

Malware Analysis Submission

Offensive Computing Malware Challenge

Submitted by:

Stephen Davis
Nick Harbour
Peter Silberman

| | |
|---------------|----------------------------------|
| Filename: | malware.exe |
| MD5: | 59A95F668E1BD00F30FE8C99AF675691 |
| Size: | 75,264 |
| Compile Time: | 2006/09/16 Sat 18:13:46 UTC |

1) Describe your malware lab.

The malware lab consisted of a Windows XP virtual machine running in host only mode. All network interactions of the malware are captured by monitoring the virtual network adapter provided by VMware which communicates only with the base operating system. To capture and emulate traffic to potential command and control servers, FakeDNS is used in virtual machine. The FakeDNS application is configured to respond to all DNS requests with the IP address of the base operating system. Wireshark is running when the malware launches to observe what types of network communication the malware is attempting to perform. Services requested by the malware may then be emulated by base operating system using netcat or custom scripts. Several tools in use on the guest malware analysis VM were:

- Strings
- ProcMon
- Process Explorer
- Wireshark
- FakeDNS
- IDA PRO
- Ollydbg (with OllyDump)
- PEID
- PEView
- FindEvil

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

The binary contains a compile time stamp which dates the specimen to September 16th 2006. Very few strings were observed in file, indicating that it may be packed. The Winsock DLL (WS2_32.dll) is imported indicating that the binary will likely perform network functions. The binary has very few imports which may be also be indicative of a packed binary.

3) Is the malware packed? If so, how did you determine

what it was?

The malware is packed. PEiD reports the second section to be highly entropic and a hardcore scan will reveal that it is packed with a mangled form of UPX. FindEvil also reports that a very small amount of the binary is valid x86 disassembly. The first section (beginning at RVA 0x1000) contains data in the file and is expanded in memory to be a large section, a common technique used by packers to create a landing zone for the unpacked code. The packed binary contains very few imports which is also indicative of a packed binary.

The binary can be easily unpacked with OllyDbg and the OllyDump plugin. The original entry point is 0x004109CC.

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?

Files Dropped:

- C:\Windows\Winsec32.exe

Potentially Dropped Files (Special command execution and are immediately deleted):

- C:\a.bat
- %temp%\1.reg

Registry keys:

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
 - Key: "Microsoft Svchost local services"
 - Value: Winsec32.exe
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
 - Key: "Microsoft Svchost local services"
 - Value: Winsec32.exe
- HKEY_CURRENT_USER\SOFTWARE\Microsoft\OLE
 - Key: "Microsoft Svchost local services"
 - Value: Winsec32.exe

Auto Starts:

- See registry keys for auto starting technique

Network Communication:

- Attempts to connect to IRC server testirc1.sh1xy2bg.NET:6667

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 uses an IRC client to connect to a server where the malware owner can control the IRC bots. The malware checks for the host name of *@legalize.it after the .login command is sent to the infected host. The password is "gemp123". The bot attempts to join the room #challenge. The bot name consists of the <country code>[windows version]<random 6 alphanumeric>.

Command List:

- login <password>

- Enables a master user to the botnet herd
- rndnick:
 - changes infected machine to random nickname
- logout:
 - bot logs out of room
- versionship/ver:
 - gets the current bot malware from a host and reports to master
- chghttp <new http host>
 - change the host of the http server to <new http host>.
- secure/sec:
 - Secure the machine, the following actions are taken:
 - Disable DCOM
 - Restrict access to IPC\$ share
 - deletes shares
 - C\$
 - If the machine is already in secure mode then it unsecures the machine by undo what it did to secure it
- lockdown.off/ld.off:
 - unsecures the bot same as if secure was called to secure the box, and then secure was called again (which unsecures it)
- visit/irc.v <url>:
 - Visits a given website specified by <url>
- web.on | httpd.on <port> <directory to share>:
 - Starts a web server listening on <port> sharing <directory to share>
 - if no <directory to share> is specified then it shares c:\Windows
- web.off/httpd.off
 - Stops the web server
- ftpd.on:
 - Starts an FTP server. The FTP server supports the following commands:
 - USER
 - PASS
 - SYST
 - REST
 - PWD
 - TYPE
 - PASV
 - LIST
 - PORT
 - RETR
 - QUIT
- ftpd.off:
 - Stops the FTP server

- log.off:
 - Turns logging off
- proxy.redirect.off:
 - Turns off proxy redirection
- ddos.off:
 - Stop DDOS attack
- syn.off:
 - Stop Syn Flood
- udp.off:
 - Stop UDP flood
- ping.off:
 - Stop the ping flood
- proc.off/com.ps.off:
 - stop process listing thread
- clone.off:
 - disables a created irc clone
- secure.stop:
 - Stops the secure thread
- scanstop:
 - Stops exploitation scan
- stats/st:
 - List exploit stats
- reconnect/irc.r:
 - reconnect to IRC server
- disconnect/irc.d:
 - disconnect from IRC server
- quit/irc.q <server>:
 - Quit from <server>
- status/irc.s:
 - Get bot status
- id/irc.i:
 - get bot ID
- reboot:
 - Reboot system
- threads/threads.l:
 - List the bots current running threads

- addalias/irc.aa <alias>
 - Adds alias name to bot
- log/irc.lg:
 - Show RealmBot log
- clearlog/clg:
 - clears the Realmbot log
- netinfo/ni:
 - Print network information:
 - Type
 - IP Address
 - HostName
- supersyn <ip> <port> <length>:
 - Starts a SYN flood to <ip> on <port> of <length>
- sysinfo/sys:
 - Lists system information:
 - OS Major Version (Minor Version)
 - Current user
 - Hostname
 - IP Address
 - Windows Directory
 - Date
 - Time
 - Uptime
 - CPU
 - RAM used
 - RAM free
 - Disk used
 - Disk Free
- remove/rm:
 - Uninstall the bot
- proc.on/com.ps <type>:
 - Display running processes, <type> can be full which will dump processes and loaded dlls.
- uptime/com.up:
 - Get computer's uptime
- driveinfo/com.driv <drive>:
 - Get information on a <drive>
- testdlls/com.dll:
 - checks to make sure dlls that are needed for bot operations are loaded
- opencmd/cmd1:
 - Opens command shell
- closecmd:

- closes command shell
- irc.who:
 - List those logged in
- flusharp/farp:
 - Flush arp cache
- flushdns/util.fdns:
 - Flush DNS cache
- currentip/cip:
 - get current IP
- irc.nick/irc.n <nick>:
 - Change nick to <nick>
- join/irc.j <channel>:
 - Join <channel>
- part/irc.pt <channel>:
 - Part <channel>
- raw/irc.ra <cmd>:
 - Execute IRC raw command
- killthreads/killit <all>:
 - Will kill threads if <all> is specified will kill all threads. Otherwise <all> is an integer specifying threads to kill.
- clone.quit/clone.q <thread>:
 - Causes a clone to quite
- clone.rndnick/clone.rn <thread>:
 - causes a cloned user to rename their name
- prefix/irc.pr <prefix>:
 - Change prefix
- open/com.o <file>:
 - Open <file>
- setserver/irc.se <server>:
 - Set bot to change to <server>
- dns/irc.dn <hostname>:
 - perform dns lookup on <hostname>
- killprocess/kpc <name>:
 - Terminates a process by name
- prockillid/pkid <pid>:
 - Terminates a process by PID

- delete/del <file>:
 - Deletes a specified file
- mirc.cmd <command>:
 - Issues a command to the mIRC window (if any) on the victim system
- cmd/cmd1 <command>:
 - Send a command to the current cmd.exe shell if it is established.
- list/com.fl <directory>:
 - Lists files in a specified directory
- readfile/com.rf <filename>:
 - Sends a specified file to the controller via a private IRC message
- ident <on/off>:
 - Enables or disables a thread which identifies itself to the server
- keylog.on/cmd.kl.on <mode>:
 - Enables a keystroke logger thread to capture either all keystrokes or keystrokes from specific websites.
- net <net command>:
 - Execute a windows networking command with the same syntax to the "net" command in windows.
- gethost/irc.gh:
 - Return IP address and Hostname for a system
- addalias/irc.aa <alias>:
 - add alias to alias list
- privmsg/irc.pm <user> <message>:
 - Sends a private message to another user
- action/irc.ac:<user> <irc action>:
 - Sends an irc action to another user
- cycle/irc.cy <sleep time> <channel>:
 - Forces bot to leave room for x seconds and then reconnect to a channel
- mode/irc.m <user> <mode switches>:
 - Sets IRC modes for a given bot
- rawclone/clone.ra <thread ID> <user>:
 - Creates an clone in the channel of a bot
- clone.mode/clone.m <thread ID> <mode switch>:
 - Sets mode of given clone
- clone.nick/clone.ni <thread ID> <nick>:
 - Changes an clones nick
- clone.part/clone.p <thread ID> <part msg>:

- Parts a bot from a channel with a message
- irc.repeat/irc.rp <integer> <repeat msg>:
 - Repeat some value
- getclip/com.gc:
 - Get victims clipboard data
- delay/irc.de <sleep len> :
 - Cause the bot malware to Sleep()
- update <url> <bot id> <crc> <file length>:
 - Updates malware on a given host
- execute/com.e <show windows> <command> :
 - Executes a file on the host
- rename/com.mv <filename> <new name>
 - Moves a file with a new filename
- clone.make/clone.start <host> <port> <channel> <channel pass>:
 - Creates a clone in a given irc channel
- synflood/ddos.ack/ddos.random <ip> <port> <length>:
 - Starts syn ddos attack
- download/dl <url> <dest> <run>:
 - Downloads a file from a remote URL saves it to a destination and can be executed
- redirect/daemon.rd <local port> <redirected destination ip> <redirected port>:
 - Simple TCP redirection on host
- clone.privmsg/clone.pm <thread ID> <nick> <message> :
 - Sends a privmsg from a clone
- clone.action/clone.ac <thread ID> <nick> <message>:
 - Sends an IRC action description from a clone
- advscan/asc <port> <thread id> <delay> <minutes>:
 - Uses the exploit of VNC to spread the IRC bot among local subnets.
- udpflood/ddos.udpf/u <ip> <number> <size> <delay> <port>:
 - Starts udp flood
- pingflood/ddos.pingf/p <ip> <number> <size> <delay>:
 - Starts ping flood
- httpcon/util.hcom <host> <port> <method> <URL> <referer>:
 - Opens an HTTP connection to a given host
- ftp.upload <host> <mode> <arg> <arg> <file>:
 - Uploads a file on an infected host to an ftp server

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.

The commands are documented as part of our answers to question 5. We took control of the malware by setting up an IRC server using a custom dns tool to redirect all traffic from the malware VM to the host machine. The IRCd was configured to allow us to change our host name using the /sethost command. We then changed our host to legalize.it. Alternatively, we could have patched the malware to skip over the password and host name check functions. After joining the channel, #challenge, we had our malware VM bot sitting in the room, we then gave the malware bot voice privileges and began the .login process using the password "gemp123". We were able to test out the commands listed above using either /privmsg or doing a channel wide command. In order for the bot to accept a command a "." must be precede any of the command strings listed above.

7) How would you classify this malware? Why?

This malware would be classified as an IRC Bot/worm. It has the command and control channel of an IRC bot. But has the ability to spread itself like a worm via a VNC exploit.

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

The purpose of this malware is to give the attacker control over the computer to potentially launch Distributed Denial of Service (DDOS) attacks or spread the bot further. The bot is capable of spreading using a VNC exploit which it can target random IP's or just go attacking the local subnet. The commands listed above give an attacker full control of an infected host, which can be used as a foothold to conduct further attackers, or as a denial of service tool. This malware is designed to be persistent with an updating feature for newer malware versions.

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

Yes the malware is open source. <http://darksun.ws/download/Bots/>

The string contained at 0x0041C7D4 : "Crxbot Alias REalmbot -by Lindem-" gave away what to google for:

<http://www.google.com/search?hl=en&q=crx+realmbot>

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

To detect the malware we wrote a python script. The python script checks the registry for the keys the malware installs to make itself persistent. It also checks to see if Winsec32.exe is currently running. If all these checks succeed then the detection tool reports a potential infection. We did not write the removal capabilities but that is also very trivial. To remove the malware just kill the process, delete the registry keys, and delete winsec32.exe.

