Global Information Assurance Certification Paper

Reverse Engineering of an Unknown Malware

Abstract: Reverse engineer an unknown piece of malware using strict methodology and industry standard tools. This process reveals the internals of the malware and how it may be used. I determine the following characteristics as detailed in my findings.

James Shewmaker

GREM Practical v1.0

Introduction

Presented with an msrll.zip file that contains a piece of unknown malware, I analyze the malware to determine use and purpose from the behavior and contents. The examination begins from a cursory analysis of the file, followed by a behavior analysis, and finally with a code level analysis.

Laboratory Setup

I use a compact but expandable system for analysis. It is set up for maximum flexibility and minimum collateral damage. The following include:

Host Environment

Shuttle xPC model SN41G2
Windows XP Pro
DVD-ROM
3.5" floppy
Athlon 2100+
512 MB RAM
160 GB Western Digital WD1600JB
120 GB Samsung SV1204H

VMware Guests

FreeBSD 4.7-RELEASE
Windows 98 Second Edition
Windows XP Pro SP1

Description

This small form-factor PC is portable and expandable, which is why I chose it over any laptop. It also includes a legacy RS-232 serial port (uncommon on recent consumer hardware). This serial port is handy for logging on to a server with a serial connection instead of using a network connection. This machine's features include typical onboard network, dual video adapters, PCI slot, USB 2.0, and IEEE 1394 (firewire). The DVD-ROM currently uses the 5.25" bay, but it may be removed for imaging drives easily.

To isolate any behavior and possible damage I use VMware's Workstation[1]. I have multiple VMware images for testing: Windows 98 Second Edition, Windows XP Professional, and FreeBSD 4.7. An original version is archived, and ready to update with the latest patches available. If for some reason I want to try an original version, I can use the archived image instead.

[1] VMware workstation product information and trial download: http://www.vmware.com/products/desktop/ws_features.html

The 160 GB drive is formatted NTFS and is used for the Windows XP host operating system, as well as storage for the VMware images. The 120 GB drive has wiped by the command `dd if=/dev/zero of=/dev/hdc` which copies a zero bit over each one on the drive. It is then formatted with ext2fs using default settings. This 120 GB drive is only used for working storage during imaging and mounting of images during testing.

For this analysis, I installed useful ports and software utilities (see Table 1) before moving the virtual machine to the host only network.

VMware Workstation on the host computer uses a virtual switch in Host-Only Mode to allow the target server and both test clients to communicate in an isolated environment. The clients also use a gateway of 192.168.242.2, but the host is not configured as a gateway. Internet addresses are assigned by the VMware's DHCP server and represented in Figure 1.

To move log files and the specimen to the environment, I temporarily plug the host XP machine into a live hub (while still clean), and download the specimen to the host onto the desktop. Any new files needed, and any interaction between the host and the guest machines will be used via the physical floppy drive, if only for the convenience of not adding network activity to the lab environment. After downloading of the specimen to the host computer, the network cable is completely removed to keep the lab as isolated as possible.

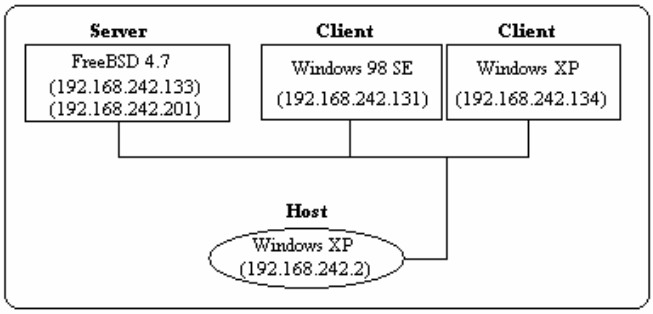

Figure 1

FreeBSD Image

The FreeBSD image provides a comfortable environment to begin analysis of a system or software[2]. I also use this image for any server side needs and for monitoring the virtual network. A generic kernel was used and all third party software is installed from the `/usr/ports` system. Appendix A contains the

[2] Instructions for obtaining, installing, and using the ports system with FreeBSD is available at http://www.freebsd.org/handbook/

output of the `dmesg` command, which shows what hardware was detected by the init process upon boot.

Software	Version	Platform	Description
BIND8	8.3.6	FreeBSD	commonly used DNS server
BitchX	1.1	FreeBSD	commonly used IRC client
ircd-hybrid	7.0.2	FreeBSD	commonly used IRC server
nmap	3.77	FreeBSD	utility to scan the network
netcat	1.10_2	FreeBSD	utility to read and write data across the network
openssl	0.9.7e_1	FreeBSD	opensource toolkit for SSL
snort	2.2.0	FreeBSD	utilty for network intrusion detection and sniffing
unzip	5.51	FreeBSD and Windows	utility to extract from .ZIP files
Filemon	6.12	Windows	utility to monitor file activity
IDAPro	4.52	Windows	disassembler for executable files
Ollydbg	1.10	Windows	debugger for executable files
Ollydmp	2.21b	Windows	plug-in to dump a process to file
Regmon	6.12	Windows	utility to monitor registry activity
TDImon	1.01	Windows	utility to monitor network activity
VMware Workstation	4.52	Windows	virtual machine engine to isolate the test environment

Table 1

Windows XP Pro Image

The primary testing image I use is a Windows XP Professional Service Pack 1 installation. I installed Regmon, Filemon, and TDImon from Sysinternals[3]. These utilities can monitor various aspects of the testing image as it changes. For binary analysis, I have installed Ollydbg[4] for debugging. In the code analysis I also make use of Ollydump, a plug-in for Ollydbg that allows me to dump process to a file. All defaults were used when installing. A list of hardware and drivers used on this machine is included in Appendix A.

Windows 98 Second Edition Image

The secondary test image is a Windows 98 Second Edition image patched to the latest, with the same software installed as the Windows XP image. This image is handy to test for different behavior that I may find on an older system. This

[3] These and many other useful tools are available at http://www.sysinternals.com/ntw2k/utilities.shtml
[4] The Ollydbg homepage is at http://home.t-online.de/home/Ollydbg/

image is less useful for debugging than Windows XP, as there are less known anti-debugging tricks[5]. As with the other guest operating systems, the System hardware listing, is included in Appendix A.

Properties of the Malware Specimen

I begin the analysis in the FreeBSD virtual machine, so I can examine the binary file in a comfortable and clean environment. First, I format a standard floppy, copy the binary file to the floppy, and move the write-protect tab to the read-only setting. This extra precaution helps me keep the original file intact while testing with different machines.

Now that I have the file on a floppy, I enable the floppy drive for the virtual machine, boot up the FreeBSD virtual machine, and put the floppy in my virtual lab's floppy drive. Once I login, I mount the floppy read-only, then copy and verify the binary file. I also run the file command on the zip file and its contents to ensure the files match their extensions. Figure 2 shows the commands and their results. It appears to be a Microsoft Windows Portable Executable file. One thing I see immediately that the file size did not change much while zipped, so it is likely that the file is compressed or encrypted in someway already.

```
#mount -t msdos -o ro,noexec,nodev,noatime /dev/fd0 /mnt/floppy
#file /mnt/floppy/msrll.zip
/mnt/floppy/msrll.zip: Zip archive data, at least v2.0 to extract
#unzip /mnt/floppy/msrll.zip
Archive:  /mnt/floppy/msrll.zip
[/mnt/floppy/msrll.zip] msrll.exe password:
  inflating: msrll.exe
#file msrll.exe
msrll.exe: MS Windows PE Intel 80386 GUI executable not relocatable
#ls -l /mnt/floppy/msrll.zip
-rwxr-xr-x  1 root  wheel  39100 Nov 20 13:21 /mnt/floppy/msrll.zip
#ls -l msrll.exe
-rw-r--r--  1 root  wheel  41984 May 10  2004 msrll.exe
#md5 /mnt/floppy/msrll.zip
MD5 (/mnt/floppy/msrll.zip) = 696c78651244b1ad0363a400a23d48ef
#md5 msrll.exe
MD5 (msrll.exe) = 84acfe96a98590813413122c12c11aaa
#
```

Figure 2

As nothing appears out of the ordinary, I examine this file with the strings command to pull out any labels or other human readable strings of characters. If I find an interesting string, I can use that information to analyze the file in further detail. Potentially interesting ones are included below. During the code analysis I uncover more strings, which are included in Appendix B. Also during the code

[5] Červeň, p 96

analysis, I find that this binary is packed with Aspack to both compress and hide the actual code.

```
!This program cannot be run in DOS mode.
VirtualAlloc
VirtualFree
kernel32.dll
ExitProcess
user32.dll
MessageBoxA
wsprintfA
LOADER ERROR
The procedure entry point %s could not be located in the dynamic link
library %s
The ordinal %u could not be located in the dynamic link library %s
kernel32.dll
GetProcAddress
GetModuleHandleA
LoadLibraryA
advapi32.dll
msvcrt.dll
msvcrt.dll
shell32.dll
user32.dll
version.dll
wininet.dll
ws2_32.dll
AdjustTokenPrivileges
_itoa
__getmainargs
ShellExecuteA
DispatchMessageA
GetFileVersionInfoA
InternetCloseHandle
WSAGetLastError
```

These strings extracted from the malware tell me very little about our specimen. The dll references hint at what I should expect from the malware, though they seem fairly generic. The potentially interesting strings are in Table 2 below.

File	Purpose
Advapi32.dll	Access the registry
Msvcrt.dll	Microsoft C Runtime Library
shell32.dll	Command interface
wininet.dll	Win32 Internet library
ws2_32.dll	Win32 socket library version 2

Table 2

The key file properties of our msrll.exe malware are:

- `File type: MS Windows Aspack Executable Program`
- `Size: 41,984 bytes`
- `Md5sum: 84acfe96a98590813413122c12c11aaa`

Behavioral Analysis

To examine the behavior of this binary, I use my Windows XP virtual machine. I use Regmon, Filemon, and TDImon tools from Sysinternals to analyze the registry, file, and network activity respectively. These utilities start logging automatically, and I save their output to a floppy by clicking on the floppy icon. These logs files are then saved independently of the machine so I can revert to a clean state without losing my logs.

I place the `msrll.exe` on the virtual machine in the `C:\SPECIMEN\` folder via another standard floppy. I use the `cmd` command prompt to run the file with the full path. This allows for command-line arguments and watching for any output to the console.

To minimize polluting the logs with access of the other monitoring tools, I create a snapshot before executing the malware and running each test independently. Although this results in different process ID numbers, I can focus on the process itself instead of the monitoring tools tracking themselves.

On the FreeBSD server, I create a logging folder to isolate all snort logging in `/usr/local/test/mal-01/`. I start a snort process with the following command to records any network traffic and highlight known traffic signatures.

```
sort –vd –l /usr/local/test/mal-01/ | tee /usr/local/test/mal-01.log
```

The `–v` flag enables verbose mode, the `–vd` flag enables headers and application data[6] and the `–l` flag[7] logs to the new directory. Then the standard output is piped through the tee command to display the standard output to the screen as well as the `mal-01.log` file. This will provide a complete log of the network traffic, when TDImon only shows connection information of the host it is running on.

[6] Snort sniffing mode flags described at http://www.snort.org/docs/snort_manual/node4.html
[7] Snort logging flag described at http://www.snort.org/docs/snort_manual/node5.html

Registry Activity

Regmon shows a large amount of activity, so I will focus on the key areas. The first item that catches my eye is the creation of a service, highlighted below at time offset 2.16355542 in Figure 3. Windows XP uses services to run on the system level independent of any active user.

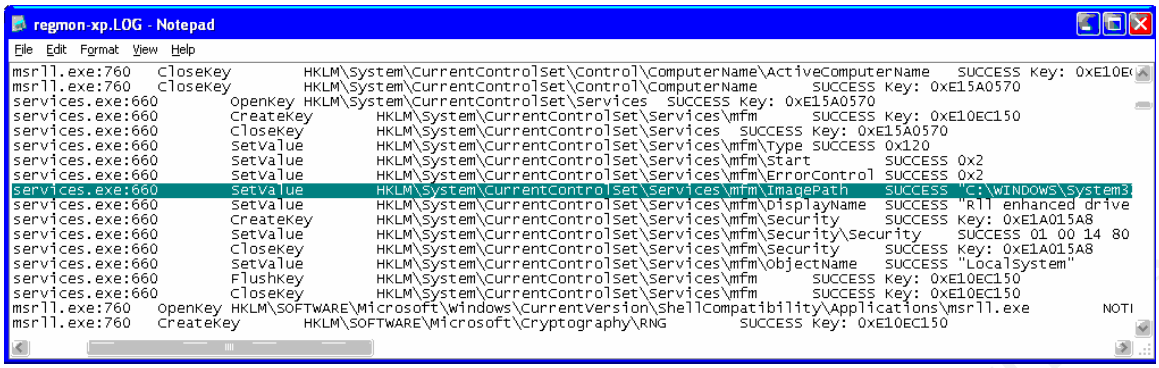

Figure 3

In Figure 4, I see the first msrll.exe process (process ID =760) start a second copy of itself (process ID=252).

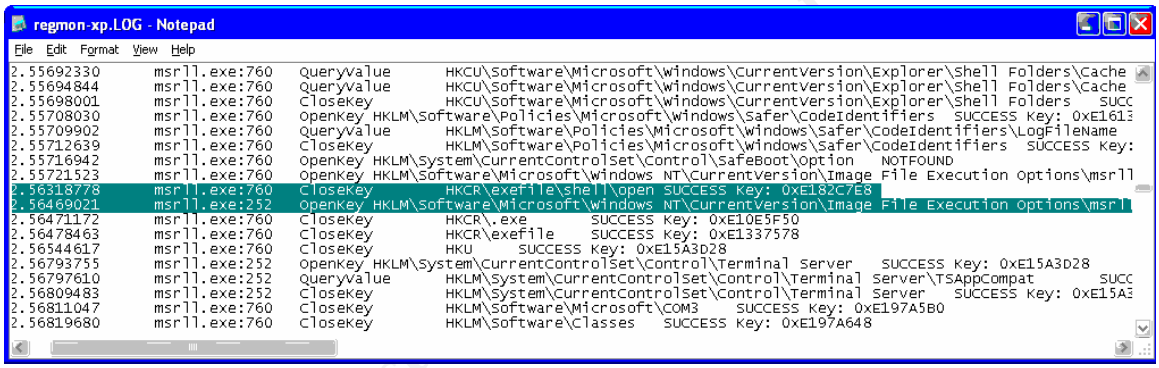

Figure 4

I also see what appears to be enumeration of Internet settings. This could be a natural side effect of typical network traffic. Highlighted below in Figure 5, I see Internet cache entries being set by the new msrll process.

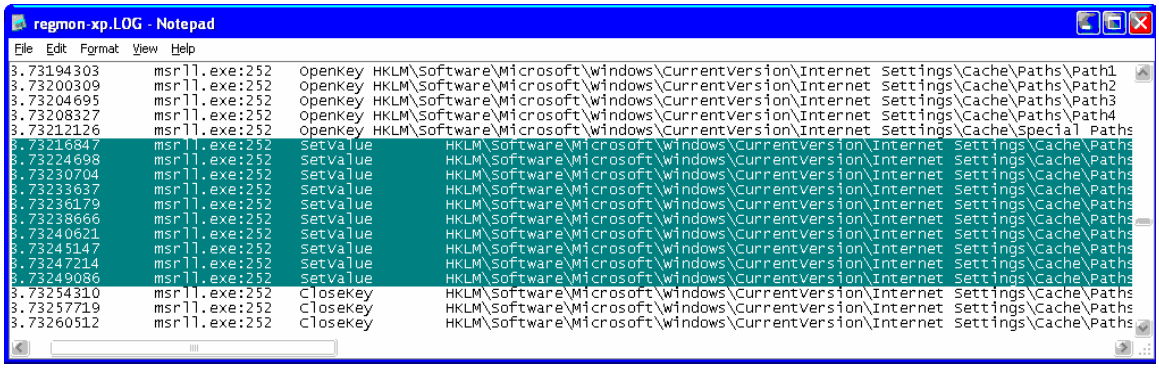

Figure 5

I also see this process examine network settings in the registry. Figure 6 shows the new process checking DNS client settings then successfully finding the hostname.

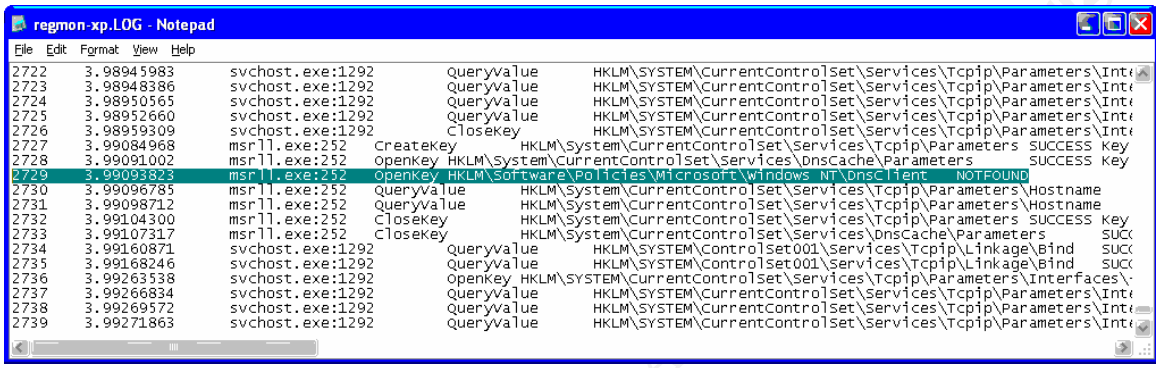

Figure 6

I see that the malware creates registry entries appropriate to run as a service `C:\WINOWS\System32\mfm\msrll.exe`. The registry activity of the malware shows network activity. I also see that two msrll.exe processes access the registry. There may be more registry activity depending on the use of the process, but without more information on its behavior I will not be able to trigger it.

File Activity

Filemon records the creation of the `C:\WINDOWS\SYSTEM32\MFM\` directory and creates a new `msrll.exe` file in Figure 7. Sometimes malware will hide as a legitimate system process as a disguise[8]. This malware is probably pretending to be a service related to MFM standard hard drives, the precursor to IDE[9].

[8] Skoudis, p. 257
[9] A summary of likely uses of the malware's disguise is at http://kb.indiana.edu/data/adlt.html?cust=4074217.7239.131

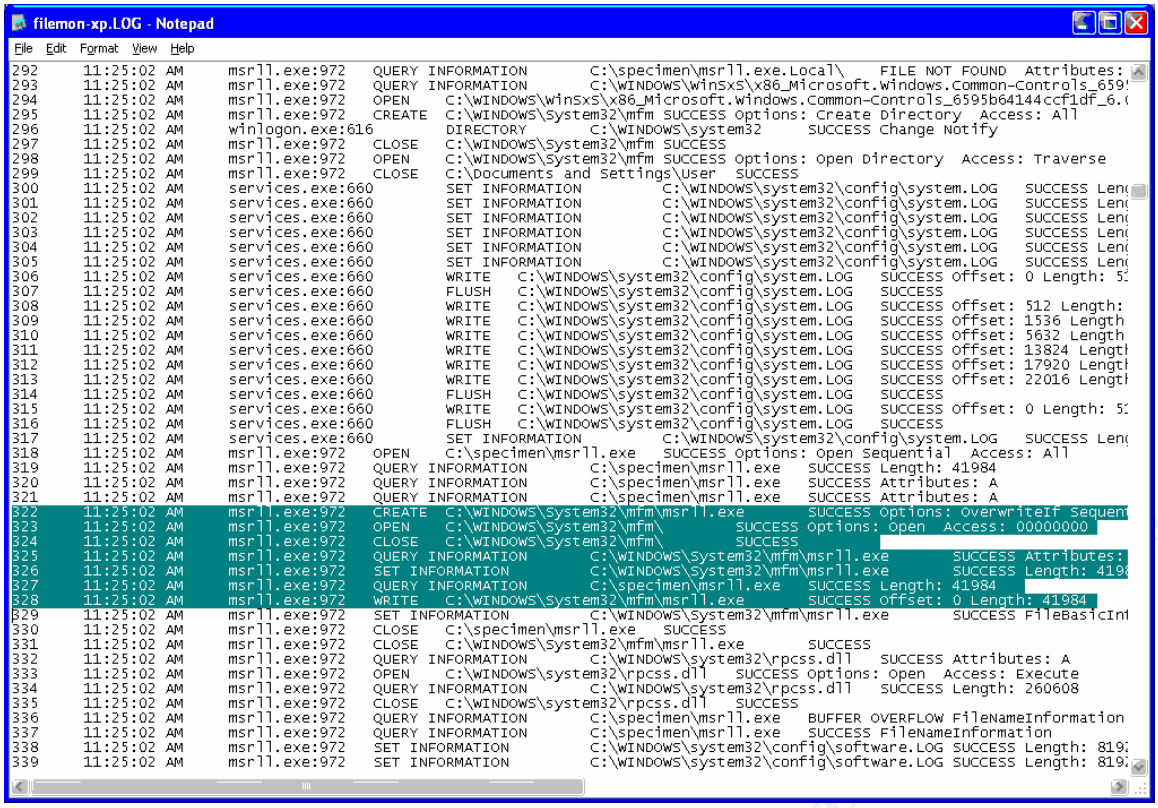

Figure 7

Filemon displays where the original process deletes itself then confirms it is unavailable by trying to read from the file (Figure 8).

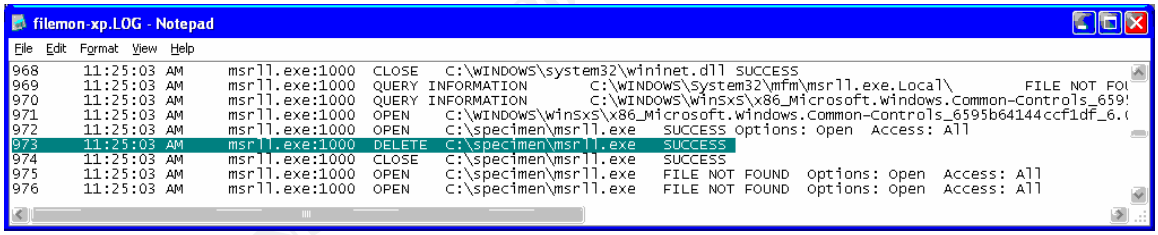

Figure 8

A `C:\WINDOWS\System32\mfm\jtram.conf` file is created, and I see more dll activity in Figure 8. Access to `rasadhlp.dll` and `rsaenh.dll` is also shown. This activity is significant because a normal executable would list its library use in an imports table and our strings output showed different libraries.

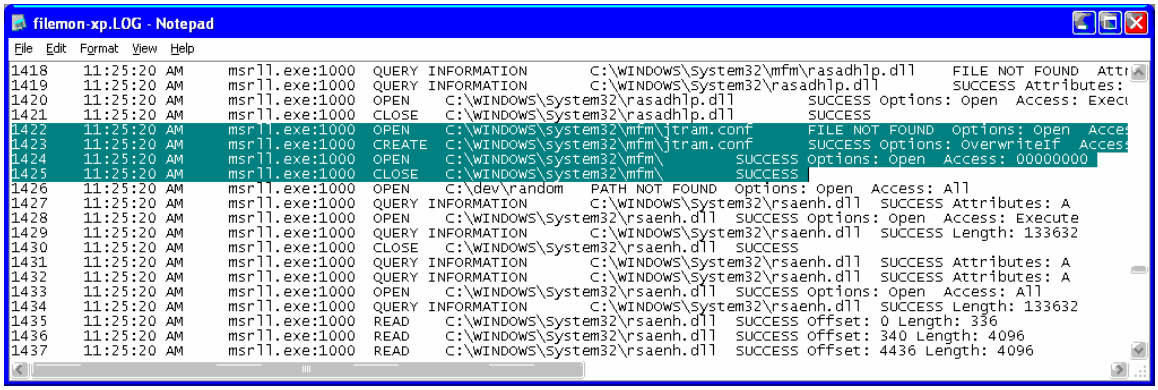

Figure 9

After quite a bit of `rsaenh.dll` use, I see where data is being written to this new file in Figure 10. The `C:\dev\random` OPEN entries may be significant as well, since it repeatedly tries and fails.

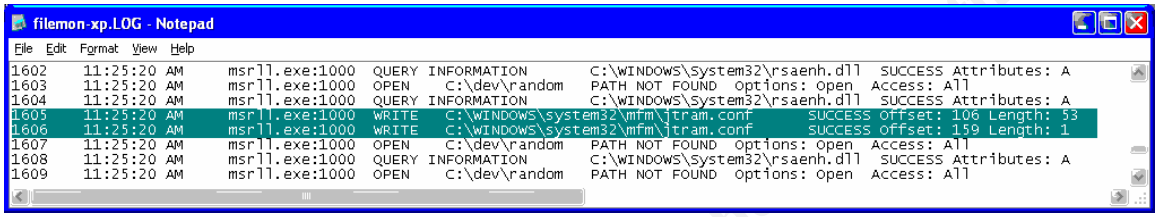

Figure 10

The `msrll.exe` file activity included creation of the `C:\WINDOWS\System32\mfm\` directory, creation of a new `msrll.exe` and `jtram.conf` files, dll usage, and `C:\dev\random` open attempts.

Network Activity

I monitor network activity with snort on the FreeBSD machine and TDImon on the Windows machine. Running TDImon ensures I see at least connection information of any network traffic on the client machine. Snort is used to gather all content of the packets as well as the header information.

One of the first things I see with TDImon is the malware opens a local port for listening. Figure 11 shows the necessary network changes involved while deploying this server on port 2200.

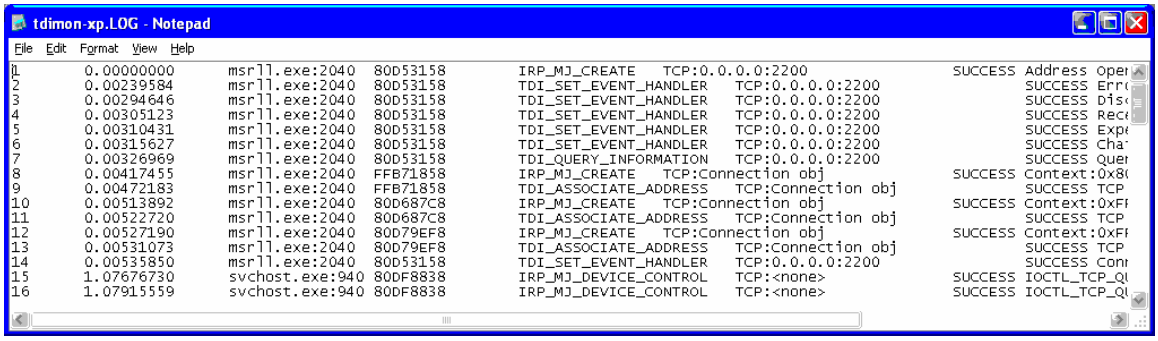

Figure 11

A DNS request and another service listening on port 113 (Figure 12), which is typically used for IDENT servers to identify which user on the system is connecting.

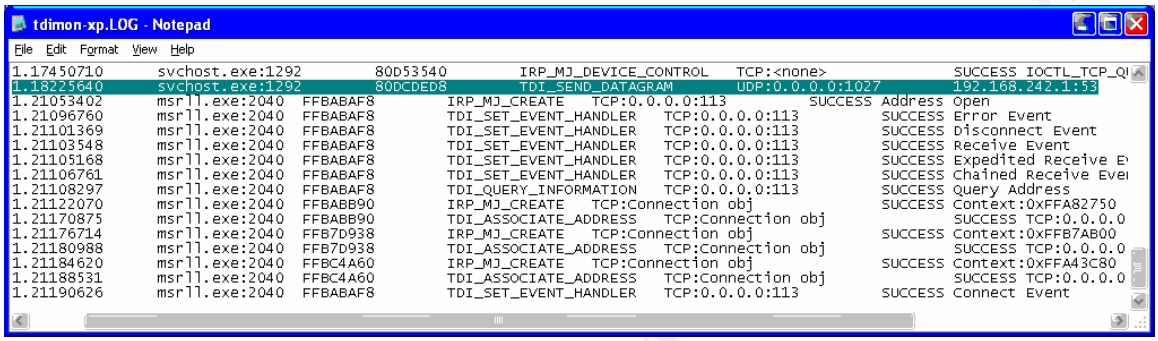

Figure 12

Snort captures the DNS requests: `collective7.zxy0.com` and `collective7.zxy0.test.com`. The appending of test.com is the natural progression of the DNS resolver. The Windows XP machine has a domain of test.com, so it tests to see if the hostname was intended to be in the local domain.

```
6D 00 00 01 00 01 00 00 29 10 00 00 00 00 00 00  m.......).......
00                                               .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/13-10:17:47.674225 192.168.242.134:1027 -> 192.168.242.133:53
UDP TTL:128 TOS:0x0 ID:451 IpLen:20 DgmLen:66
Len: 38
00 06 01 00 00 01 00 00 00 00 00 00 0B 63 6F 6C  .............col
6C 65 63 74 69 76 65 37 04 7A 78 79 30 03 63 6F  lective7.zxy0.co
6D 00 00 01 00 01                                m.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/13-10:17:48.673203 192.168.242.134:1027 -> 192.168.242.133:53
UDP TTL:128 TOS:0x0 ID:452 IpLen:20 DgmLen:66
Len: 38
00 06 01 00 00 01 00 00 00 00 00 00 0B 63 6F 6C  .............col
6C 65 63 74 69 76 65 37 04 7A 78 79 30 03 63 6F  lective7.zxy0.co
6D 00 00 01 00 01                                m.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/13-10:17:50.010840 192.168.242.133:1061 -> 193.0.14.129:53
UDP TTL:64 TOS:0x0 ID:230 IpLen:20 DgmLen:45
```

Figure 13

I add the alias of 192.168.242.201 to the FreeBSD network adapter by running `/sbin/ifconfig lnc0 alias 192.168.242.201 netmask 255.255.255.255`. I create a zone file for zxy0.com in `/etc/namedb/zxy0.txt` and edit `/etc/namedb/named.boot` to load the new zone file. I create the bare minimum zone and add an address record for `collective7` to point to this new IP address. After restarting my DNS process with `killall -HUP named` and reverting the Windows XP machine back to the clean state, I run the malware again to see what it tries to do with the DNS information.

Figure 14 shows TDImon recording the successful DNS lookup and tries to connect to port 6667 on the 192.168.242.201 IP address. This is a commonly used IRC port number, and IRC is a typical component of many malware samples.

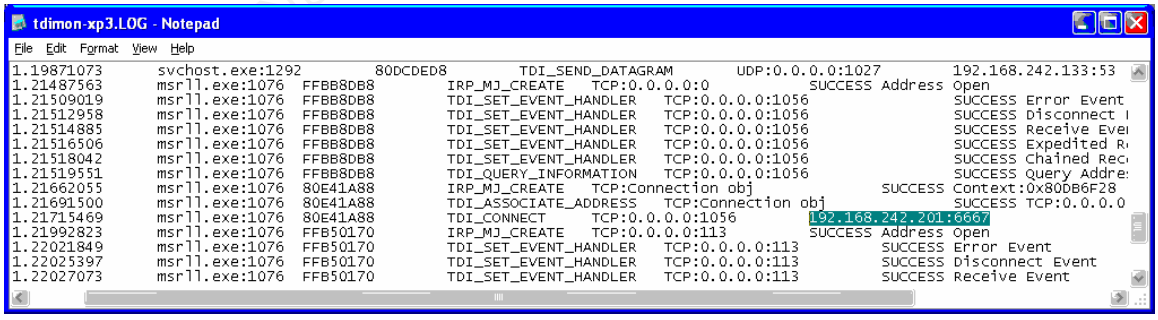

Figure 14

I launch a basic configuration of ircd-hybrid on the FreeBSD machine and start the malware process again. I also connect with a command line IRC client named BitchX to my new IRC server for observation as the user malobserv (commands are shown in Figure 15). The flags used are `-c` channel, `-H` host, `-p` port, `-n` nickname.

```
collective7# /usr/local/etc/rc.d/ircd-hybrid.sh start
ircd: version hybrid-7.0.2
ircd: pid 1027
ircd: running in background mode from /usr/local
 ircd
collective7# su jim
%./BitchX -c #mills -H 192.168.242.201 -p 6667 -n malobserv
```

Figure 15

The client once again tries to connect to port 6667 and is successful, and I also see the use of port 113 as the IDENT lookup on XP machine. The client's name appears random and it joins the #mils channel (see Figure 16 below).

```
12/13-10:40:03.440806 192.168.242.201:6667 -> 192.168.242.134:1056
TCP TTL:64 TOS:0x0 ID:602 IpLen:20 DgmLen:117 DF
***AP*** Seq: 0xFD4F0631  Ack: 0x50FF3D6D  Win: 0xE420  TcpLen: 20
3A 63 6F 6C 6C 65 63 74 69 76 65 37 2E 7A 78 79  :collective7.zxy
30 2E 63 6F 6D 20 33 30 32 20 59 67 6A 4D 53 52  0.com 302 YgjMSR
63 5A 54 20 3A 59 67 6A 4D 53 52 63 5A 54 3D 2B  cZT :YgjMSRcZT=+
52 4C 4D 55 50 50 4E 59 47 70 40 31 39 32 2E 31  RLMUPPNYGp@192.1
36 38 2E 32 34 32 2E 31 33 34 20 0D 0A           68.242.134 ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

12/13-10:40:03.545569 192.168.242.134:1056 -> 192.168.242.201:6667
TCP TTL:128 TOS:0x0 ID:454 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x50FF3D6D  Ack: 0xFD4F067E  Win: 0x43C8  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

12/13-10:40:35.455199 192.168.242.134:1056 -> 192.168.242.201:6667
TCP TTL:128 TOS:0x0 ID:455 IpLen:20 DgmLen:53 DF
***AP*** Seq: 0x50FF3D6D  Ack: 0xFD4F067E  Win: 0x43C8  TcpLen: 20
4A 4F 49 4E 20 23 6D 69 6C 73 20 3A 0A           JOIN #mils :.

:
```

Figure 16

The client only sends WHO and PING after the JOIN #mils command in Figure 17, as is normal when an initial IRC connection is made. It appears that the client is fully connected and waiting for the next step.

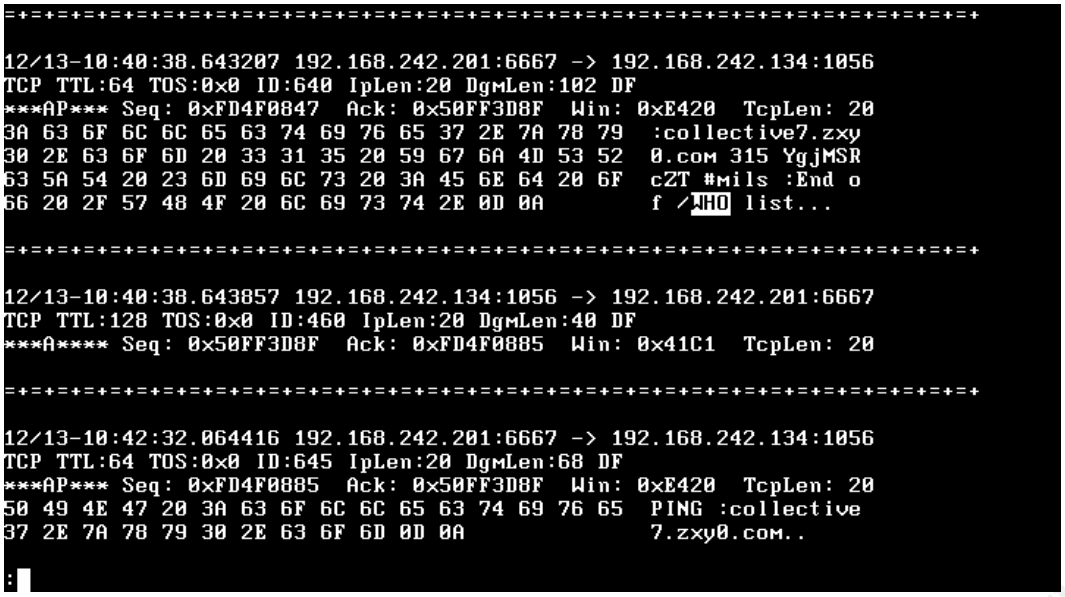

Figure 17

The client will join the #mils channel if I am already on the channel, but it seems to not respond to anything typed into the chat or with a private message (ie: `/msg YgjMSRcZT`). As I have ran out of clues to follow, and the monitoring tools have not uncovered any new ground I try to connect to the malware's service on port 2200.

When I connect a telnet client to its server on port 2200, I only get a prompt of `#=` and any attempts to send non-whitespace data result only in disconnection after the second whitespace. Trying to communicate with the new user on my ircd-hybrid server also has no response.

I generally observed the same behavior on the Windows 98 system. The differences were minimal, namely the use of `C:\WINDOWS\System\mfm\jtram.con` as the configuration which follows Windows 98 practices. Since Windows 98 does not have services in the same way that XP does, the malware sets a registry value (shown in Figure 18 below) to run the `C:\WINDOWS\System\mfm\msrll.exe` file on startup.

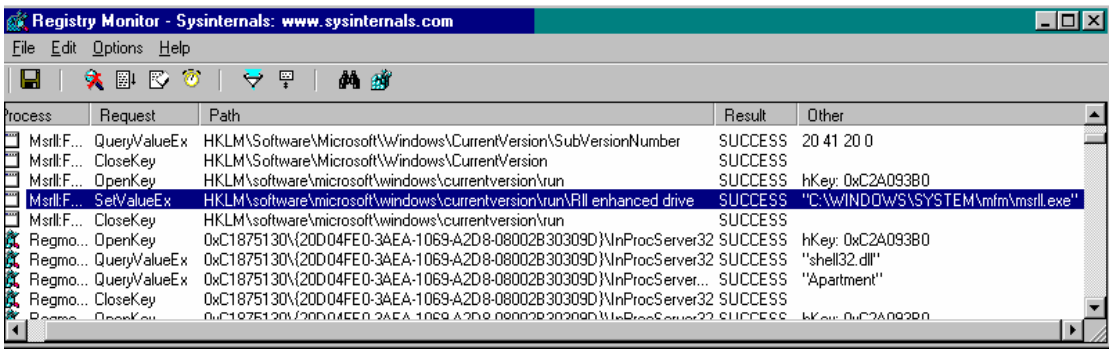

Figure 18

Now that I have watched individual behavior, I run the malware on both Windows clients. Neither client shows any different behavior when multiple copies are connected. I will continue the behavior analysis during the code analysis process.

I attempt to determine the services listening on the ports by running nmap from the FreeBSD server. Nmap is a great tool that can help identify machines and services over the network. I run nmap with the flags `-sV` for scan and verify services, `-p 1-` for scanning ports one and above, followed by the IP address of the client machine. In Figure 19, nmap was unable to determine the service listening on port 2200, calls the port 113 service "auth?" and correctly determines that the MAC address is in the range assigned to VMware. Port 139 is standard for Windows networking (SMB).

```
collective7# nmap -sV -p 1- 192.168.242.134

Starting nmap 3.77 ( http://www.insecure.org/nmap/ ) at 2005-12-15 12:29 GMT
Interesting ports on 192.168.242.134:
(The 65532 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE     VERSION
113/tcp  open  auth?
139/tcp  open  netbios-ssn
2200/tcp open  unknown
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at http://www.insecure.org/cgi-bin/servi
cefp-submit.cgi :
SF-Port2200-TCP:V=3.77%D=12/15%Time=43A161FE%P=i386-portbld-freebsd4.7%r(N
SF:ULL,2,"#:")%r(GenericLines,2,"#:")%r(GetRequest,2,"#:")%r(HTTPOptions,2
SF:,"#:")%r(RTSPRequest,2,"#:")%r(RPCCheck,2,"#:")%r(DNSVersionBindReq,2,"
SF:#:")%r(DNSStatusRequest,2,"#:")%r(Help,2,"#:")%r(SSLSessionReq,2,"#:")%
SF:r(SMBProgNeg,2,"#:")%r(X11Probe,2,"#:")%r(LPDString,2,"#:")%r(LDAPBindR
SF:eq,2,"#:")%r(LANDesk-RC,2,"#:")%r(TerminalServer,2,"#:")%r(NCP,2,"#:")%
SF:r(NotesRPC,2,"#:")%r(WMSRequest,2,"#:")%r(oracle-tns,2,"#:");
MAC Address: 00:0C:29:61:FF:04 (VMware)

Nmap run completed -- 1 IP address (1 host up) scanned in 166.875 seconds
collective7#
```

Figure 19

So far the malware has demonstrated the typical backdoor behavior of opening up a listening port. It also behaves like a Trojan where it attempts to masquerade as a RLL driver in the local services. To uncover any new info in our lab environment, I will have to combine the behavior analysis with analyzing the code of this malware.

Code Analysis

My code analysis process takes advantage of the monitoring tools I have already used, but now in conjunction with my debugger. I can run a procedure and observe the behavior immediately. This method quickly isolates the observed behavior to a certain part of the malware. I also insert comments into Ollydbg (by pressing the ";" key) as I see interesting information or anything that might be a procedure call. Even if I do not understand a part of the code, I at least label it so that I know at least I have seen the code before. I can peek ahead at a call or jump by selecting the instruction and pressing `Enter` then return to the paused instruction by pressing the `*` key. These simple steps help maintain an organized analysis.

Another important factor in this code analysis is that I keep the monitoring footprint to minimum impact. I use breakpoints sparingly and only run the monitoring tools when I am ready to observe that procedure. This removes a large number of anti-debugging and anti-disassembly tricks from complicating the analysis. However, it is still possible to check the Windows Registry or physical files if the software is not active, so I still prefer to animate through the malware code. Ollydbg provides `Ctrl+F7` and `Ctrl+F8` to animate into and animate over instructions respectively, which can be interrupted with a key press if I would like to pause at the current command. With these analysis concepts in mind, I begin to analyze the malware's internal code.

Initially I see that the compressed malware is only a few bytes more than the zipped file, which usually indicates that the internal file is already compressed or encrypted in some way. The strings output did not yield any connection information, so the malware must have at least that portion of itself hidden. I will step through the initial lines of code to gain a deeper understanding.

To step through the malware for code analysis I use Ollydbg. I open Ollydbg, select `File` then `Open` and select `C:\specimen\msrll.exe`. Ollydbg displays an abnormal entry point warning, shown in Figure 20.

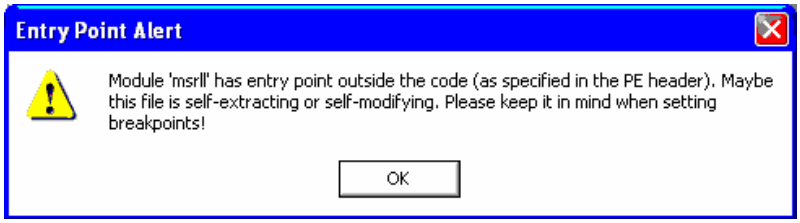

Figure 20

This warning means the executable does not start at a standard entry point. To view the malware's layout, I select `View` then `Memory`. I highlight the ".aspack" section (Figure 21) and begin an online search for "aspack".

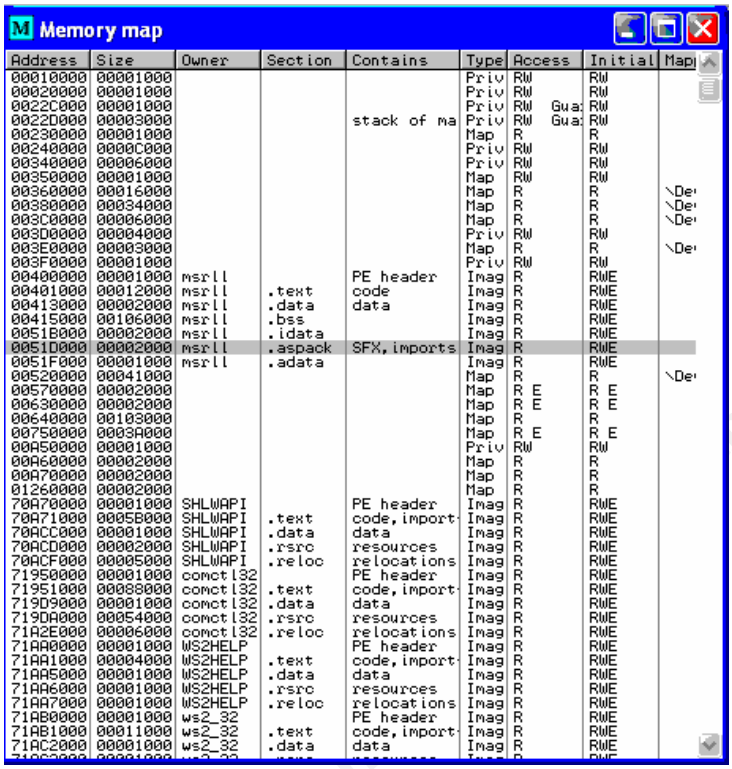

Figure 21

I find and try eight different unpackers in an effort to decode this malware. Eventually I was able to use "AspackDie 1.1" to unpack it[10] and attempt to open the file with Ollydbg. Unfortunately, Ollydbg will not load this unpacked malware.

In order to continue the code analysis, I unpack the code manually using Ollydbg and the Ollydump plug-in. I see from Figure 21 the PE header of the msrll file

[10] A list of utilities to unpack files protected with aspack is at http://www.exetools.com/unpackers.htm

starts at address x40000; this information will be necessary when we dump the unpacked file later.

Ollydbg initializes and places me at 0x51D001, which is what the memory map said was the entry point. I step once into the first instruction (PUSHAD) of the file by pressing `F7`. Now the next instruction is a function call. Since the file is packed, this call is likely the beginning of the unpack routine, so that the file can run normally when executed.

To see the malware unpacked, I need to pause the program after the unpack routine, but just before running the original program.[11] In the Registers windows, I select ESP with the mouse, right-click, and select `Follow in Dump` (Figure 22).

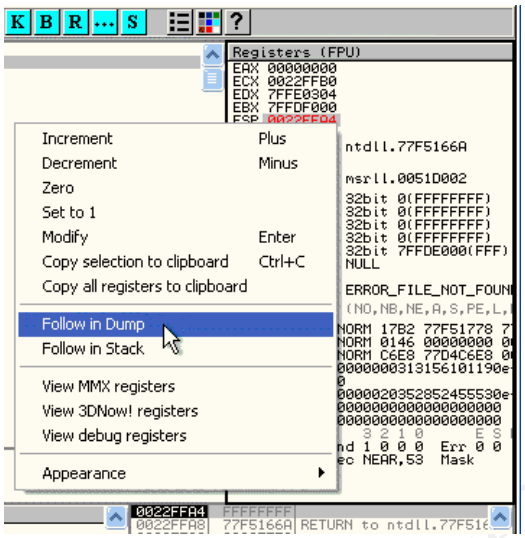

Figure 22

Now that Ollydbg has moved the Dump window to the new address (x0022FFA4), I select the first four bytes and right click, selecting `Breakpoint`, "`Hardware, on access`", and `Dword` (show in Figure 23). Now the program will break when the program reads these four bytes, which should be the first instruction of the program after this unpacking business.

[11] CrackZ confirms our OEP assumption quoting the Aspack author at http://www.woodmann.com/crackz/Packers.htm#aspack_asprotect

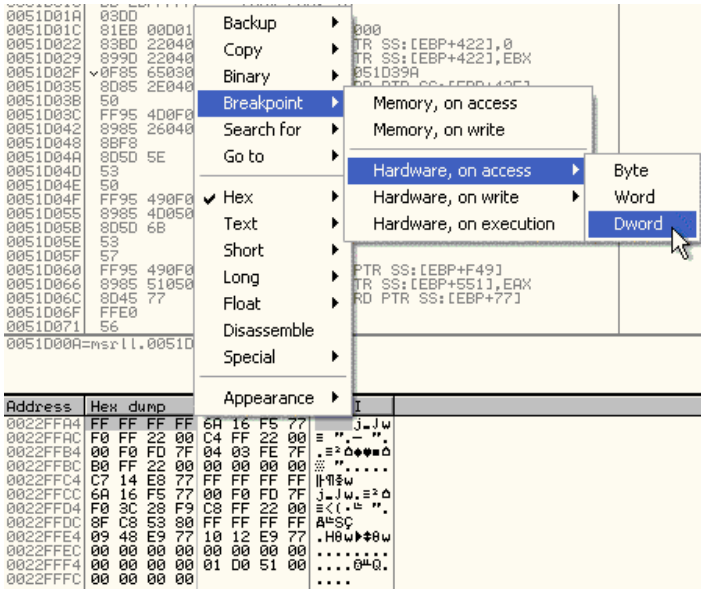

Figure 23

To execute to the breakpoint, I press F9. It breaks at address x51D3B0. I should be passing the unpacking code and nearing the start of the original msrll, now that my breakpoint has been triggered. I step into the jump and two more commands before Ollydbg stops on a "DB 6A" at x401240 (See Figure 24).

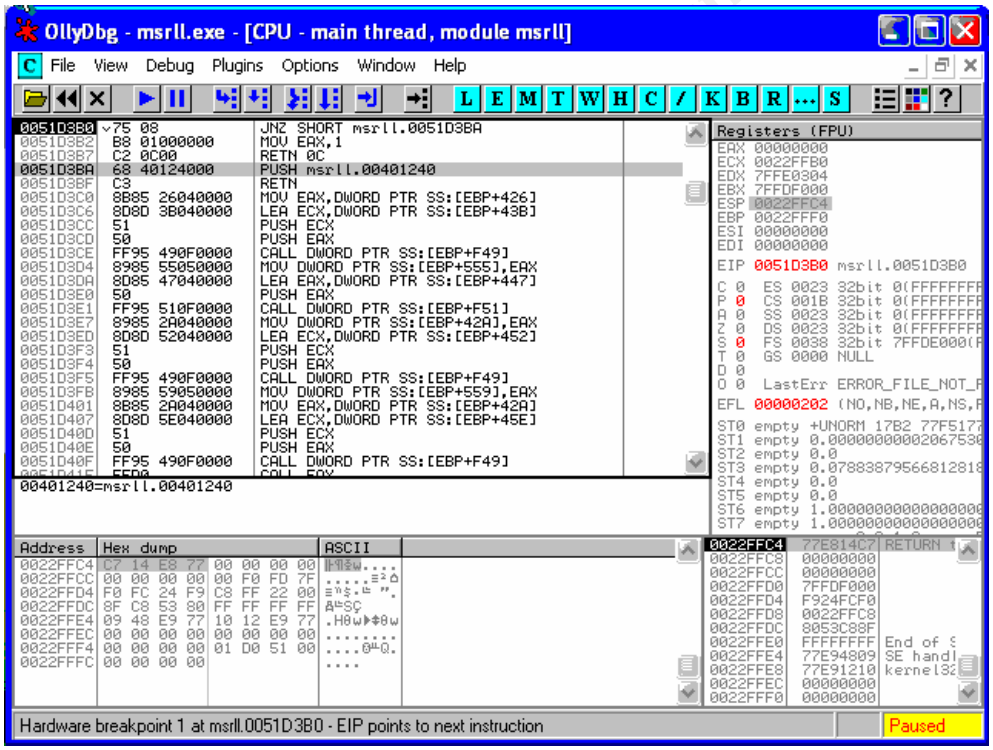

Figure 24

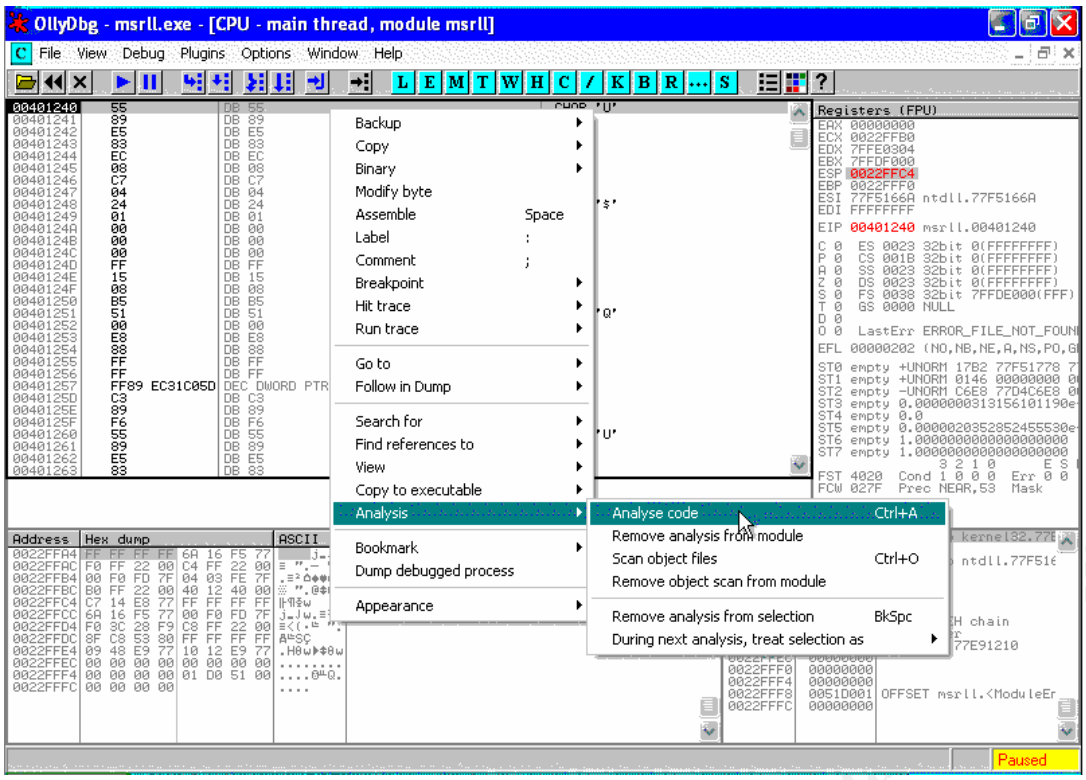

Figure 25

Since I am currently in the data section of the file, Ollydbg does not try to interpret the bytes into assembly, so I force it to analyze the code with another right-click menu from the first instruction (Figure 25 above). Now Ollydbg displays useful assembly code of what must be the original msrll program. A side by side comparison of the unanalyzed and analyzed code is in Figure 26. This address of x4101240 must be the original entry point (OEP).

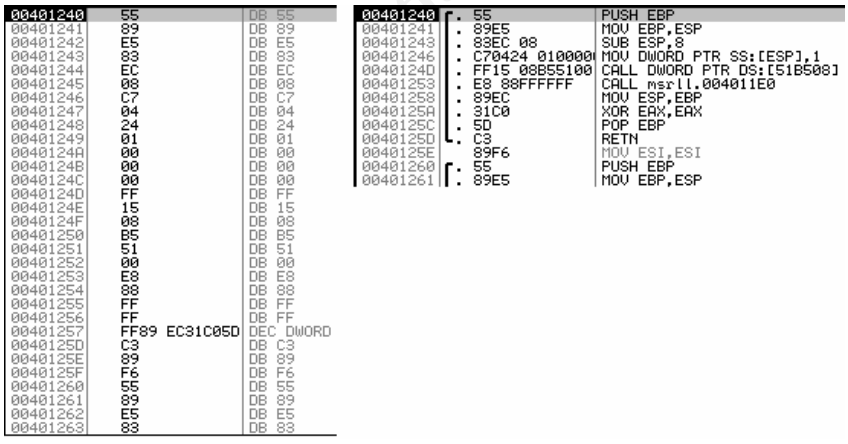

Figure 26

I can save my progress so far by dumping this unpacked code into a file. I use a plug-in for Ollydbg called Ollydump[12]. Ollydump automatically places my current instruction (EIP) in the OEP offset, and I save the dump to `C:\specimen\unpacked_msrll.exe`. I use Ollydbg's command on this new file, which reveals more intriguing items to watch while debugging, such as "PASS", "smurf," and "jolt2." The complete strings output of this unpacked file is included in Appendix B.

Now I continue to animate through the code, pausing and adding comments to subroutines as I encounter them. This is a long process but is the easiest way to find interesting items without making assumptions and possibly missing something important.

The loop beginning at x4118AE checks the full path of the executable. I see at x40BE66 that it starts building the string "mfm" which must be where the msrll.exe file is copied to the `C:\WINDOWS\System32\mfm\` directory and the jtram.conf file is created (see Figure 26 below).

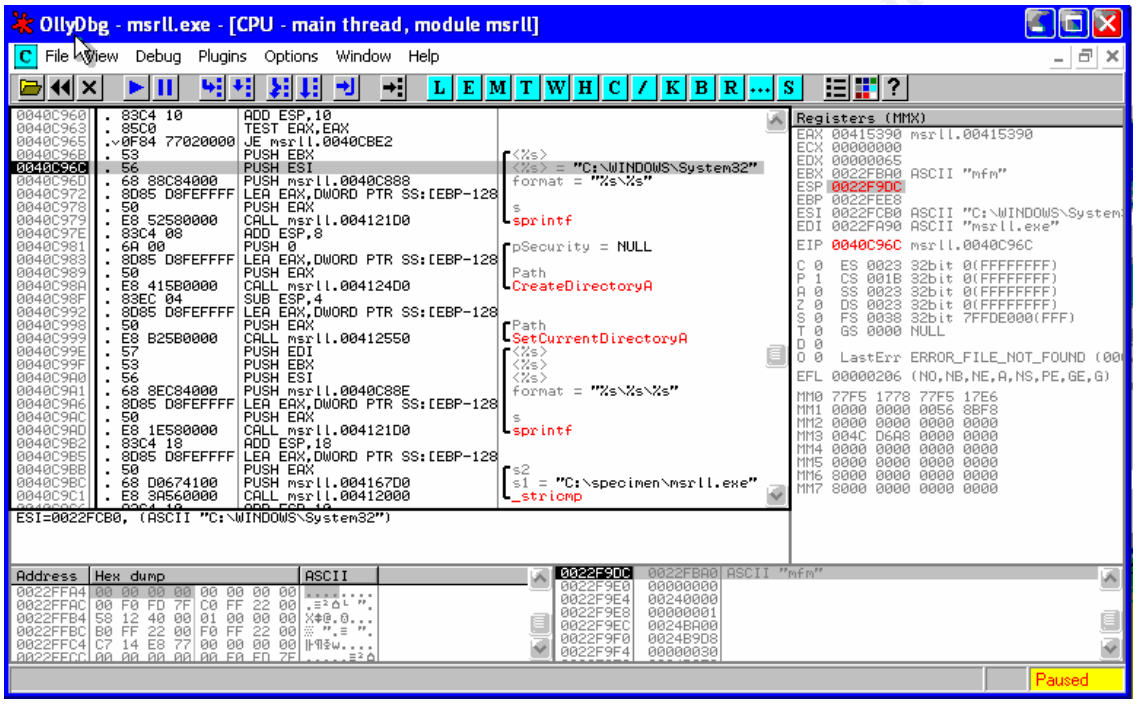

Figure 27

Also in Figure 27, I see the malware create a new copy of the file in this new folder. The malware runs the new copy and terminates the original copy. My first

[12] Ollydump and other Ollydbg plug-ins are available at http://ollydbg.win32asmcommunity.net/stuph/

impression of the `/d` command flag is to delete a file, in this case the original malware. Figure 28 shows deletion, execution, and exit of the current process.

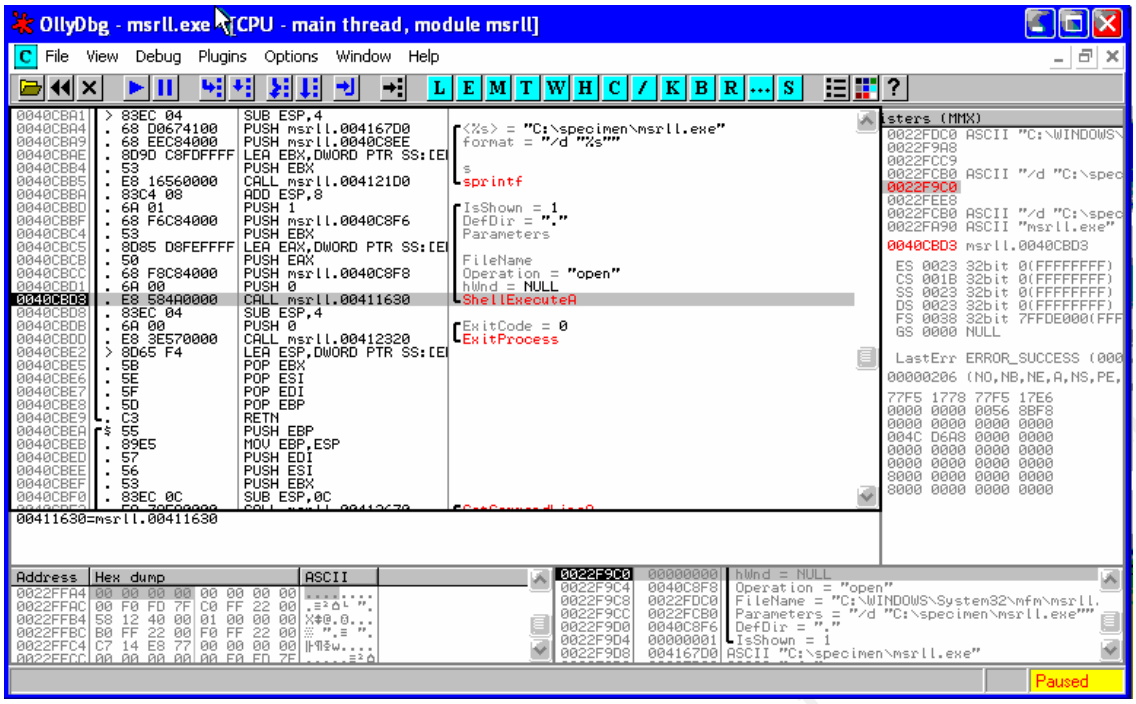

Figure 28

Since I was not tracing through this new file, it runs outside of the debugger as normal, resulting in the jtram.conf file. I would like to see this file getting created, so I close the debugger, delete it, and open `C:\WINDOWS\System32\mfm\msrll.exe` in Ollydbg. Now I will have to repeat the manual unpacking again, since this new file was copied from the original specimen executable.

After animating over the file until I see references to this file, I see it create an empty jtram.conf file at x409FD5 (see Figure 29). A little later, I see where the malware starts writing to the file, after some encryption of the data.

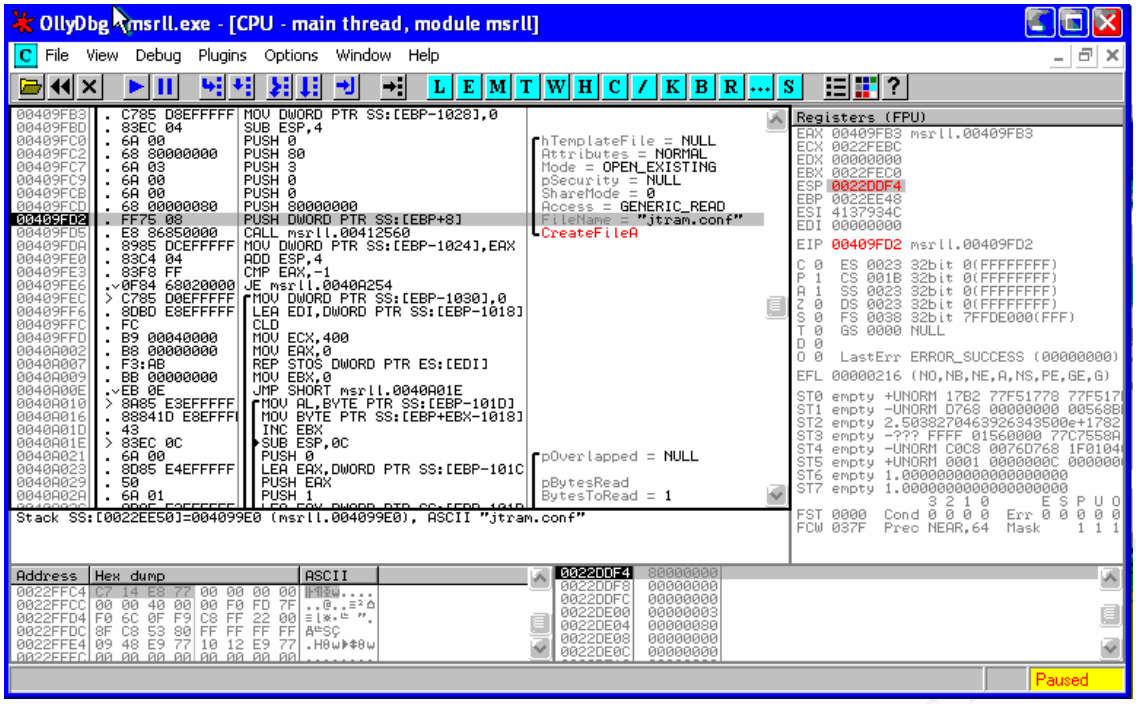

Figure 29

The string used for encryption is stored in the ESI register, which contains `DiCHFc2ioiVmb3cb4zZ7zWZH1oM=` as a key. This key is used repeatedly to encode each of the settings (see stack area in the bottom right of Figure 30).

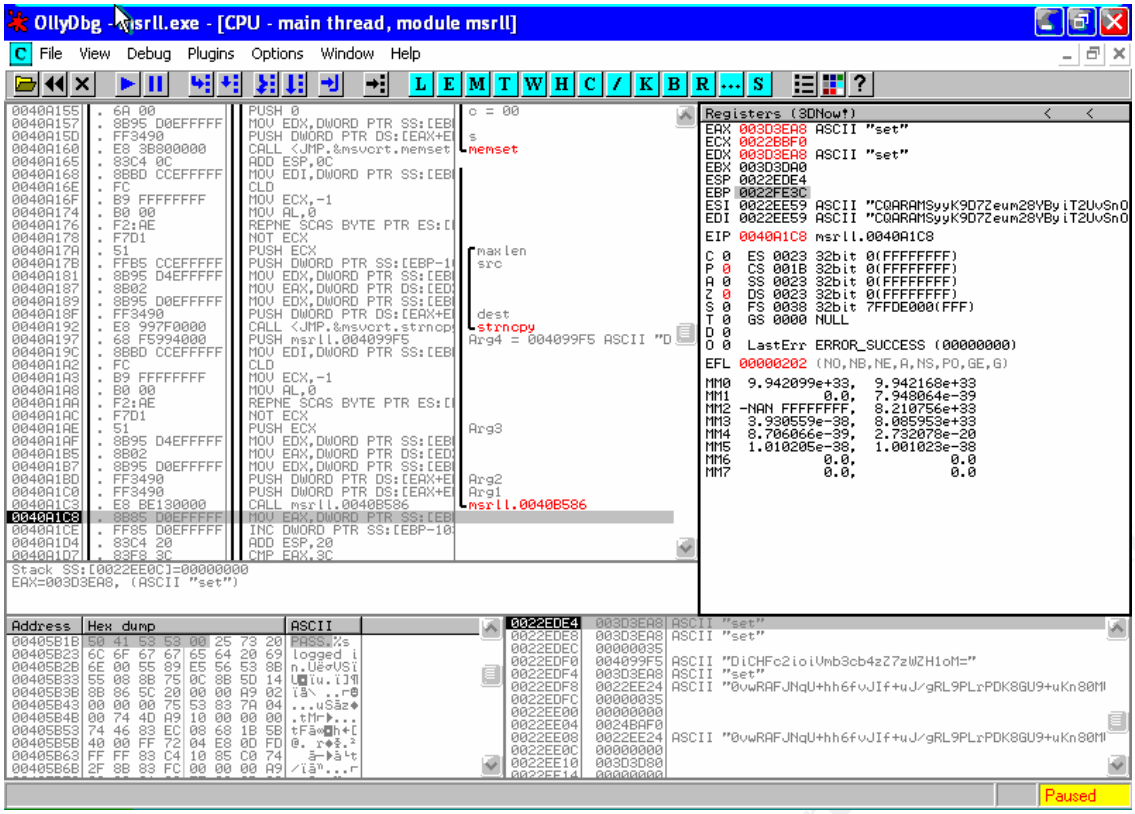

Figure 30

The two password strings look like md5 hashes, similar to what would be used on a Unix like system. I noticed the similarity of the first characters were similar to a shadow password file I examined recently, so I created a sample shadow password file with these passwords, ran a password cracking program known as "john." I run the cracker on a separate machine while I proceed with the analysis hoping that it will uncover a password.

The unencrypted jtram.conf is displayed in Figure 31. This information was gathered by watching the stack before each line was encrypted. Now that I have all the settings, I do not need to decrypt the file outside of the malware process by reproducing the decryption functionality. The dcc.pass reference is likely related to irc's dcc command, used for sending and receiving files.

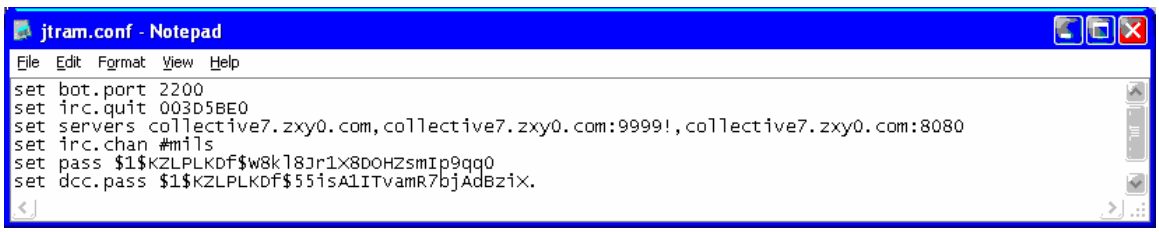

Figure 31

I look through the rest of the code and find a few references that hint at the malware's capabilities. I see a few printf-formatted strings with IRC syntax, as seen below in Figure 32 (also seen in Appendix B).

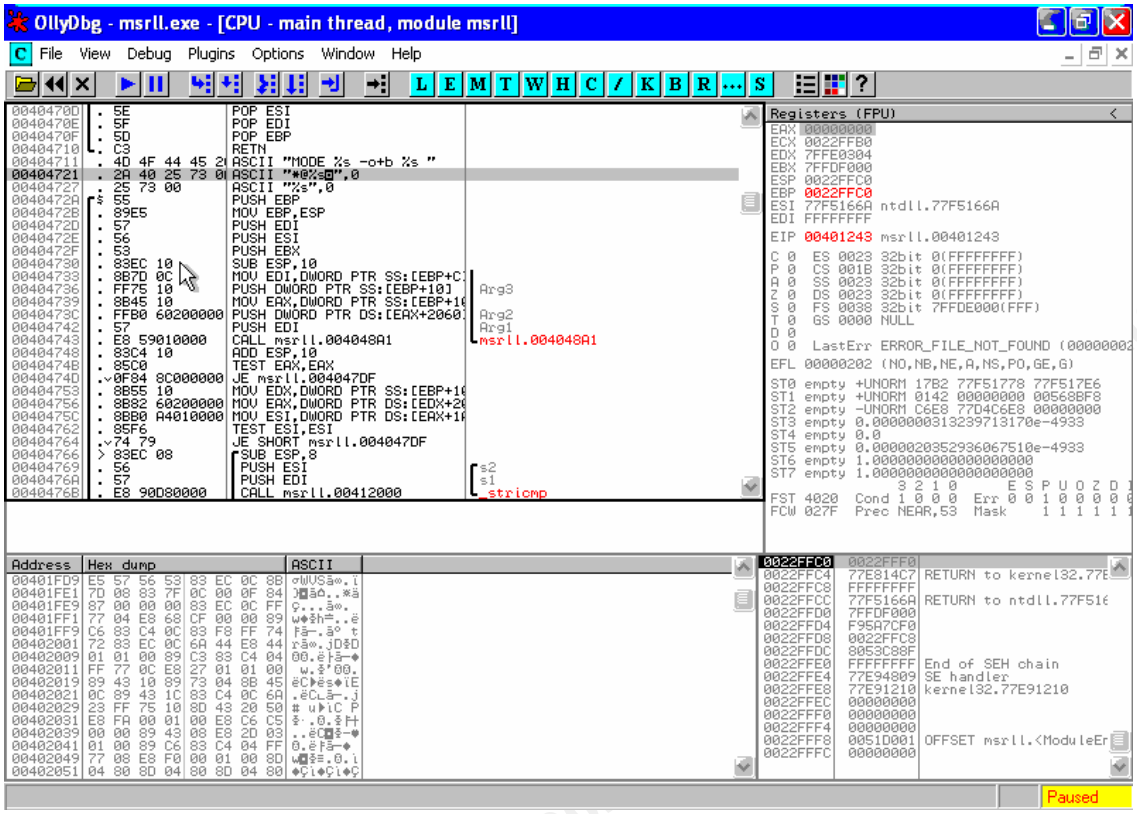

Figure 32

In Figure 33, I see the string "smurf done" which sounds like a reference to a smurf attack[13]. A smurf attack is a special form of Denial of Service attack. A Denial of Service attack is designed to exhaust resources of the target[14].

[13] Smurf attacks are a type of Denial of Service attack. For a full description on smurf attacks, see http://www.nordu.net/articles/smurf.html

[14] Skoudis, p. 121

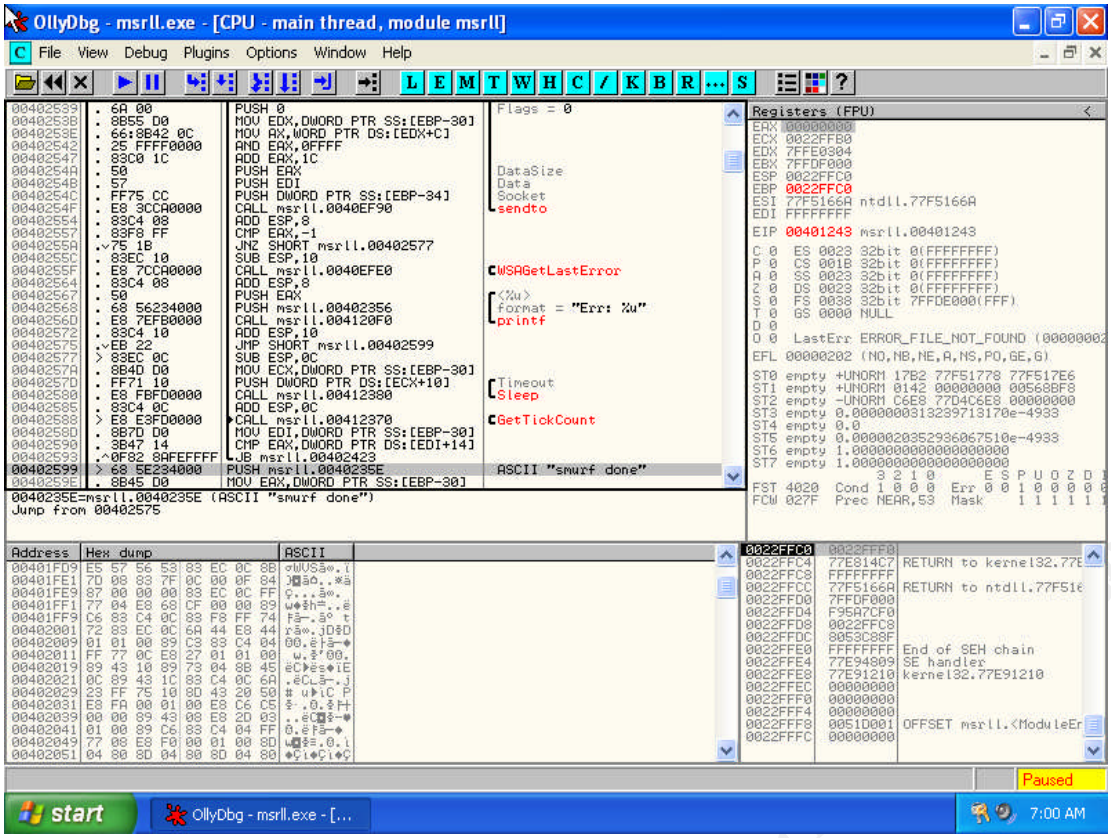

Figure 33

To isolate the precise moment where the malware reads any input from any port, I set a breakpoint at addresses at two places I see a call to the ws2_32.dll recv function, x40DD22 and x40E70D. Setting the hardware breakpoints on access in the dump window as I have done previously should trigger the breakpoints. Unfortunately, when sending data to either port 2200 or to the irc client, neither breakpoint is triggered. This could mean that there is another way to receive traffic or that there is an anti-debugging feature that disrupts the breakpoint.

Without running the process normally then breaking at the network input time, it is difficult to learn more about the authentication of the malware irc client or server on port 2200. Tracing and stepping to these points are unsuccessful: timeouts in network traffic from slowing the process down.

Taking a second look at the strings output of the unpacked malware, I see that parts of the code were taken from `LibTomCrypt 0.83` and `mIRC v6.12` (see Appendix B). There is also what appears to be a command listing, shown partly in Figure 34 below.

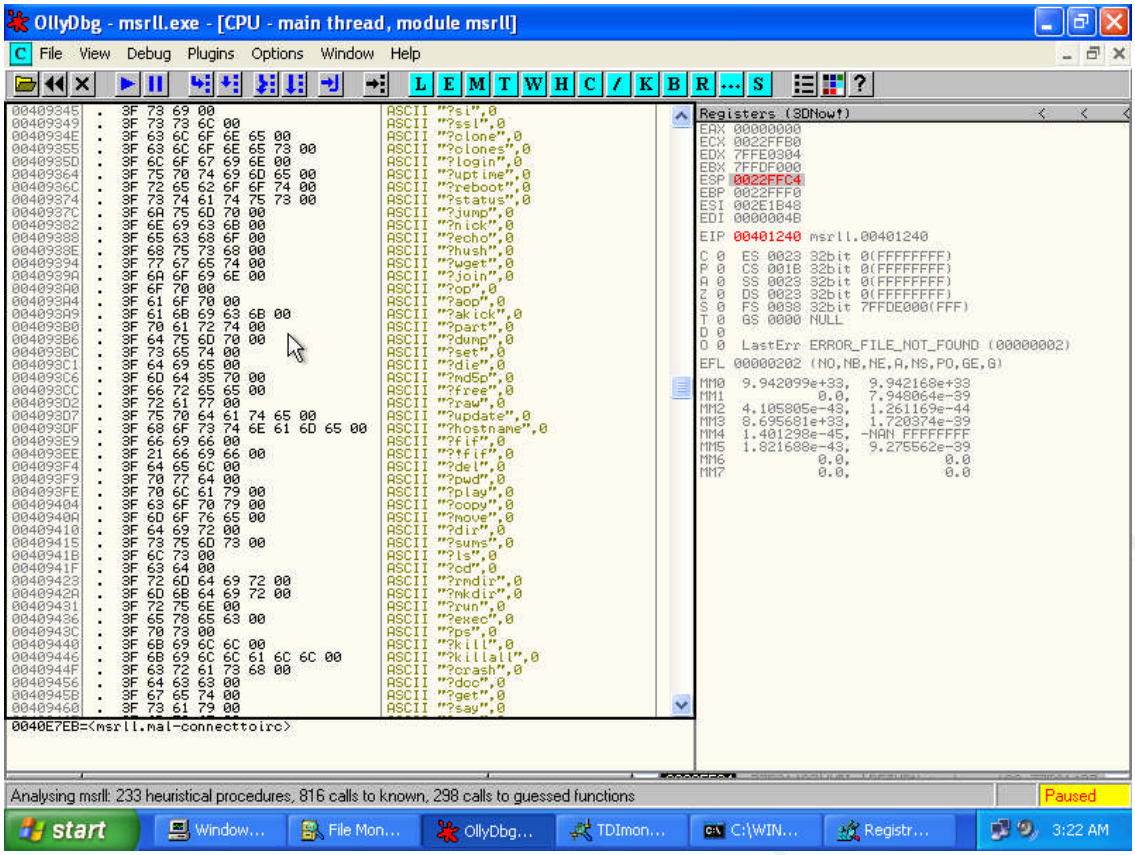

Figure 34

The code analysis was integral to learning about the capabilities of the malware. I would not have been able to see these features without being able to step into the code and unpack the malware. The unpacked strings results show that the malware is intended to send and receive files, run a program, listen as a service, and communicate via IRC.

Summary

This malware is an IRC server and client. The specimen has an IRC client that connects to `collective7.zxy0.com` and its own IRC server on port 2200. It has an IDENT server running on port 113 to respond to client authorization attempts typical in IRC usage. There is a reference of smurf and jolt processes, so it also appears to have a Denial of Service attack component.

The configuration file's password settings look to be standard md5 hashes, typical of a Unix like `/etc/passwd` file, but after 24 full days of using the john cracker program on a Pentium 4 2.4 GHz FreeBSD 4.10-STABLE server as the

only user process, no password match was found for the pass or dcc.pass values.

The dcc.pass setting implies that the client can be used for sending and retrieving files via the dcc functionality in IRC. This could be used to mine email address, financial information, or any other type of data stored in files on the victim.

Removal of the tool is easiest by booting into Safe Mode, keeping the process from running automatically. If the system is Windows 98 or earlier, remove the `HKLM/Software/Microsoft/Windows/CurrentVersion/Run/Rll enhanced drive` registry value. In other Windows systems, delete the entire `HKLM/System/CurrentControlSet/Services/mfm` key and reboot. To delete the actual files, remove the entire `C:\WINDOWS\System\mfm\` or `C:\WINDOWS\System32\mfm\` directory.

An egress proxy restricting outgoing IRC use would significantly limit the damage spread by this malware. Machines that do not protect their registry from unauthorized changes should be locked down to ensure they would not automatically run this malware. Many brands of antivirus products will also detect this and similar malware, even though they would not qualify as a virus.

A major weakness of the msrll.exe malware is it depends on name resolution to communicate. To thwart use of compromised victims, local DNS servers could redirect requests for the zxy0.com domain elsewhere.

The malware uses standard security and networking Windows libraries. The functionality appears to be built up so that the malware behaves practically the same on older versions of Windows as it does on the latest ones. The flexibility of the IRC file transfer and control would be valuable to any villain.

REFERENCES

“Aaron's Homepage.” 20 Nov. 2004. <http://www.exetools.com/unpackers.htm>

Čereň, Pavol. Crackproof Your Software. San Francisco: No Starch Press, 2002.

CrackZ. “Packers & Unpackers.” 06 May 2004. 20 Dec. 2004. <http://www.woodmann.com/crackz/Packers.htm#aspack_asprotect>

“FreeBSD Handbook.” 2004. The FreeBSD Project. 20 Nov. 2004. <http://www.freebsd.org/handbook/>

“For the PC, what are the differences between MFM, RLL, IDE, EIDE, ATA, ESDI, and SCSI hard drives?” 2005. The Trustees of Indiana University. 11 Nov. 2004. <http://kb.indiana.edu/data/adlt.html?cust=4074217.7239.131>

“Ollydbg stuph.” Win32ASM Community. 10 Jun. 2004. <http://ollydbg.win32asmcommunity.net/stuph/>

“Preventing Smurf Attacks.” Nordunet Information Service. 20 Nov. 2004. <http://www.nordu.net/articles/smurf.html>

“Products -- VMware Workstation 4.5.” 2004. VMware. 15 Nov. 2004 <http://www.vmware.com/products/desktop/ws_features.html>

Roesh, Martin and Chris Green. “SnortUsers Manual 2.2.0.” 2003. 20 Nov. 2004. <http://www.snort.org/docs/snort_manual/snort_manual.html>

Russinovich, Mark and Bryce Cogswell. “Sysinternals Freeware - Utilities for Windows NT and Windows 2000.” 15 Nov. 2004 <http://www.sysinternals.com/ntw2k/utilities.shtml>

Skoudis, Ed with Lenny Zeltser. Malware: Fighting Malicious Code. Upper Saddle River: Pretence Hall, 2004

Yuschuk, Oleh. **“**OllyDbg v1.10.” 08 Jun. 2004. 15 Nov. 2004. <http://home.t-online.de/home/Ollydbg/>

Appendix A – System Configurations

Guest FreeBSD Environment

```
Copyright (c) 1992-2002 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
        The Regents of the University of California. All rights reserved.
FreeBSD 4.7-RELEASE #0: Wed Oct  9 15:08:34 GMT 2002
    root@builder.freebsdmall.com:/usr/obj/usr/src/sys/GENERIC
Timecounter "i8254"  frequency 1193182 Hz
CPU: AMD Athlon(tm) XP 2100+ (1730.25-MHz 686-class CPU)
  Origin = "AuthenticAMD"  Id = 0x681  Stepping = 1

Features=0x383fbff<FPU,VME,DE,PSE,TSC,MSR,PAE,MCE,CX8,APIC,SEP,MTRR,PGE,MCA,CMOV,PAT,PSE3
6,MMX,FXSR,SSE>
  AMD Features=0xc0400000<AMIE,DSP,3DNow!>
real memory  = 100663296 (98304K bytes)
avail memory = 92639232 (90468K bytes)
Preloaded elf kernel "kernel" at 0xc050f000.
md0: Malloc disk
Using $PIR table, 9 entries at 0xc00fdf30
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Intel 82443BX (440 BX) host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1: <Intel 82443BX (440 BX) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
isab0: <Intel 82371AB PCI to ISA bridge> at device 7.0 on pci0
isa0: <ISA bus> on isab0
atapci0: <Intel PIIX4 ATA33 controller> port 0x1050-0x105f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
ata1: at 0x170 irq 15 on atapci0
uhci0: <Intel 82371AB/EB (PIIX4) USB controller> port 0x1060-0x107f irq 9 at device 7.2
on pci0
usb0: <Intel 82371AB/EB (PIIX4) USB controller> on uhci0
usb0: USB revision 1.0
uhub0: Intel UHCI root hub, class 9/0, rev 1.00/1.00, addr 1
uhub0: 2 ports with 2 removable, self powered
chip1: <Intel 82371AB Power management controller> port 0x1040-0x104f at device 7.3 on
pci0
pci0: <VGA-compatible display device> at 15.0
bt0: <Buslogic Multi-Master SCSI Host Adapter> port 0x1440-0x145f mem 0xf8000000-
0xf800001f irq 11 at device 16.0 on pci0
bt0: BT-958 FW Rev. 5.07B Ultra Wide SCSI Host Adapter, SCSI ID 7, 192 CCBs
lnc0: <PCNet/PCI Ethernet adapter> port 0x1080-0x10ff irq 10 at device 17.0 on pci0
lnc0: PCnet-PCI II address 00:0c:29:28:c2:5d
lnc0: driver is using old-style compatibility shims
pci0: <unknown card> (vendor=0x1274, dev=0x1371) at 18.0 irq 9
orm0: <Option ROMs> at iomem 0xc0000-0xc7fff,0xc8000-0xc8fff,0xdc000-0xdffff,0xe4000-
0xe7fff on isa0
fdc0: <Intel 82077 or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq 2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbdc0: <Keyboard controller (i8042)> at port 0x60,0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq 1 on atkbdc0
kbd0 at atkbd0
psm0: failed to get data.
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: model IntelliMouse, device ID 3
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
```

```
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: Generic chipset (NIBBLE-only) in COMPATIBLE mode
plip0: <PLIP network interface> on ppbus0
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
ppi0: <Parallel I/O> on ppbus0
ad0: 4095MB <VMware Virtual IDE Hard Drive> [8322/16/63] at ata0-master UDMA33
acd0: DVD-ROM <VMware Virtual IDE CDROM Drive> at ata1-master PIO4
Waiting 15 seconds for SCSI devices to settle
Mounting root from ufs:/dev/ad0s1a
```

Guest Windows XP Environment

```
Resource Summary Report   -  Page: 1

****************** SYSTEM SUMMARY ******************

Windows Version: Windows 5.1 Service Pack 1 (Build 2600)
Registered Owner: Jim Shewmaker
Registered Organization:
Computer Name: TESTXP
Machine Type: AT/AT COMPATIBLE
System BIOS Version: PTLTD  - 6040000
System BIOS Date: 04/21/04
Processor Type: x86 Family 6 Model 8 Stepping 1
Processor Vendor: AuthenticAMD
Number of Processors: 1
Physical Memory: 224 MB

****************** DISK DRIVE INFO ******************

  Drive A:
    Type: 3.5" 1.44MB floppy disk drive
    Total Space: 1,474,560 bytes
    Heads: 2
    Cylinders: 80
    Sectors Per Track: 18
    Bytes Per Sector: 512

  Drive C:
    Type: Fixed disk drive
    Total Space: 10,725,732,352 bytes
    Free Space: 9,234,546,688 bytes
    Heads: 255
    Cylinders: 1305
    Sectors Per Track: 63
    Bytes Per Sector: 512

  Drive D:
    Type: CD-ROM drive
    Total Space: 654,311,424 bytes

****************** IRQ SUMMARY ******************

IRQ Usage Summary:
  (ISA)  0    System timer
  (ISA)  1    Standard 101/102-Key or Microsoft Natural PS/2 Keyboard
  (ISA)  3    Communications Port (COM2)
  (ISA)  4    Communications Port (COM1)
  (ISA)  6    Standard floppy disk controller
  (ISA)  8    System CMOS/real time clock
  (ISA)  9    Microsoft ACPI-Compliant System
  *(PCI) 11    SCSI Controller
  (ISA) 12    PS/2 Compatible Mouse
  (ISA) 14    Primary IDE Channel
  (ISA) 15    Secondary IDE Channel
  (PCI) 18    AMD PCNET Family PCI Ethernet Adapter
  (PCI) 19    Intel(r) 82371AB/EB PCI to USB Universal Host Controller
```

```
  (PCI) 19    Creative AudioPCI (ES1371,ES1373) (WDM)

****************** DMA USAGE SUMMARY ******************

DMA Usage Summary:

Resource Summary Report   -  Page: 2

   2    Standard floppy disk controller
   4    Direct memory access controller

****************** MEMORY SUMMARY ******************

Memory Usage Summary:
  [000A0000 - 000BFFFF]  PCI bus
    [000A0000 - 000BFFFF]  VgaSave
  [000CC000 - 000CFFFF]  PCI bus
  [000D0000 - 000D3FFF]  PCI bus
  [000D4000 - 000D7FFF]  PCI bus
  [000D8000 - 000DBFFF]  PCI bus
  [000E0000 - 000E3FFF]  PCI bus
  [0E000000 - FFDFFFFF]  PCI bus
    *[F4000000 - F400001F]  SCSI Controller
    *[F5000000 - F5FFFFFF]  Video Controller (VGA Compatible)
    *[F6000000 - F6FFFFFF]  Video Controller (VGA Compatible)

****************** IO PORT SUMMARY ******************

I/O Ports Usage Summary:
  [00000000 - 00000CF7]  PCI bus
    [00000000 - 0000000F]  Direct memory access controller
    [00000010 - 0000001F]  Motherboard resources
    [00000020 - 00000021]  EISA programmable interrupt controller
    [00000024 - 00000025]  Motherboard resources
    [00000028 - 00000029]  Motherboard resources
    [0000002C - 0000002D]  Motherboard resources
    [00000030 - 00000031]  Motherboard resources
    [00000034 - 00000035]  Motherboard resources
    [00000038 - 00000039]  Motherboard resources
    [0000003C - 0000003D]  Motherboard resources
    [00000040 - 00000043]  System timer
    [00000050 - 00000053]  Motherboard resources
    [00000060 - 00000060]  Standard 101/102-Key or Microsoft Natural PS/2 Keyb
    [00000061 - 00000061]  System speaker
    [00000064 - 00000064]  Standard 101/102-Key or Microsoft Natural PS/2 Keyb
    [00000070 - 00000071]  System CMOS/real time clock
    [00000072 - 00000077]  Motherboard resources
    [00000080 - 00000080]  Motherboard resources
    [00000081 - 0000008F]  Direct memory access controller
    [00000090 - 0000009F]  Motherboard resources
    [000000A0 - 000000A1]  EISA programmable interrupt controller
    [000000A4 - 000000A5]  Motherboard resources
    [000000A8 - 000000A9]  Motherboard resources
    [000000AC - 000000AD]  Motherboard resources
    [000000B0 - 000000B5]  Motherboard resources
    [000000B8 - 000000B9]  Motherboard resources
    [000000BC - 000000BD]  Motherboard resources
    [000000C0 - 000000DF]  Direct memory access controller
    [00000170 - 00000177]  Secondary IDE Channel
    [000001CE - 000001CF]  VgaSave
    [000001F0 - 000001F7]  Primary IDE Channel
    [00000200 - 00000207]  Game Port for Creative
    [00000274 - 00000277]  ISAPNP Read Data Port
    [00000279 - 00000279]  ISAPNP Read Data Port
    [000002E8 - 000002EF]  VgaSave
    [000002F8 - 000002FF]  Communications Port (COM2)

Resource Summary Report   -  Page: 3

    [00000376 - 00000376]  Secondary IDE Channel
    [00000378 - 0000037F]  Printer Port (LPT1)
```

```
    [000003B0 - 000003BB]  VgaSave
    [000003C0 - 000003DF]  VgaSave
    [000003F0 - 000003F5]  Standard floppy disk controller
    [000003F6 - 000003F6]  Primary IDE Channel
    [000003F7 - 000003F7]  Standard floppy disk controller
    [000003F8 - 000003FF]  Communications Port (COM1)
    [000004D0 - 000004D1]  EISA programmable interrupt controller
    [00000A79 - 00000A79]  ISAPNP Read Data Port
  [00000D00 - 0000FFFF]  PCI bus
    [00001000 - 0000103F]  Motherboard resources
    [00001040 - 0000104F]  Motherboard resources
    [00001050 - 0000105F]  Intel(r) 82371AB/EB PCI Bus Master IDE Controller
    [00001060 - 0000107F]  Intel(r) 82371AB/EB PCI to USB Universal Host Contr
    [00001080 - 000010FF]  AMD PCNET Family PCI Ethernet Adapter
    [00001400 - 0000143F]  Creative AudioPCI (ES1371,ES1373) (WDM)
    *[00001440 - 0000145F]  SCSI Controller
    *[00001460 - 0000146F]  Video Controller (VGA Compatible)
```

Guest Windows 98SE Environment

```
Resource Summary Report   -  Page: 1

******************** SYSTEM SUMMARY ********************

Windows version: 4.10.2222
Computer Name: Unknown
System BUS Type: ISA
BIOS Name: Unknown
BIOS Date: 04/21/04
BIOS Version: EPP revision 9.00
Machine Type: IBM PC/AT
Processor Vendor: AuthenticAMD
Processor Type: AMD Athlon(tm) XP 2100+
Math Co-processor: Present
Registered Owner: James Shewmaker
Registered Company: bluenotch

******************** IRQ SUMMARY ********************

IRQ Usage Summary:
  00 - System timer
  01 - Standard 101/102-Key or Microsoft Natural Keyboard
  02 - EISA programmable interrupt controller
  03 - Communications Port (COM2)
  04 - Communications Port (COM1)
  05 - Printer Port (LPT1)
  06 - Standard Floppy Disk Controller
  07 - AMD PCNET Family Ethernet Adapter (PCI-ISA)
  07 - ACPI IRQ Holder for PCI IRQ Steering
  08 - System CMOS/real time clock
  09 - ACPI IRQ Holder for PCI IRQ Steering
  09 - SCI IRQ used by ACPI bus
  09 - Intel 82371AB/EB PCI to USB Universal Host Controller
  11 - ACPI IRQ Holder for PCI IRQ Steering
  11 - BusLogic MultiMaster PCI SCSI Host Adapters
  12 - VMware Pointing Device
  13 - Numeric data processor
  14 - Primary IDE controller (dual fifo)
  14 - Intel 82371AB/EB PCI Bus Master IDE Controller
  15 - Secondary IDE controller (dual fifo)
  15 - Intel 82371AB/EB PCI Bus Master IDE Controller

******************** IO PORT SUMMARY ********************

I/O Port Usage Summary:
  0000h-000Fh - Direct memory access controller
  0010h-001Fh - Motherboard resources
  0020h-0021h - EISA programmable interrupt controller
```

```
  0024h-0025h - Motherboard resources
  0028h-0029h - Motherboard resources
  002Ch-002Dh - Motherboard resources
  0030h-0031h - Motherboard resources
  0034h-0035h - Motherboard resources
  0038h-0039h - Motherboard resources
  003Ch-003Dh - Motherboard resources
  0040h-0043h - System timer
  0050h-0053h - Motherboard resources
  0060h-0060h - Standard 101/102-Key or Microsoft Natural Keyboard
  0061h-0061h - System speaker


System Resource Report    -  Page: 2

  0064h-0064h - Standard 101/102-Key or Microsoft Natural Keyboard
  0070h-0071h - System CMOS/real time clock
  0072h-0077h - Motherboard resources
  0080h-0080h - Motherboard resources
  0081h-008Fh - Direct memory access controller
  0090h-009Fh - Motherboard resources
  00A0h-00A1h - EISA programmable interrupt controller
  00A4h-00A5h - Motherboard resources
  00A8h-00A9h - Motherboard resources
  00ACh-00ADh - Motherboard resources
  00B0h-00B5h - Motherboard resources
  00B8h-00B9h - Motherboard resources
  00BCh-00BDh - Motherboard resources
  00C0h-00DFh - Direct memory access controller
  00F0h-00FFh - Numeric data processor
  0170h-0177h - Intel 82371AB/EB PCI Bus Master IDE Controller
  0170h-0177h - Secondary IDE controller (dual fifo)
  01F0h-01F7h - Primary IDE controller (dual fifo)
  01F0h-01F7h - Intel 82371AB/EB PCI Bus Master IDE Controller
  02F8h-02FFh - Communications Port (COM2)
  0376h-0376h - Secondary IDE controller (dual fifo)
  0376h-0376h - Intel 82371AB/EB PCI Bus Master IDE Controller
  0378h-037Fh - Printer Port (LPT1)
  03B0h-03BBh - VMware SVGA II
  03C0h-03DFh - VMware SVGA II
  03F0h-03F5h - Standard Floppy Disk Controller
  03F6h-03F6h - Intel 82371AB/EB PCI Bus Master IDE Controller
  03F6h-03F6h - Primary IDE controller (dual fifo)
  03F7h-03F7h - Standard Floppy Disk Controller
  03F8h-03FFh - Communications Port (COM1)
  04D0h-04D1h - EISA programmable interrupt controller
  0CF8h-0CFFh - PCI bus
  1000h-103Fh - Motherboard resources
  1040h-104Fh - Motherboard resources
  1050h-1057h - Primary IDE controller (dual fifo)
  1050h-105Fh - Intel 82371AB/EB PCI Bus Master IDE Controller
  1058h-105Fh - Secondary IDE controller (dual fifo)
  1060h-107Fh - Intel 82371AB/EB PCI to USB Universal Host Controller
  1080h-108Fh - VMware SVGA II
  10A0h-10BFh - BusLogic MultiMaster PCI SCSI Host Adapters
  1400h-147Fh - AMD PCNET Family Ethernet Adapter (PCI-ISA)

******************** UPPER MEMORY USAGE SUMMARY ********************

Memory Usage Summary:
  000A0000h-000AFFFFh - VMware SVGA II
  000B0000h-000BFFFFh - VMware SVGA II
  000C0000h-000C7FFFh - VMware SVGA II
  000C8000h-000C8FFFh - AMD PCNET Family Ethernet Adapter (PCI-ISA)
  03000000h-03007FFFh - VMware SVGA II
  F8000000h-FBFFFFFFh - Intel 82443BX Pentium(r) II Processor to PCI bridge (
  FC000000h-FCFFFFFFh - VMware SVGA II
  FD000000h-FDFFFFFFh - VMware SVGA II
  FE000000h-FE00001Fh - BusLogic MultiMaster PCI SCSI Host Adapters
```

Appendix B – Strings in Unpacked Malware

```
[^_]
?insmod
?rmmod
?lsmod
=t0A
%s: <mod name>
%s: mod list full
%s: err: %u
mod_init
mod_free
%s: cannot init %s
%s: %s loaded (%u)
%s: mod allready loaded
%s:%s err %u
[^_]
%s:%s not found
%s: unloading %s
[^_]
[%u]: %s hinst:%x
[^_]
unloading %s
%s: invalid_addr: %s
%s%s [port]
[^_]
finished %s
F Pj
[^_]
%s <ip> <port> <t_time> <delay>
[^_]
sockopt: %u
sendto err: %u
sockraw: %u
syn: done
F Pj
F Pj
F Pj
[^_]
%s <ip> <duration> <delay>
[^_]
sendto: %u
jolt2: done
F Pj
[^_]
%s <ip> <p size> <duration> <delay>
hi#@
7h`"@
[^_]
Err: %u
smurf done
Pj h
PhV#@
h^#@
[^_]
&err: %u
G Pj
[^_]
?ping
?udp
?syn
?smurf
?jolt
PONG :%s
0h (@
[^_]
[^_]
%s!%s@%s
%PSh
[^_]
%s!%s
; u
SVh=+@
[^_]
irc.nick
NICK %s
MODE
hV,@
=P1A
[^_]
NETWORK=
[^_]
[^_]
irc.pre
[^_]
%s__
__%s
%s``
_%s`
_%s__
__%s__
__%s___
NICK %s
NICK
%s %s
[^_]
[^_]
;&uJ
j'SV
[^_]
[^_]
irc.chan
WSj
[^_]
%s %s
WHO %s
PPhV,@
hu7@
[^_]
USERHOST %s
logged into %s(%s) as %s
<$hE:@
PhR:@
[^_]
j%VW
[^_]
nick.pre
%s-%04u
irc.user
irc.usereal
irc.real
irc.pass
tsend(): connection to %s:%u failed
USER %s localhost 0 :%s
NICK %s
Ph <@
[^_]
8:u:
: t;
[^_]
PING
PRIVMSG
JOIN
QUIT
PART
KICK
```

```
trecv(): Disconnected from %s err:%u
=P2A
[^_]
[^_]
NOTICE
%s %s :%s
hrD@
Ph}D@
[^_]
MODE %s -o+b %s *@%s
C'PSWh
Sh'G@
[^_]
MODE %s -bo %s %s
CmPW
Sh'G@
[^_]
[^_]
%s.key
h$I@
Ph'G@
[^_]
sk#%u %s is dead!
s_check: %s dead? pinging...
PING :ok
s_check: send error to %s disconnecting
expect the worst
s_check: killing socket %s
irc.knick
jtr.%u%s.iso
ison %s
servers
s_check: trying %s
h(K@
Ph9K@
hTK@
h^K@
PhkK@
ShtK@
8:u.
uYVh|K@
hA<@
[^_]
%s.mode
MODE %s %s
ShRP@
Sh$I@
PShZP@
[^_]
[^_]
hP3A
hX3A
h^3A
hf3A
h<R@
hk3A
[^_]
[^_]
mode %s +o %s
akick
mode %s +b %s %s
KICK %s %s
[^_]
irc.pre
Set an irc sock to preform %s command on
  Type
%csklist
 to view current sockets, then
%cdccsk
 <#>
Ph`W@
%s: dll loaded
%s: %d
[^_]
hA<@
RhHY@
RhHY@
said %s to %s
usage: %s <target> "text"
hHY@
[^_]
%s not on %s
usage: %s <nick> <chan>
htZ@
[^_]
PASS
%s logged in
Sh [@
sys: %s bot: %s
preformance counter not avail
usage: %s <cmd>
%s free'd
unable to free %s
0h+\@
RSVj
later!
unable to %s errno:%u
service:%c user:%s inet connection:%c
contype:%s reboot privs:%c
Ph@]@
kill
 %-5u %s
h#^@
[^_]
%s: %s
%s: somefile
PhHY@
[^_]
host: %s ip: %s
C Ph
capGetDriverDescriptionA
XP++
cpus:%u
 CAM
WIN%s (u:%s)%s%s mem:(%u/%u) %u%% %s %s
hp4A
hib@
=l4A
=l4A
hh4A
=l4A
5d4A
[^_]
open
%s: %s (%u)
[^_]
[^_]
NICK
%s %s
%s bad args
3hTg@
[^_]
akick
KICK
%s[%u] %s
%s removed
couldnt find %s
%s added
%s allready in list
usage: %s +/- <host>
8-u0
8+uN
```

```
7h*h@
h?h@
[^_]
jtram.conf
%s /t %s
jtr.home
%s\%s
%s: possibly failed: code %u
%s: possibly failed
%s: exec of %s failed err: %u
u.exf
Ph+j@
Ph?j@
[^_]
jtr.id
%s: <url> <id>
h]j@
[^_]
[^_]
hyn@
IRC
DCC
DATH
IATH
IREG
CLON
ICON
RNL
RBN
WSN
WCON
LSN
SSL
S>S
  #%u [fd:%u] %s:%u [%s%s] last:%u
   |\=> [n:%s fh:%s] (%s)
   |
   |---[%s] (%u) %s
   |     |-[%s%s] [%s]
   |=> (%s) (%.8x)
h@o@
B$PRhco@
=p5A
h}o@
F'PV
[^_]
%s <pass> <salt>
3h`s@
[^_]
%s <nick> <chan>
!%s!
h5t@
PING %s
mIRC v6.12 Khaled Mardam-Bey
VERSION %s
dcc.pass
temp add %s
$h%u@
[^_]
[^_]
%s%u-%s
%s opened (%u)
%u bytes from %s in %u seconds saved to
%s
(%s %s): incomplete! %u bytes
couldnt open %s err:%u
(%s) %s: %s
(%s) urlopen failed
(%s): inetopen failed
Whjv@
hrv@
hGY@
Ph w@
[^_]
no file name in %s
h6w@
[^_]
[^_]
%s created
[^_]
%s %s to %s Ok
3hI~@
[^_]
%0.2u/%0.2u/%0.2u  %0.2u:%0.2u  %15s  %s
%s (err: %u)
[^_]
ShHY@
err: %u
%s %s :ok
[^_]
unable to %s %s (err: %u)
ShHY@
[^_]
%-16s %s
%-16s (%u.%u.%u.%u)
[^_]
hHY@
[%s][%s] %s
[^_]
closing %u [%s:%u]
unable to close socket %u
[^_]
[^_]
using sock #%u %s:%u (%s)
Invalid sock
usage %s <socks #>
leaves %s
:0 * * :%s
hHY@
[^_]
joins: %s
chat
ACCEPT
resume
err: %u
DCC ACCEPT %s %s %s
dcc_resume: cant find port %s
send
dcc.dir
%s\%s\%s\%s
unable to open (%s): %u
resuming dcc from %s to %s
DCC RESUME %s %s %u
[^_]
h iA
h iA
[^_]
?ssl
?clone
?clones
?login
?uptime
?reboot
?status
?jump
?nick
?echo
?hush
?wget
?join
?aop
?akick
```

```
?part
?dump
?set
?die
?md5p
?free
?raw
?update
?hostname
?fif
?!fif
?del
?pwd
?play
?copy
?move
?dir
?sums
?rmdir
?mkdir
?run
?exec
?kill
?killall
?crash
?dcc
?get
?say
?msg
?sklist
?unset
?uattr
?dccsk
?con
?killsk
VERSION*
PING
IDENT
[^_]
%ud %02uh %02um %02us
%02uh %02um %02us
%um %02us
[^_]
[^_]
[^_]
[^_]
jtram.conf
jtr.*
DiCHFc2ioiVmb3cb4zZ7zWZH1oM=
conf_dump: wrote %u lines
[^_]
[^_]
> u
[^_]
get of %s incomplete at %u bytes
get of %s completed (%u bytes), %u
seconds %u cps
error while writing to %s (%u)
[^_]
chdir: %s -> %s (%u)
,Ph\
dcc_wait: get of %s from %s timed out
dcc_wait: closing [#%u] %s:%u (%s)
PRhP
[^_]
SEND
%4s #%.2u %s %ucps %u%% [sk#%u] %s
%u Send(s) %u Get(s) (%u transfer(s)
total) UP:%ucps DOWN:%ucps Total:%ucps
PRQh0
[^_]
send of %s incomplete at %u bytes
send of %s completed (%u bytes), %u
seconds %u cps
cant open %s (err:%u) pwd:{%s}
DCC SEND %s %u %u %u
$Sho
[^_]
[^_]
[^_]
%s %s
%s exited with code %u
%s\%s
%s: %s
exec: Error:%u pwd:%s cmd:%s
[^_]
dcc.pass
bot.port
%s bad pass from "%s"@%s
%s: connect from %s
h0;A
jtr.bin
msrll.exe
jtr.home
2200
jtr.id
run5
irc.quit
servers
collective7.zxy0.com,collective7.zxy0.com
:9999!,collective7.zxy0.com:8080
irc.chan
#mils
pass
$1$KZLPLKDf$W8kl8Jr1X8DOHZsmIp9qq0
$1$KZLPLKDf$55isA1ITvamR7bjAdBziX.
m220
=P;A
SSL_get_error
SSL_load_error_strings
SSL_library_init
SSLv3_client_method
SSL_set_connect_state
SSL_CTX_new
SSL_new
SSL_set_fd
SSL_connect
SSL_write
SSL_read
SSL_shutdown
SSL_free
SSL_CTX_free
kernel32.dll
QueryPerformanceCounter
QueryPerformanceFrequency
RegisterServiceProcess
jtram.conf
[^_]
irc.user
%s : USERID : UNIX : %s
QUIT :FUCK %u
Killed!? Arrg! [%u]
QUIT :%s
SeShutdownPrivilege
%s\%s
%s\%s\%s
Rll enhanced drive
software\microsoft\windows\currentversion
\run
/d "%s"
open
WSVh
```

```
[^_]
[^_]
>*uj
>*uY
>*t!
 < u&
[^_]
-N;u
[^_]
./0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabc
defghijklmnopqrstuvwxyz
IQhx
8$t+
IQRS
IQRS
5pSA
IQRS
5pSA
IQRS
5pSA
5pSA
[^_]
IQhx
[^_]
[^_]
Ph~f
[^_]
Ph~f
[^_]
=,;A
=(;A
[^_]
[^_]
= ;A
[^_]
[^_]
[^_]
[^_]
usage %s: server[:port] amount
[^_]
%s: %s
%s %s %s <PARAM>
JOIN
PART
%s: [NETWORK|all] %s <"parm"> ...
[^_]
USER %s localhost 0 :%s
NICK %s
PSVh
[^_]
md5.c
md != NULL
QZ^&
[^_]
j*h@
buf != NULL
[^_]
hash != NULL
[^_]
message digest
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmno
pqrstuvwxyz0123456789
123456789012345678901234567890123456789012
345678901234567890123456789012345678901234
iw&a
ki}|
RZ/1
IQRS
[^_]
sprng
sprng.c
buf != NULL
rc6.c
skey != NULL
key != NULL
[^_]
ct != NULL
pt != NULL
[^_]
+0+x
[^_]
#4EVgx
$5FWhy
#4EVgx
$5FWhy
#4EVgx
$5FWhy
gN]HU
[^_]
desired_keysize != NULL
ctr.c
ctr != NULL
key != NULL
count != NULL
[^_]
ct != NULL
pt != NULL
[^_]
j)h0
j(h0
j'h0
WVSS
[^_]
jMh0
jLh0
jKh0
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmno
pqrstuvwxyz0123456789+/
?456789:;<=
 !"#$%&'()*+,-./0123
base64.c
outlen != NULL
out != NULL
in != NULL
WVSP
[^_]
@A;E
j)hP
j(hP
j'hP
[^_]
jVhP
jUhP
jThP
_ARGCHK '%s' failure on line %d of file
%s
crypt.c
name != NULL
[^_]
[^_]
[^_]
[^_]
cipher != NULL
WVSV
[^_]
hash != NULL
WVSW
[^_]
WVSP
[^_]
prng != NULL
```

```
WVSP
[^_]
WVSP
[^_]
LibTomCrypt 0.83
Endianess: little (32-bit words)
Clean stack: disabled
Ciphers built-in:
   Blowfish
   RC2
   RC5
   RC6
   Serpent
   Safer+
   Safer
   Rijndael
   XTEA
   Twofish
   CAST5
   Noekeon
Hashes built-in:
   SHA-512
   SHA-384
   SHA-256
   TIGER
   SHA1
   MD5
   MD4
   MD2
Block Chaining Modes:
   CFB
   OFB
   CTR
PRNG:
   Yarrow
   SPRNG
   RC4
PK Algs:
   RSA
   DH
   ECC
   KR
Compiler:
   WIN32 platform detected.
   GCC compiler detected.
Various others:  BASE64  MPI  HMAC
/dev/random
[^_]
Microsoft Base Cryptographic Provider
v1.0
bits.c
buf != NULL
t9VWS
[^_]
prng != NULL
[^_]
<"tx< tf<     t
< tV<   t
< tJ<   tF
-LIBGCCW32-EH-SJLJ-GTHR-MINGW32
Q4_]
AAAA
AAAA
AAAA
AAAA
AAAA
AAAA
AAAA
AAAA
WVSQ
[^_]
,[^_]
uE;}
 <ip> <total secs> <p size> <delay>
 modem
 Lan
 Proxy
 ??
 none
m220 1.0 #2730 Mar 16 11:47:38 2004
unable to %s %s (err: %u)
unable to kill %s (%u)
%s killed (pid:%u)
AVICAP32.dll
unable to kill %u (%u)
pid %u killed
error!
ran ok
MODE %s +o %s
set %s %s
Mozilla/4.0
Accept: */*
<DIR>
Could not copy %s to %s
%s copied to %s
0123456789abcdef
%s unset
unable to unset %s
(%s) %s
%s %s
libssl32.dll
libeay32.dll
<die|join|part|raw|msg>
AdjustTokenPrivileges
CloseServiceHandle
CreateServiceA
CryptAcquireContextA
CryptGenRandom
CryptReleaseContext
GetUserNameA
LookupPrivilegeValueA
OpenProcessToken
OpenSCManagerA
RegCloseKey
RegCreateKeyExA
RegSetValueExA
RegisterServiceCtrlHandlerA
SetServiceStatus
StartServiceCtrlDispatcherA
AddAtomA
CloseHandle
CopyFileA
CreateDirectoryA
CreateFileA
CreateMutexA
CreatePipe
CreateProcessA
CreateToolhelp32Snapshot
DeleteFileA
DuplicateHandle
EnterCriticalSection
ExitProcess
ExitThread
FileTimeToSystemTime
FindAtomA
FindClose
FindFirstFileA
FindNextFileA
FreeLibrary
GetAtomNameA
GetCommandLineA
GetCurrentDirectoryA
```

```
GetCurrentProcess
GetCurrentThreadId
GetExitCodeProcess
GetFileSize
GetFullPathNameA
GetLastError
GetModuleFileNameA
GetModuleHandleA
GetProcAddress
GetStartupInfoA
GetSystemDirectoryA
GetSystemInfo
GetTempPathA
GetTickCount
GetVersionExA
GlobalMemoryStatus
InitializeCriticalSection
IsBadReadPtr
LeaveCriticalSection
LoadLibraryA
MoveFileA
OpenProcess
PeekNamedPipe
Process32First
Process32Next
QueryPerformanceFrequency
ReadFile
ReleaseMutex
RemoveDirectoryA
SetConsoleCtrlHandler
SetCurrentDirectoryA
SetFilePointer
SetUnhandledExceptionFilter
Sleep
TerminateProcess
WaitForSingleObject
WriteFile
_itoa
_stat
_strdup
_stricmp
__getmainargs
__p__environ
__p__fmode
__set_app_type
_beginthread
_cexit
_errno
_fileno
_iob
_onexit
_setmode
_vsnprintf
abort
atexit
atoi
clock
fclose
fflush
fgets
fopen
fprintf
fread
free
fwrite
malloc
memcpy
memset
printf
raise
rand
realloc
setvbuf
signal
sprintf
srand
strcat
strchr
strcmp
strcpy
strerror
strncat
strncmp
strncpy
strstr
time
toupper
ShellExecuteA
DispatchMessageA
ExitWindowsEx
GetMessageA
PeekMessageA
GetFileVersionInfoA
VerQueryValueA
InternetCloseHandle
InternetGetConnectedState
InternetOpenA
InternetOpenUrlA
InternetReadFile
WSAGetLastError
WSASocketA
WSAStartup
__WSAFDIsSet
accept
bind
closesocket
connect
gethostbyaddr
gethostbyname
gethostname
getsockname
htonl
htons
inet_addr
inet_ntoa
ioctlsocket
listen
ntohl
recv
select
send
sendto
setsockopt
shutdown
socket
ADVAPI32.DLL
KERNEL32.dll
msvcrt.dll
msvcrt.dll
SHELL32.DLL
USER32.dll
VERSION.dll
WININET.DLL
WS2_32.DLL
]^SP
]kSW
VirtualAlloc
VirtualFree
PQVS
t.x,
[^YX
kernel32.dll
```

```
ExitProcess
user32.dll
MessageBoxA
wsprintfA
LOADER ERROR
The procedure entry point %s could not be
located in the dynamic link library %s
The ordinal %u could not be located in
the dynamic link library %s
 (08@P`p
|$,3
T$ v
(C@;
t$h3
D4l|M
_^]2
;;F,s
,;F0s
 ;F4s
;F8s
0>@D
_^][
T4$F
`u(j
L4#H
L4$F
_^][
_^][
_^][
D$$W3
0"@D
5>@D
D$ %
;|$(
8_^]
_^]2
kernel32.dll
GetProcAddress
GetModuleHandleA
LoadLibraryA
advapi32.dll
msvcrt.dll
msvcrt.dll
shell32.dll
user32.dll
version.dll
wininet.dll
ws2_32.dll
AdjustTokenPrivileges
_itoa
__getmainargs
ShellExecuteA
DispatchMessageA
GetFileVersionInfoA
InternetCloseHandle
WSAGetLastError
advapi32.dll
AdjustTokenPrivileges
CloseServiceHandle
CreateServiceA
CryptAcquireContextA
CryptGenRandom
CryptReleaseContext
GetUserNameA
LookupPrivilegeValueA
OpenProcessToken
OpenSCManagerA
RegCloseKey
RegCreateKeyExA
RegSetValueExA
RegisterServiceCtrlHandlerA
SetServiceStatus
StartServiceCtrlDispatcherA
kernel32.dll
AddAtomA
CloseHandle
CopyFileA
CreateDirectoryA
CreateFileA
CreateMutexA
CreatePipe
CreateProcessA
CreateToolhelp32Snapshot
DeleteFileA
DuplicateHandle
EnterCriticalSection
ExitProcess
ExitThread
FileTimeToSystemTime
FindAtomA
FindClose
FindFirstFileA
FindNextFileA
FreeLibrary
GetAtomNameA
GetCommandLineA
GetCurrentDirectoryA
GetCurrentProcess
GetCurrentThreadId
GetExitCodeProcess
GetFileSize
GetFullPathNameA
GetLastError
GetModuleFileNameA
GetModuleHandleA
GetProcAddress
GetStartupInfoA
GetSystemDirectoryA
GetSystemInfo
GetTempPathA
GetTickCount
GetVersionExA
GlobalMemoryStatus
InitializeCriticalSection
IsBadReadPtr
LeaveCriticalSection
LoadLibraryA
MoveFileA
OpenProcess
PeekNamedPipe
Process32First
Process32Next
QueryPerformanceFrequency
ReadFile
ReleaseMutex
RemoveDirectoryA
SetConsoleCtrlHandler
SetCurrentDirectoryA
SetFilePointer
SetUnhandledExceptionFilter
Sleep
TerminateProcess
WaitForSingleObject
WriteFile
msvcrt.dll
_itoa
_stat
_mbsdup
_strcmpi
msvcrt.dll
__getmainargs
__p__environ
```

```
__p__fmode
__set_app_type
_beginthread
_cexit
_errno
_fileno
_iob
_onexit
_setmode
_vsnprintf
abort
atexit
atoi
clock
fclose
fflush
fgets
fopen
fprintf
fread
free
fwrite
malloc
memcpy
memset
printf
raise
rand
realloc
setvbuf
signal
sprintf
srand
_mbscat
strchr
strcmp
_mbscpy
strerror
strncat
strncmp
strncpy
strstr
time
toupper
```

```
shell32.dll
ShellExecuteA
USER32.dll
DispatchMessageA
ExitWindowsEx
GetMessageA
PeekMessageA
version.dll
GetFileVersionInfoA
VerQueryValueA
wininet.dll
InternetCloseHandle
InternetGetConnectedState
InternetOpenA
InternetOpenUrlA
InternetReadFile
ws2_32.dll
WSAGetLastError
WSASocketA
WSAStartup
__WSAFDIsSet
accept
bind
closesocket
connect
gethostbyaddr
gethostbyname
gethostname
getsockname
htonl
htons
inet_addr
inet_ntoa
ioctlsocket
listen
htonl
recv
select
send
sendto
setsockopt
shutdown
socket
```