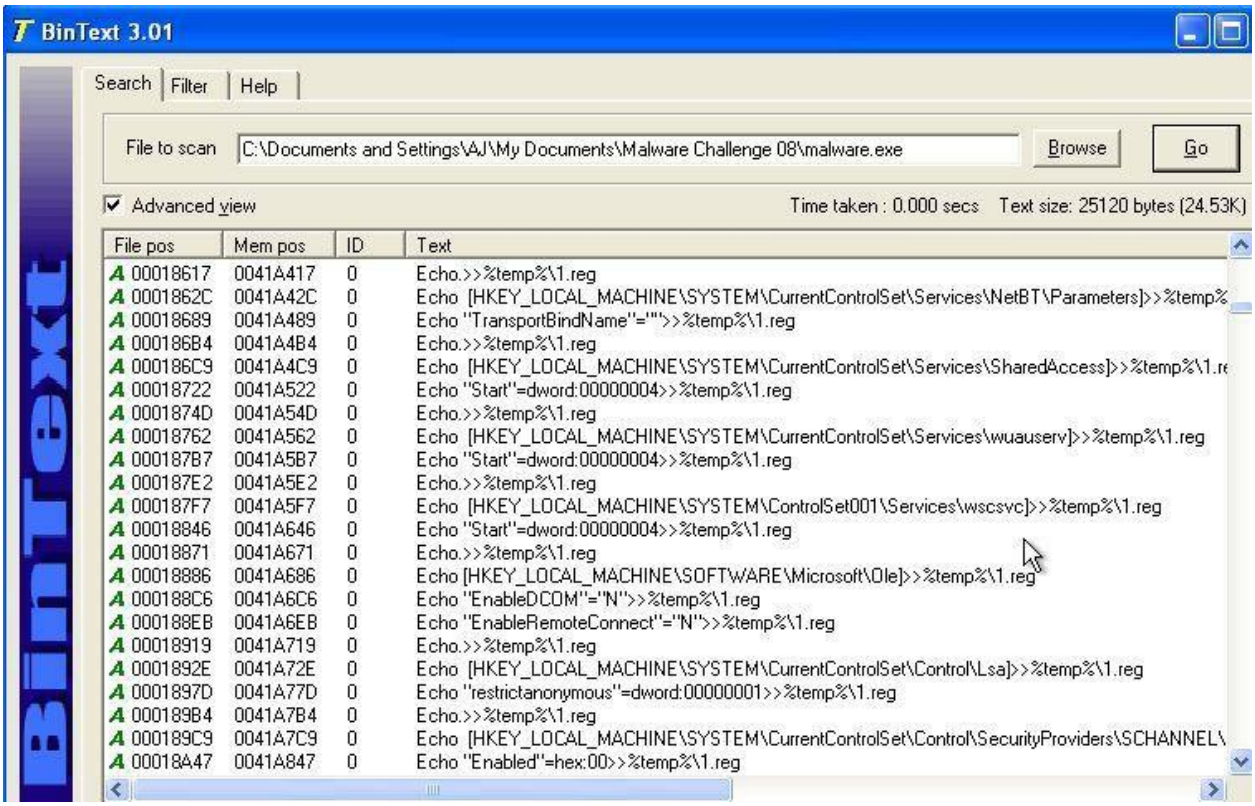
```
Shell - Konsole
bt Malware Challenge 08 # strings --all malware.exe | head -n 6
!This program cannot be run in DOS mode.
Rich
ABC0
ABC1
ABC2
3.03
bt Malware Challenge 08 # cd 1
bt 1 # strings --all malware.exe | head -n 6
!This program cannot be run in DOS mode.
Rich
UPX0
UPX1
UPX2
9.99
bt 1 # upx -d malware.exe
                       Ultimate Packer for eXecutables
    Copyright (C) 1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006
UPX 2.03        Markus Oberhumer, Laszlo Molnar & John Reiser    Nov 7th 2006

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    169472 <-     75264   44.41%    win32/pe     malware.exe

Unpacked 1 file.
bt 1 #
```

2.5 Strings Analysis

After unpacking the file with UPX tool, I ran the **strings command** and did a simple output redirection to a text file for better viewing of malware.exe and see any embedded ASCII and UNICODE strings that may be contained in this file. This analysis can give us clues to the composition and the various characteristics of this file

```
strings –o malware.exe > strings-malware.exe.txt
```

The result is a text file of 3885 lines. This can be a bit hard to analyze for lack of filtering tools. I use the excellent BinText tool based on Windows for a quick look inside malware.exe. On the Filter tab of BinText, I changed the minimum text length from 5 to 15, in order to eliminate the noise produced by the file. This left a few useful strings as you can see below



The registry keys that are being targeted by malware.exe and some of the keywords such as “join”, listening on etc, all hints that we are dealing with an IRC bot.

3 BEHAVIORAL ANALYSIS

This section will focus on the dynamic analysis of the malware. In order to detect and monitor how this executable behaves, a completely new tool set will be required.

3.1 Tools used

For this analysis, I will use a clean image of Windows XP SP3 with the unpacked malware.exe file being the only carryover from the analysis above.

To analyze the malware’s activity, I will be using Process Monitor[5] from Microsoft. Process Monitor combines the sysinternals tools, Filemon (to monitor all access to the file system), Regmon (to monitor

[5] http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx

all access to the Registry) and TDImon (to record all usage of the system's TCP/IP sockets). Process Monitor is also used to terminate any process after a short while, unless it terminates itself.

I will be using Wireshark[6] to monitor and sniff network traffic.

Finally, I will also be using RegShot[7] to perform a basic before-and-after comparison of the file system and registry.

After, all the tools above were installed, a snapshot was taken of the Windows XP image.

We are now ready to launch the malware!!!

3.2 Memory Modifications

There were new processes created in the system. Information that is more relevant is in the sections below. However, the new process name that is spawned is **Winsec32.exe**.

This process is created under %Windir%\winsec32.exe and the main module size is 495,616 bytes.

3.3 File System behavior

Because most malware change files stored on the file system, a great way to analyze their behavior is to execute them on the test system and monitor the file changes. I perform this analysis using the Filemon tool from Sysinternals as discussed in the tools section above.

After analyzing the data from Filemon and the malware code as detailed in section 4 Code Analysis, following is the list of file changes made by the malware to the system. The chronological order was determined after the code analysis but is included in this section. The data generated by filemon is further filtered to distill the data we care about.

Chronological order of files launched/accessed by the malware

```
Open File: \\.\PIPE\lsarpc (OPEN_EXISTING)

Create/Open File: \Device\Tcp (OPEN_ALWAYS)

Create/Open File: \Device\Ip (OPEN_ALWAYS)

Create/Open File: \Device\Ip (OPEN_ALWAYS)

Open File: \\.\Ip (OPEN_EXISTING)

Get File Attributes: C:\WINDOWS\Winsec32.exe Flags: (SECURITY_ANONYMOUS)

Copy File: c:\malware.exe to C:\WINDOWS\Winsec32.exe

Open File: C:\WINDOWS\explorer.exe (OPEN_EXISTING) Open

File: C:\WINDOWS\Winsec32.exe (OPEN_EXISTING)
```

[6] http://www.wireshark.org/

[7] http://sourceforge.net/projects/regshot

```
Set File Time: C:\WINDOWS\Winsec32.exe

Set File Attributes: C:\WINDOWS\Winsec32.exe Flags:
(FILE_ATTRIBUTE_HIDDEN,FILE_ATTRIBUTE_READONLY,FILE_ATTRIBUTE_SYSTEM,SECURITY_ANONYMOUS)

Open File: C:\WINDOWS\AppPatch\sysmain.sdb (OPEN_EXISTING)

Open File: C:\WINDOWS\AppPatch\systest.sdb (OPEN_EXISTING)

Open File: \Device\NamedPipe\ShimViewer (OPEN_EXISTING)

Open File: C:\WINDOWS\ ()

Find File: C:\WINDOWSWinsec32.exe
```

3.4 Registry Behavior

Using Regshot, comparing and analyzing the pre-infection and post-infection snapshots of the registry informs us of the following registry changes to the infected host. Also from Regshot, we can see that the malware is going to run as a service, so it could be more difficult to deal with. The changes also include new registry keys added which ensure that the malware auto-starts every time the infection host starts up. Another tool that was used to analyze registry behavior was Regmon from Sysinternals.

Registry Keys added – 2

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices

HKEY_CURRENT_USER\Software\Microsoft\OLE
```

Registry Values added – 4

The purpose of the two registry values shown below under "Auto-start" below is to ensure that the newly created svchost.exe process is triggered every time Windows boots.

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Microsoft Svchost local
services: "Winsec32.exe"
```

Auto-start:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices\Microsoft Svchost
local services: "Winsec32.exe"

HKEY_CURRENT_USER\Software\Microsoft\OLE\Microsoft Svchost local services:
"Winsec32.exe"
```

This registry value change needs further analysis

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssi
st\{75048700-EF1F-11D0-9888-006097DEACF9}\Count\HRZR_EHACNGU:P:\Qbphzragf naq
Frggvatf\NW\Qrfxgbc\Znyjner Punyyratr 08\znyjner.rkr: 01 00 00 00 06 00 00 00
60 EA 9A 54 34 35 C9 01
```

The malware also creates a mutex to prevent only 1 bot per infected host. The key is "RasPbFile" and uses this to extract the system directory, time and name of the infected machine to create the bot entry.

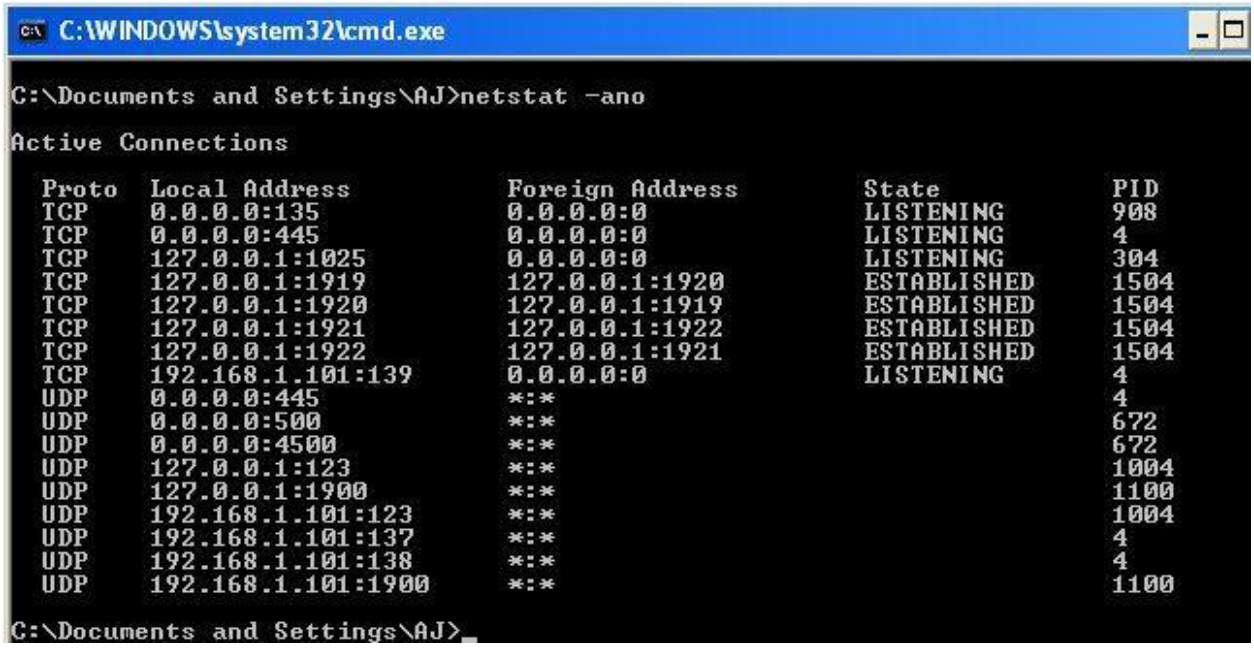
3.5 Monitoring Processes and threads

Monitoring processes and thread activity can reveal important information about a malware's internal structure. I use the threads that are spawned here and create breakpoints when analyzing them in the code analysis section using a disassembler. While Task Manager can be used to monitor processes, Process Explorer from Sysinternals was used for monitoring and killing the malware when needed or any other unwanted tasks quickly.

3.6 Port Monitoring

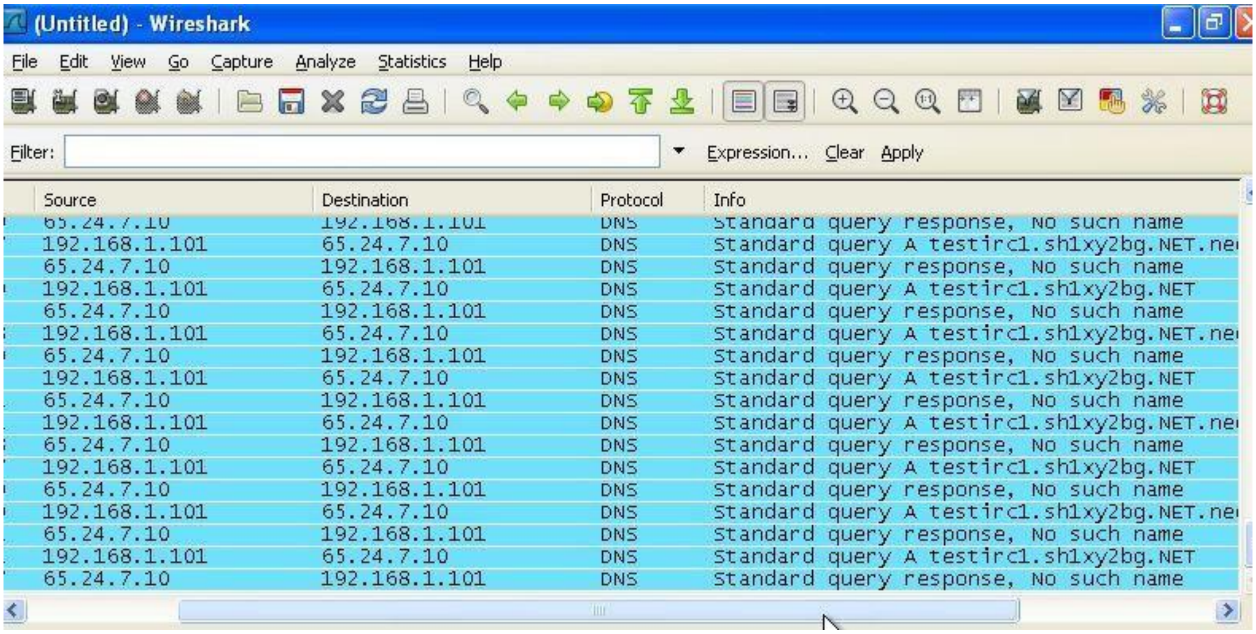
During the initial DNS queries by the malware, we don't see any port activity on the well known IRC ports such as 6667.

To monitor port activity, fport from Foundstone is an excellent too. Here, using netstat I monitored port activity after restarting the infected host. Using Process Explorer, I verified that the malware indeed auto-starts upon reboot. Windows machines begin allocating ephemeral client ports starting at 1024 when the system starts up. The low client port used here suggests that the mIRC client is one of the first network-enabled applications that run.



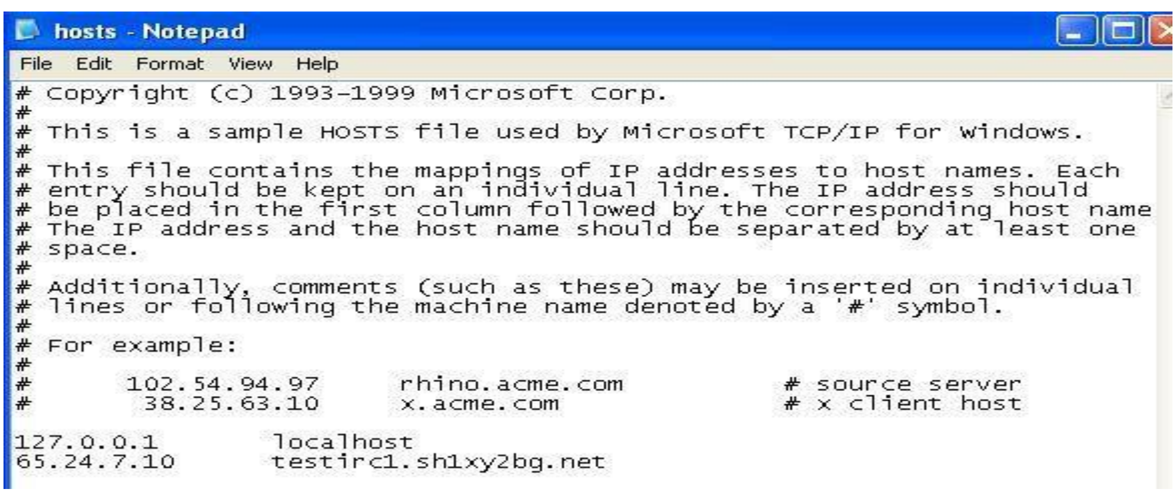
3.7 Network Behavior

Wireshark was used for capturing network traffic as indicated in the tools section in 3.1. As shown in the figure below, the malware seems to be making DNS queries to a specific site **testirc1.sh1xy2bg.net**. It keeps getting a response back that such a site doesn't exist.

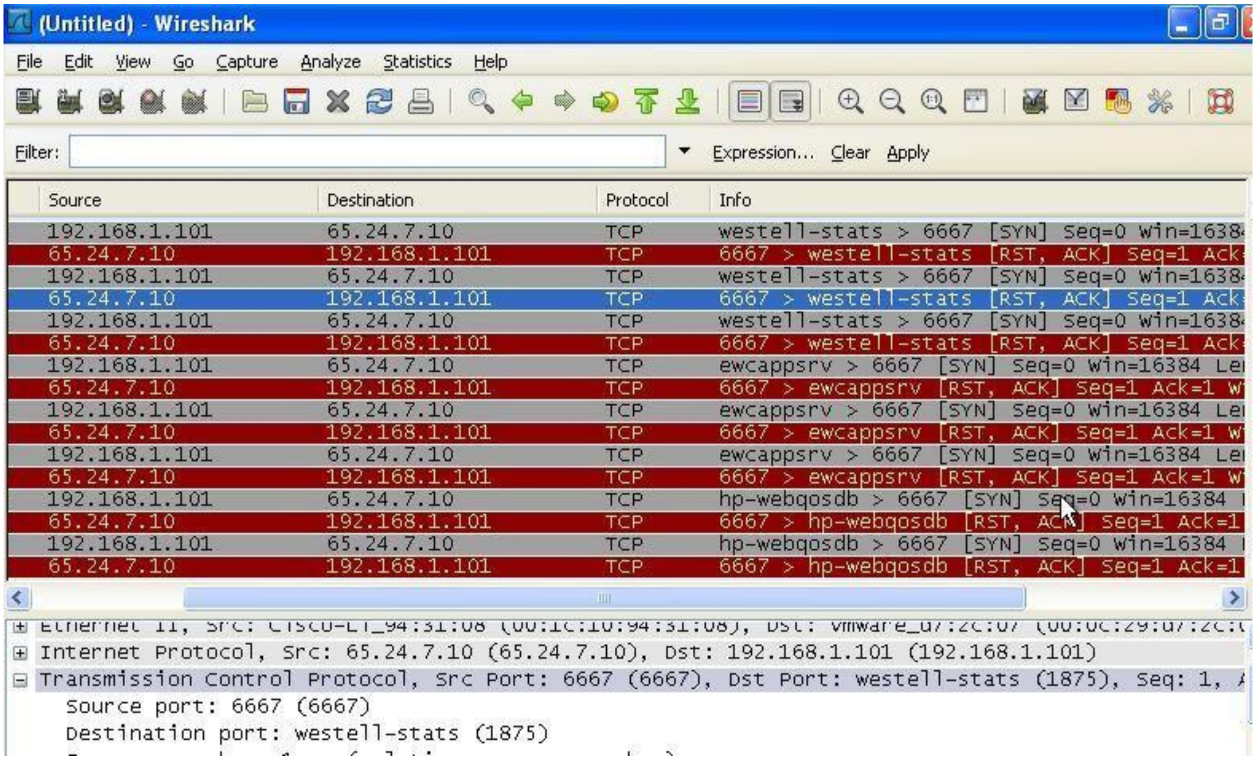


So, our malware wants to go to the internet. We need to trick the malware that it is indeed connecting to the testirc1 server. Rather than bringing up a DNS server on the network, I added an entry to the infected machine's hosts file that resolved testirc1.sh1xy2bg.net to IP address as shown below. The purpose of this configuration step was to redirect the trojan to a server under our control so that we can observe the nature of the connection that the program would make to the system whose name it was trying to resolve.

Manipulating the DNS records in this case was trivial. Had the malware's author hard-coded an IP address into the program, I would have had to configure local routing tables and network parameters to redirect traffic to my system.

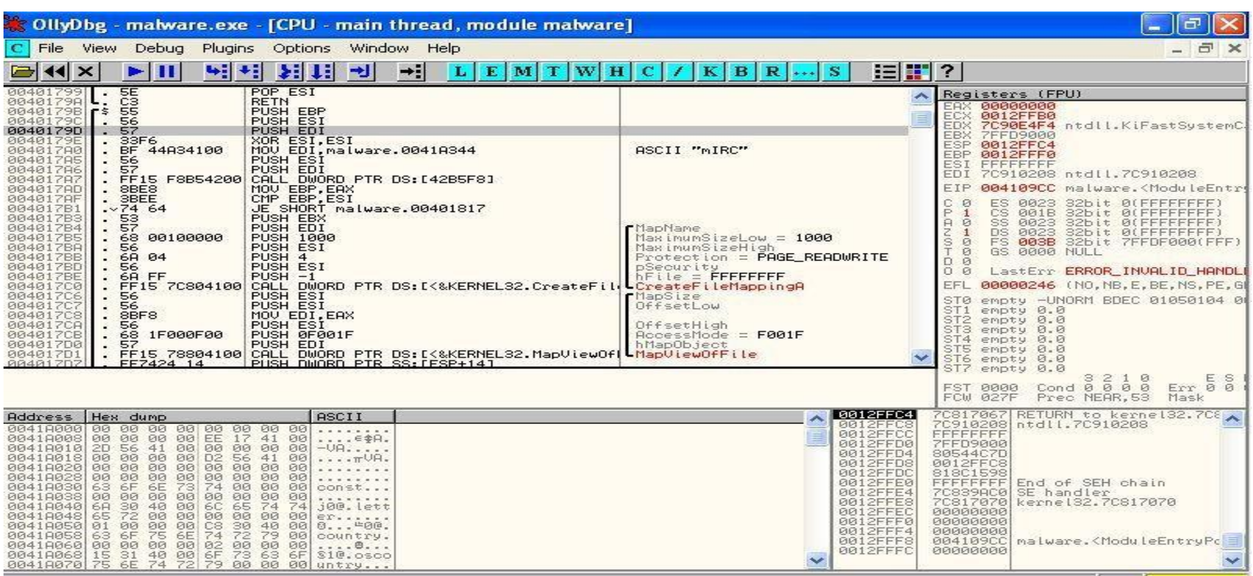


After modifying the host file and running the malware again, tells us that the malware is looking for port 6667/TCP. Port 6667 is normally associated with IRC and numerous Trojans.



4 CODE ANALYSIS

OllyDbg was used to unpack the malware and dump the strings that were found. Notice the advertisement of the kind of malware we are dealing with in the figure below



4.1 DLLs Accessed

The following are the list of DLLs handled by the malware throughout its process of execution

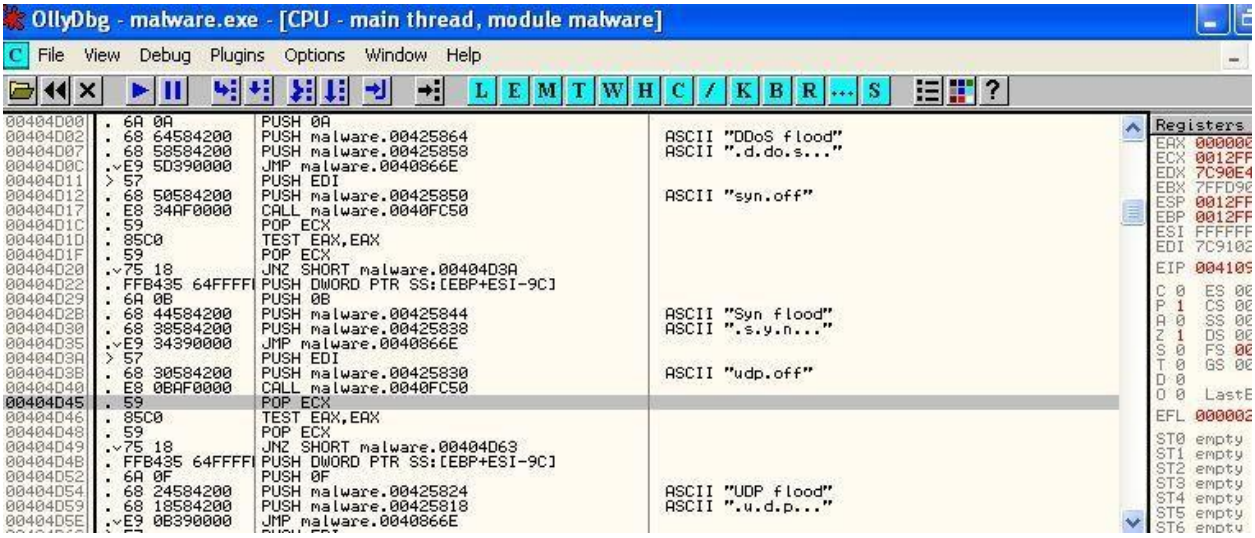
```
C:\WINDOWS\system32\ntdll.dll C:\WINDOWS\system32\kernel32.dll C:\WINDOWS\system32\WS2_32.dll
```

```
C:\WINDOWS\system32\ADVAPI32.dll C:\WINDOWS\system32\RPCRT4.dll C:\WINDOWS\system32\Secur32.dll

C:\WINDOWS\system32\msvcrt.dll C:\WINDOWS\system32\WS2HELP.dll C:\WINDOWS\system32\user32.dll

C:\WINDOWS\system32\GDI32.dll C:\WINDOWS\system32\oleaut32.dll C:\WINDOWS\system32\ole32.dll

C:\WINDOWS\system32\IMM32.DLL C:\WINDOWS\system32\pstorec.dll C:\WINDOWS\system32\ATL.DLL

C:\WINDOWS\system32\wininet.dll C:\WINDOWS\system32\icmp.dll C:\WINDOWS\system32\netapi32.dll

C:\WINDOWS\system32\dnsapi.dll C:\WINDOWS\system32\iphlpapi.dll C:\WINDOWS\system32\mpr.dll

C:\WINDOWS\system32\SHELL32.dll C:\WINDOWS\system32\odbcint.dll C:\WINDOWS\system32\odbc32.dll
```

4.2 Malware Analysis

Performing the code analysis of the malware bought out some interesting points

- The malware starting section deals with the dlls and files it needs to access
- When creating winsec32.exe, it enforces the file attributes Readonly and Hidden. This is the reason if you try going to C:\Windows\Winsec32.exe, you can't find it
- The malware once installed on the infected machine can help participate in DoS and/or DDos attacks against other networks and hosts



- The malware talks back to the command and control server under the name "**Crxbot Alias REalmbot" by Lindem**.
- The bot reports information back to the Command and Control (C&C) server. This includes reporting to the C&C server if smtp is not enabled, if the bot is attacking or participating in an attack, download a URL or even if it created further clones.
- The malware scans for and can spread across the following weakly restricted network shares Admin$, C$, D$ and IPC$
- The malware has reporting ability on the infections (cloning) etc. This data is used by the C&C to take payment for any malicious activity using "**e-gold**"

- Even creates a custom HTML reporting page for the C&C or malware author including timestamps, files, directories etc. Not standards compliant though with HTTP/1.0 and uses tables
- Microsoft Visual C++ Runtime Library was used

4.3 IRC communication

I intentionally put of setting up an IRC server to trick the malware into thinking that it is communicating with testirc1.sh1xy2bg.net after modified the host file in section 3.3. I could have gone blindfolded but doing the code analysis gave me much more data to use as detailed below.

4.3.1 Domains/URLs accessed by the malware

http://www.nivdav.net/winsec32.exe

- testirc1.sh1xy2bg.net http://www.w32-
- gen.us
-

4.3.2 IRC Channel

- The IRC channel the malware logs into is **#chalenge happy12**
- The Userid is Nick
- The password is gemp123

4.3.3 Commands present in the malware

Besides commands to perform DDos attacks, SYN attacks etc, some other commands present in the malware are

```
!open , !nick, !rnick, !ban, !stop, !run, !clone, !server, !join, !rehash,
!take, !let, !me, !ame, !quit, !say, !msg, -msg, !mode, !cycle, !seen, !exit,
!raw, !info, !alias, !keylog, !pay, !rest, !pwd, !port, !retr
```

4.3.4 IRC Traffic with remote C&C server

In response to the observed behavior in section 3.7, I installed and configured ircd-hybrid software[8] on my Linux virtual machine to provide IRC services to the trojan. After starting the trojan again, I was able to monitor the conversation with the configured IRC server.

The trojan attempted to log in to the IRC channel “#chalenge happy12” using the nickname “Nick”. The malware also attempts to change the Bot ID based on the mutex it created during the initial infection of the host.
The bot participated in periodic “pings” to ensure that the IRC client on the infected host is alive. Also, when two instances of the trojan were launched simultaneously, while the first one connected with Nick, the second instance, was not allowed to use the same nickname, since IRC protocol requires the

[8] http://www.ircd-hybrid.com/

nicknames to be unique on the same IRC network. This forced the trojan to generate a different nickname for itself in a pseudo-random manner.

The generated outbound IRC traffic is provided below. Note: Sample IRC traffic has been summarized for brevity

```
NICK USA[XP]9853154

USER zokebip 0 0 :USA[XP]9853154

USERHOST USA[XP]9853154

MODE USA[XP]9853154 +i-x+s

JOIN #chalenge happy12

NOTICE USA[XP]9853154 :.VERSION http://www.W32-gen.us (-National Virus Site-).

NOTICE #chalenge :USA[XP]9853154 has just versioned me.

PRIVMSG #chalenge :RealmBoT (irc.p.l.g) ....  Status: Ready. Bot Uptime: 0d 0h 0m.

PRIVMSG #chalenge :RealmBoT (irc.p.l.g) ....  Bot ID: 1.

PRIVMSG #chalenge :RealmBoT (portscan.p.l.g) ....  Exploit Statistics: VNC: 0, Total: 0
in 0d 0h 0m.

PRIVMSG #chalenge :RealmBoT (irc.p.l.g) ....  Uptime: 0d 0h 35m.

PRIVMSG #chalenge :[REALMBOT] : Failed to start scan, port is invalid.

NICK USA[XP]3027712

USER ltnxdswau 0 0 :USA[XP]3027712

USERHOST USA[XP]3027712

MODE USA[XP]3027712 +i-x+s

NICK USA[XP]5454035

USER ltnxdswau 0 0 :USA[XP]5454035

USERHOST USA[XP]5454035

MODE USA[XP]5454035 +i-x+s

NICK USA[XP]5409471

USER inwewch 0 0 :USA[XP]5409471

USERHOST USA[XP]5409471

MODE USA[XP]5409471 +i-x+s
```

5 MALWARE CHALLENGE QUESTIONS & ANSWERS

The questions required as part of this challenge have been answered at various sections of this document. This section serves as a placeholder to summarize some of the answers as well as point to more pertinent sections for detailed analysis/answers for the questions asked.

1. Describe your malware lab.

The setting up of a controlled and sanitized environment is absolutely essential for analyzing malware. A special "test lab" is created for this purpose as outlined in section 1 of this document.

- Two virtual machines were created, one Windows based and other Linux based. Both these virtual machines were networked. Proper precautions were taken to isolate them from the host machine and the rest of the network
- Fresh copies of Operating systems were installed on the virtual machines. Initial analysis revealed that I was dealing with a Win32 binary, the Windows machine acts as the "victim host" and where applicable, the *nix machine is used as the "sniffer machine"
- The only tools that were installed on the victim host were the tools required for malware analysis

Apart from this, as further analysis revealed, the malware tries to communicate with a remote server. An IRC daemon was setup and appropriate DNS and host entries were created and configured. In light of new information generated during the later stages of the analysis, the lab was tweaked and modified appropriately.

2. What information can you gather about the malware without executing it?

By examining the malware with a number of tools we can perform analysis without executing it. Section 2 of this document gives detailed analysis to answer this question.

- The first step was to perform an MD5 hash search of the extracted malware. Using Google or any other search engine should give us a preliminary results back if any exist
- The second step involved filtering with a single virus scanner or a set of virus scanners. I performed mine using an online anti-virus scanner which returned results classifying the malware as an IRC bot. Further analysis confirms this result
- Next, we verify the file information. The malware is confirmed as a Win32 PE executable. I also looked for any copyright messages or any company names or any distinct keywords that will help in analysis later
- Another important step was to dump strings of the analyzed object with tools like "strings". In this instance, while the malware was packed with UPX, it was not encrypted. I also used the BinText tool to filter the strings analysis
- Also, code analysis as detailed in section 4 of this document was performed with the debugging turned off as I did not want to accidentally run the code instead of disassembling it. Code analysis also gives valuable information. In depth analysis after gaining more knowledge of the malware was performed later.

3. Is the malware packed? If so, how did you determine what it was?

Most of the 32-bit computer viruses belong to a class of computer worms that are packed with some sort of run-time packer, such as UPX or ASPACK. Malware authors take advantage of packing because they can hide the intent of their code. Furthermore, packed malware is more efficient because when smaller they require less data to travel over the wire.

The malware we are dealing with was packed with UPX. After confirming that the malware was an executable PE file, using a tool called Stud_PE verified that the file was packed with UPX. Unpacking the file was trickier however, as the malware author took precautions against unpacking. More information is in section 2.4 around the unpacking.

4. Describe the malware's behavior. What files does it drop? What registry keys does it create and/or modify? What network connections does it create? How does it auto-start, etc?

Detailed analysis was performed to reveal the malware's behavior and reveal its entire functionality.

Malware's Behavior

- The malware is a network-aware worm that attempts to replicate across the existing network(s) by exploiting weakly restricted shares (common for Randex family of worms)
- A malicious backdoor trojan that runs in the background and allows remote access to the compromised system
- Creates startup registry entry
- The malware talks back to the command and control server under the name "**Crxbot Alias REalmbot" by Lindem**.
- The bot reports information back to the Command and Control (C&C) server. This includes reporting to the C&C server if smtp is not enabled, if the bot is attacking or participating in an attack, download a URL or even if it created further clones.
- The malware scans for and can spread across the following weakly restricted network shares Admin$, C$, D$ and IPC$
- The malware has reporting ability on the infections (cloning) etc. This data is used by the C&C to take payment for any malicious activity using "**e-gold**"
- Even creates a custom HTML reporting page for the C&C or malware author including timestamps, files, directories etc. Not standards compliant though with HTTP/1.0 and uses tables
- Microsoft Visual C++ Runtime Library was used
- The malware drops a file called Winsec32.exe in the windows system folder.

Registry Keys added/modified

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
HKEY_CURRENT_USER\Software\Microsoft\OLE
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]+ Microsoft Svchost
local services = "Winsec32.exe"
```

Network connections created.

The following Host Name was requested from a host database:

* testirc1.sh1xy2bg.NET

How does it autostart?

The changes also include new registry keys added which ensure that the malware auto-starts every time the infection host starts up.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices]
+ Microsoft Svchost local services = "Winsec32.exe"

{HKEY_CURRENT_USER\Software\Microsoft\OLE] + Microsoft Svchost local services =
"Winsec32.exe"
```

5. What type of command and control server does the malware use? Describe the server and interface this malware uses as well as the domains and URLs accessed by the malware.

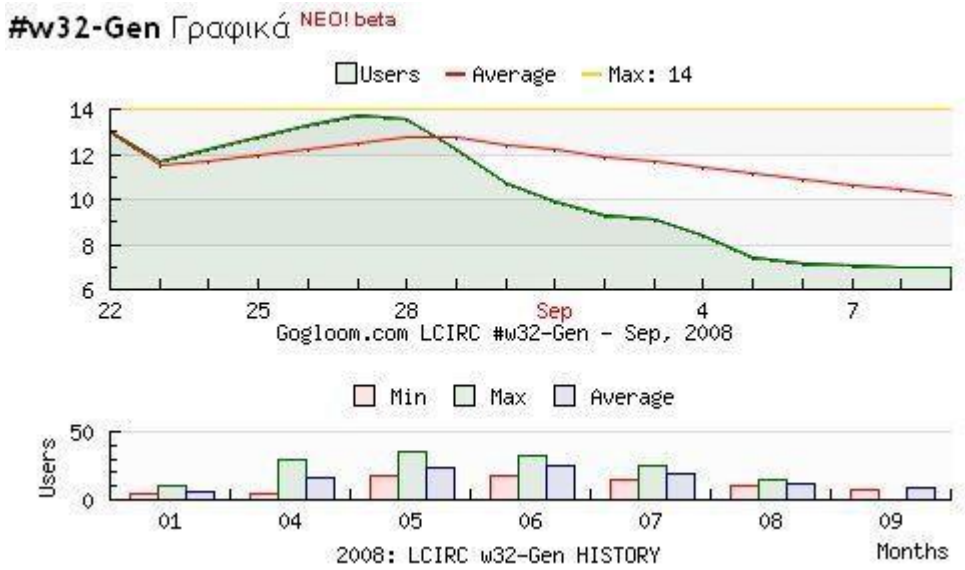
The malware drops winsec32.exe which acts as an IRC controlled backdoor.

- It attempts to connect to a predefined IRC server, in this case testirc1.sh1xy2bg.net. It
- attempts to join a specific channel so that the infected machine can be controlled. Once connected to the channel, the malware acts as a bot and waits for commands from any of a specified list of authorized users. In this case, the channel is challenge happy12
- Similar to most IRC channels which bots control, this channel is password protected. The userid and password are hardcoded into the malware. In this case, the userid is Nick with password gemp123

Domains/URLs accessed by the malware

- http://www.nivdav.net/winsec32.exe (expired
- domain!) testirc1.sh1xy2bg.net http://www.w32-
- gen.us

The last one was very interesting. While you cannot log onto the domain, there exists an IRC channel by the same name. These guys sell bots for paypal and others. They even shared the success of this particular bot



6. What commands are present within the malware and what do they do? If possible, take control of the malware and run some of these commands, documenting how you did it.

Commands present within the malware

Besides commands to perform DDos attacks, SYN attacks etc, some other commands present in the malware are

```
!op , !deop, !rnick, !ban, !stop, !run, !clone, !who, !server, !join, !rehash,
!take, !let, !me, !ame, !quit, !say, !msg, -msg, !mode, !cycle,
!seen, !exit, !raw, !info, !alias, !list,
```

Some of the commands that I executed:

On the IRC, a /list command was executed. This command listed the available channels on the server which was just the single instance of #happy12 chalenge.

The /who command was also executed which showed that the only user, other than myself, logged into the IRC server was the mIRC bot

7. How would you classify this malware? Why?

I would classify the packed malware.exe as a Trojan with the payload of an IRC bot. A trojan in this sense is any piece of code that masquerades as something that it is not. I would probably revoke the classification of trojan and simply call it an IRC bot under two conditions:

a. The computer operator installed it with both the knowledge and understanding of what would result, or
b. The software did not gain entrance to the system masquerading as something else (for example if attackers accessed the system via other means and simply loaded malware.exe onto the disk) However, since malware.exe spawns other entities (winsec32.exe), I would classify this individual component slightly different. Since malware.exe is an IRC program (used in bot-like fashion) that masquerades as a legitimate Windows system process, it would be both a trojan and an IRC bot with capability to spread over a network exploiting various Windows vulnerabilities.

8. What do you think the purpose of this malware is?

What this type of malware can do is a much broader question. The only limit to a trojan's capabilities is the expertise and creativity of its author. An IRC bot can also provide a variety of functionality for the attackers. Here is a short list based on the analysis performed:

- Fetch and install other software on the system
- Participate in DoS and/or DDos attacks on other networks and hosts
- Transfer private data off the infected machine
- Scan networks for other vulnerabilities and report results
- Runs in the background and allows remote access to the compromised system
- A network aware worm that attempts to replicate across existing network(s)

9. Bonus Question: Is it possible to find the malware's source code? If so, how did you do it?

Malware is usually developed in a high level programming language such as C or C++ which is then compiled into machine code translating into a stand-along executable program. However, it is not unusual to see malicious code, written completely in assembly language.

In my effort to look for the source, I tried looking at

- Disassembled the malware to look for hints of source code repository
- Googling for MD5 hashes of the various malware components spawned
- WWW

While the source code was not found, extensive insight into the malware was obtained by analyzing the malware's assembly code and running it through the various debugging tools

10. Bonus Question: How would you write a custom detection and removal tool to determine if the malware is present on the system and remove it?

The first step as part of the removal process is killing the appropriate processes and deleting the appropriate file and registry changes made by the malware. The malware in its current form is not dynamic and a batch script containing the registry keys to be removed and the processes to be killed would work. Care is also taken to ensure that the malware does not spawn again on restart.

This malware can also be detected by monitoring its network traffic and communications. This kind of communication can be detected by using a network based Intrusion Detection System (IDS). I would write custom snort rules to detect the trojan to scan packet contents for strings associated with the malware.
The rule set can also monitor for the presence of nickname change requests issued by a single IRC client every three seconds or so.

6 RESOURCES

1. Pedro Bueno's Malware Quizzes
2. Lenny Zeltser's Reverse Engineering Malware Paper
3. The Honeynet Project's Reverse Challenge
4. An Intro to Malware Analysis
5. Tom Liston's Malware series
6. Information Security summit 2008 Malware Challenge

7 TOOLS

1. http://gnuwin32.sourceforge.net/packages/wget.htm - wget for Windows from sourceforge
2. http://www.keyworks.net/keytools.htm - To decompile a CHM file

3. http://www.wireshark.org – To sniff network traffic
4. http://sourceforge.net/projects/regshot - Regshot is an open source registry compare utility that allows you to quickly take a snapshot of your registry and then compare it with a second one
5. http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx - Process monitor shows real time file system, registry, process/thread activity, TCP and UDP monitoring