Fight crime.
Unravel incidents... one byte at a time.

Interested in learning more?
Check out the list of upcoming events offering
"Advanced Incident Response, Threat Hunting, and Digital Forensics (FOR508)
GIAC Reverse Engineering Malware (GREM)
Practical Assignment
Version 1.0

Malware: msrl.exe

ILOT XII
James M. Balcik

12/8/2004
# Table of Contents

## Abstract

## Laboratory Setup

### Hardware

### Networking

### Software

**Windows XP SP1 Software Tools:**
- Ghost
- `md5sum`
- BinText
- Regshot
- FileMon
- RegMon
- TDIMon

**Process Explorer**
- AutoRuns
- UltraEdit-32
- `nc` (Netcat)
- PESniffer
- PEInfo
- ASPACKDIE
- IDA Pro
- OllyDbg

**Red Hat 9 Software:**
- SNORT
- `ircd`
- `irc`

## Properties of the Malware Specimen

### Operating Systems Effected:

### Strings Embedded in File:
- Fig. A - Interesting strings packed msril.exe
- Fig. B - Interesting strings unpacked msril.exe

## Behavioral Analysis

### Preparation for Infection
- Fig. 1 - Regshot

### Infection
- Fig. 2 - Regshot Compare Results
- Fig. 3 - FileMon Results
- Fig. 4 - RegMon Results
- Fig. 5 - TDIMon Results
- Fig. 6 - Snort Capture 2
- Fig. 7 - Hosts File Modification
- Fig. 8 - Snort Capture 3
- Fig. 9 - Snort Capture 4

## Code Analysis

### The Search for Authentication Code
- Fig. 10 - IDA Pro Text Search Results
Abstract

The intent of this paper is to partially fulfill the requirements of the GREM certificate and to demonstrate my knowledge of “Reverse Engineering Malware”. This paper documents the tools and processes used to analyze the msrll.exe malware.
Laboratory Setup

Hardware

The laboratory hardware consists of an HP OmniBook XE3 laptop running with the following specs:

Intel Pentium III Processor
700 MHz
384MB Memory
10GB Hard Disk Drive
DVD/CD-ROM Drive
1.44MB 3.5" Floppy drive
10/100MB Integrated Network Interface
56Kbps Integrated Modem

Networking

The laboratory networking setup uses a VMware virtual network. The virtual network allows all network activity to be contained on the laptop between the virtual PC’s and the host system. To contain the malware fully the physical network interface on the laptop is not plugged in. The VMware virtual network emulates a hub, which is convenient for sniffing network traffic.

Software

The key software in this laboratory is VMware Workstation 4.5.1. VMware allows you to run multiple operating systems on one physical PC by creating virtual PC’s that share the physical resources of the host system.

The laptop’s Microsoft Windows 2000 Server SP4 is the Host system that has VMware installed. There are 2 virtual PC’s, one running Microsoft Windows XP SP1 and one running Red Hat Linux 9. Each virtual PC has been preconfigured with software tools to analyze the malware. Also, each virtual PC is contained in a folder on the laptop which has been backed up using WinZip for later recovery of the base or clean system. This allows for quick restores to a clean state for each virtual PC. The host system has been imaged using Ghost to allow for complete system recovery of the host system and all virtual systems.

Windows XP SP1 Software Tools:
**Ghost**
Version: 7.5
By: Symantec
[http://www.symantec.com](http://www.symantec.com)

Description:
Symantec Ghost is a disk imaging software that can backup a entire disk to a image file for later recovery on that image. I used Ghost to image the entire laboratory laptop hard disk so that in the event of infection at the host level I could restore the entire system back to a clean state.

**md5sum**
Version: GNU textutils 2.0
By: Ulrich Drepper

Description:
md5sum will calculate the md5 hash of a file. Knowing the md5 hash of a file will allow you to do file comparisons to determine if the files are the exact same. If a file has changed even in the slightest way the md5 hash of the two files should not match therefore revealing that the file has been modified in some way. I used md5sum to do file comparisons on the different copies of the malware msrll.exe.

**BinText**
Version: 3.00
By: Foundstone Inc.
[http://www.foundstone.com](http://www.foundstone.com)

Description:
BinText allows you to view the ASCII text, Unicode text, and resource strings contained in any file. By viewing the ASCII text, Unicode text, and resource strings in a binary file you can begin to get hints about its functionality, if it is packed or unpacked, and the memory addresses of interesting functions. BinText was used in my analysis of the different copies of msrll.exe malware file to gain hints on packing method, functionality, and memory addresses of certain interesting code.

**Regshot**
Version: 1.61e5 Final
By: TiANWEi
[http://regshot.yeah.net](http://regshot.yeah.net)
[http://regshot.ist.md](http://regshot.ist.md)

Description:
Regshot allows you to take 2 snap shots of the registry on a system and compare them. When you compare the snap shots you will get a list displaying the keys and
values that have been added, deleted, and modified. You can save your snap shots for later comparison. This is useful when you want to figure out what changes a malware made to a system. Regshot was used to first take a snap shot of the system before infection with the msrll.exe malware. Another snap shot of the system was taken after the msrll.exe malware ran. The comparison shows all the changes to the registry keys and values. This helps to figure out what the malware did and what filtering to do in examination of other log files like those from FileMon and RegMon.

**FileMon**
Version: 6.12  
By: Mark Russinovich and Bryce Cogswell of Sysinternals  
http://www.sysinternals.com

**Description:**
FileMon monitors and displays file system changes. You can save the logged changes to a file for later review. This is useful in finding detailed file system access during a specific period of time like during infection with the malware. A key area it shows is attempts not just successful file access. Sometimes the errors are more revealing than the successful entries. FileMon was used to record all file access during the initial infection of the system with the msrll.exe malware.

**RegMon**
Version: 6.12  
By: Mark Russinovich and Bryce Cogswell of Sysinternals  
http://www.sysinternals.com

**Description:**
RegMon will monitor all registry activity and display it on screen. You can save the log to a file for later review. RegMon will show what programs are accessing the registry and what registry keys and values they are reading or writing. RegMon was used to monitor the registry while infecting the system with the msrll.exe malware.

**TDIMon**
Version: 1.0  
By: Mark Russinovich of Sysinternals  
http://www.sysinternals.com

**Description:**
TDIMon is used to monitor TCP and UDP traffic on the system. This can help with monitoring what the malware does with network communications. An example would be opening a port on the system to listen for connections. TDIMon was used to monitor TCP and UDP traffic during the infection of the system with the msrll.exe malware.
**Process Explorer**  
Version: 8.52  
By: Mark Russinovich of Sysinternals  
[http://www.sysinternals.com](http://www.sysinternals.com)

Description:  
Process Explorer displays processes that are running on the local system along with their PID, description, and company name if any related to the process. It also shows in the lower window pane open handles or dlls depending on what mode it is in. Process Explorer allows you to drill down on each process listed by double clicking on the process. This reveals a great deal of information about the running process like TCP/IP connections or the path to the program that created the process. Process Explorer was used to monitor process the msrll.exe malware created and to end them during certain points of analysis.

**AutoRuns**  
Version: 5.01  
By: Mark Russinovich and Bryce Cogswell of Sysinternals  
[http://www.sysinternals.com](http://www.sysinternals.com)

Description:  
AutoRuns shows all the registry entries that are running programs during startup of the system. This is a common way for malware to auto start on a system. AutoRuns was used to check for msrll.exe changes to the auto starting entries in the registry.

**UltraEdit-32**  
Version: 10.10a  
By: IDM Computer Solutions Inc.  
[http://www.ultraedit.com](http://www.ultraedit.com)

Description:  
UltraEdit-32 can edit text, html, hex, and program source code. UltraEdit-32 was used to view files in hex mode and to view saved log files to search using it’s advance search features.

**nc (Netcat)**  
Version: 1.10  
By: Hobbit  

Description:  
Netcat or nc is often called the network Swiss army knife because there are many uses for this tool. Netcat was used to transfer snort log files from the Red Hat 9 virtual
system to the Windows XP SP1 virtual system so that they could be viewed in UltraEdit-32 to allow for my preferred method of search and examining the file. Netcat was also used to setup listeners on the Linux system to capture any requests to certain ports from the infected system.

**PESniffer**
Version: 1.06  
By: SkymarShall/CST  
http://start.at/skymarshall (Not Active)

Description:
PE-Sniffer can scan a file for various packed executable encodings like ASPack. PE-Sniffer was used to scan msrl.exe malware for the packed executable encodings.

**PEInfo**
Version: unknown  
By: Tom Liston  
Not available to public. This tool was obtained from the SANS Reverse Engineering Malware instructor lead on-line training cd-rom.

Description:
PEInfo allows you to see the packed executable structure. By viewing the structure details you maybe given hints as to what method was used to pack the executable. PEInfo was used to figure out what packing method was used on msrl.exe malware.

**ASPACKDIE**
Version: 1.41  
By: y0da  
http://y0da.cjb.net

Description:
ASPACKDIE is an ASPACK packed executable unpacker. ASPACKDIE was used to unpack the msrl.exe malware.

**IDA Pro**
Version: 4.6  
By: DataRescue  
http://www.datarescue.com

Description:
IDA Pro is a disassembler and debugger. I used IDA Pro to sift through the msrl.exe disassembled code in search for clues of its functionality. I especially liked the flowchart feature which help me find the different decisions branches in the code.
**OllyDbg**
Version: 1.10
By: Oleh Yuschuk
[http://home.t-online.de/home/ollydbg](http://home.t-online.de/home/ollydbg)

Description:
OllyDbg is a 32-bit debugger that runs on Windows. OllyDbg was used to analyze msrll.exe disassembled code while running msrll.exe within OllyDbg. This allowed me to set break points at key areas in the code to further understand the functioning of the malware. OllyDbg was also used to patch msrll.exe so that it didn’t require a proper password to authenticate.

**Red Hat 9 Software:**

**SNORT**
Version: 2.0.4
By: Martin Roesch
[http://www.snort.org](http://www.snort.org)

Description:
SNORT is a network sniffer and an intrusion detection system or IDS. It is used here as a network sniffer to capture packets on the virtual network for analysis.

**ircd**
Version: 2.8/hybrid-6.3.1
[http://www.ircd-hybrid.com](http://www.ircd-hybrid.com)

Description:
ircd is an IRC server daemon that runs on most UNIX based platforms. It is used here to run an irc server on the Red Hat 9 system to provoke additional behavior from the malware.

**irc**
Version: 20030709

Description:
irc is an IRC command line client for Unix/Linux. It is used here to interact with the ircd server and to further provoke and analyze the malware.
Properties of the Malware Specimen

Malware File: msrll.exe
File Type: executable
File Size: 41,984 bytes
MD5 Hash: 84acfe96a98590813413122c12c11aaa

Operating Systems Effected:
Microsoft Windows 9x, 2000, XP

Strings Embedded in File:

Fig. A shows the interesting strings found in the msrll.exe malware file before unpacking.

<table>
<thead>
<tr>
<th>File pos</th>
<th>Mem pos</th>
<th>ID</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000004D</td>
<td>0040004D</td>
<td>0</td>
<td>!This program cannot be run in DOS mode.</td>
</tr>
<tr>
<td>00000178</td>
<td>00400178</td>
<td>0</td>
<td>.text</td>
</tr>
<tr>
<td>000001A0</td>
<td>004001A0</td>
<td>0</td>
<td>.data</td>
</tr>
<tr>
<td>000001F0</td>
<td>004001F0</td>
<td>0</td>
<td>.idata</td>
</tr>
<tr>
<td>00000218</td>
<td>00400218</td>
<td>0</td>
<td>.aspack</td>
</tr>
<tr>
<td>00000240</td>
<td>00400240</td>
<td>0</td>
<td>.adata</td>
</tr>
</tbody>
</table>

Fig. B shows the interesting strings found in the msrll.exe malware file after it was unpacked using aspackdie.

<table>
<thead>
<tr>
<th>File pos</th>
<th>Mem pos</th>
<th>ID</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000004D</td>
<td>0040004D</td>
<td>0</td>
<td>!This program cannot be run in DOS mode.</td>
</tr>
<tr>
<td>00000080</td>
<td>00400080</td>
<td>0</td>
<td>PE</td>
</tr>
<tr>
<td>00000178</td>
<td>00400178</td>
<td>0</td>
<td>.text</td>
</tr>
<tr>
<td>000001A0</td>
<td>004001A0</td>
<td>0</td>
<td>.data</td>
</tr>
<tr>
<td>000001C8</td>
<td>004001C8</td>
<td>0</td>
<td>.bss</td>
</tr>
<tr>
<td>000001F0</td>
<td>004001F0</td>
<td>0</td>
<td>.idata</td>
</tr>
<tr>
<td>00000218</td>
<td>00400218</td>
<td>0</td>
<td>.aspack</td>
</tr>
<tr>
<td>00000240</td>
<td>00400240</td>
<td>0</td>
<td>.adata</td>
</tr>
<tr>
<td>00000268</td>
<td>00400268</td>
<td>0</td>
<td>.newIID</td>
</tr>
<tr>
<td>0000130D</td>
<td>0040130D</td>
<td>0</td>
<td>PW</td>
</tr>
<tr>
<td>00001326</td>
<td>00401326</td>
<td>0</td>
<td>?insmod</td>
</tr>
<tr>
<td>0000132E</td>
<td>0040132E</td>
<td>0</td>
<td>?rmmod</td>
</tr>
<tr>
<td>00001335</td>
<td>00401335</td>
<td>0</td>
<td>?lsmod</td>
</tr>
<tr>
<td>00001399</td>
<td>00401399</td>
<td>0</td>
<td>%s: &lt;mod name&gt;</td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>--------------------------------------</td>
<td></td>
</tr>
<tr>
<td>000013A8</td>
<td>004013A8</td>
<td><code>%s: mod list full</code></td>
<td></td>
</tr>
<tr>
<td>000013BA</td>
<td>004013BA</td>
<td><code>%s: err: %u</code></td>
<td></td>
</tr>
<tr>
<td>000013C6</td>
<td>004013C6</td>
<td><code>mod_init</code></td>
<td></td>
</tr>
<tr>
<td>000013CF</td>
<td>004013CF</td>
<td><code>mod_free</code></td>
<td></td>
</tr>
<tr>
<td>000013D8</td>
<td>004013D8</td>
<td><code>%s: cannot init %s</code></td>
<td></td>
</tr>
<tr>
<td>000013EB</td>
<td>004013EB</td>
<td><code>%s: %s loaded (%u)</code></td>
<td></td>
</tr>
<tr>
<td>000013FE</td>
<td>004013FE</td>
<td><code>%s: mod already loaded</code></td>
<td></td>
</tr>
<tr>
<td>00001416</td>
<td>00401416</td>
<td><code>%s: %s err %u</code></td>
<td></td>
</tr>
<tr>
<td>000015B5</td>
<td>004015B5</td>
<td><code>%s: %s not found</code></td>
<td></td>
</tr>
<tr>
<td>000015C5</td>
<td>004015C5</td>
<td><code>%s: unloading %s</code></td>
<td></td>
</tr>
<tr>
<td>000016AE</td>
<td>004016AE</td>
<td>[%u]: %s hinst=%x</td>
<td></td>
</tr>
<tr>
<td>00001712</td>
<td>00401712</td>
<td>unloading %s</td>
<td></td>
</tr>
<tr>
<td>000017A0</td>
<td>004017A0</td>
<td>%s: invalid_addr: %s</td>
<td></td>
</tr>
<tr>
<td>000017B5</td>
<td>004017B5</td>
<td>%s %s [port]</td>
<td></td>
</tr>
<tr>
<td>000018E8</td>
<td>004018E8</td>
<td>finished %s</td>
<td></td>
</tr>
<tr>
<td>00001A40</td>
<td>00401A40</td>
<td>%s &lt;ip&gt; &lt;port&gt; &lt;t_time&gt; &lt;delay&gt;</td>
<td></td>
</tr>
<tr>
<td>00001B32</td>
<td>00401B32</td>
<td>sockopt: %u</td>
<td></td>
</tr>
<tr>
<td>00001B3E</td>
<td>00401B3E</td>
<td>sendto err: %u</td>
<td></td>
</tr>
<tr>
<td>00001B4D</td>
<td>00401B4D</td>
<td>sockraw: %u</td>
<td></td>
</tr>
<tr>
<td>00001B59</td>
<td>00401B59</td>
<td>syn: done</td>
<td></td>
</tr>
<tr>
<td>00001FBC</td>
<td>00401FBC</td>
<td>%s &lt;ip&gt; &lt;duration&gt; &lt;delay&gt;</td>
<td></td>
</tr>
<tr>
<td>00002096</td>
<td>00402096</td>
<td>sendto: %u</td>
<td></td>
</tr>
<tr>
<td>000020A2</td>
<td>004020A2</td>
<td>jolt2: done</td>
<td></td>
</tr>
<tr>
<td>00002260</td>
<td>00402260</td>
<td>%s &lt;ip&gt; &lt;p size&gt; &lt;duration&gt; &lt;delay&gt;</td>
<td></td>
</tr>
<tr>
<td>00002356</td>
<td>00402356</td>
<td>Err: %u</td>
<td></td>
</tr>
<tr>
<td>0000235E</td>
<td>0040235E</td>
<td>smurf done</td>
<td></td>
</tr>
<tr>
<td>000025DE</td>
<td>004025DE</td>
<td>&amp;err: %u</td>
<td></td>
</tr>
<tr>
<td>00002753</td>
<td>00402753</td>
<td>?ping</td>
<td></td>
</tr>
<tr>
<td>00002759</td>
<td>00402759</td>
<td>?udp</td>
<td></td>
</tr>
<tr>
<td>0000275E</td>
<td>0040275E</td>
<td>?syn</td>
<td></td>
</tr>
<tr>
<td>00002763</td>
<td>00402763</td>
<td>?smurf</td>
<td></td>
</tr>
<tr>
<td>0000276A</td>
<td>0040276A</td>
<td>?jolt</td>
<td></td>
</tr>
<tr>
<td>00002820</td>
<td>00402820</td>
<td>PONG :%s</td>
<td></td>
</tr>
<tr>
<td>0000299D</td>
<td>0040299D</td>
<td>%s!%s@%s</td>
<td></td>
</tr>
<tr>
<td>00002B3D</td>
<td>00402B3D</td>
<td>%s!%s</td>
<td></td>
</tr>
<tr>
<td>00002B7D</td>
<td>00402B7D</td>
<td>irc.nick</td>
<td></td>
</tr>
<tr>
<td>00002BE0</td>
<td>00402BE0</td>
<td>NICK %s</td>
<td></td>
</tr>
<tr>
<td>00002C56</td>
<td>00402C56</td>
<td>MODE</td>
<td></td>
</tr>
<tr>
<td>00002E34</td>
<td>00402E34</td>
<td>?bu</td>
<td></td>
</tr>
<tr>
<td>00002EEA</td>
<td>00402EEA</td>
<td>NETWORK=</td>
<td></td>
</tr>
<tr>
<td>00002FF8</td>
<td>00402FF8</td>
<td>irc.pre</td>
<td></td>
</tr>
<tr>
<td>000032A8</td>
<td>004032A8</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032AC</td>
<td>004032AC</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032B0</td>
<td>004032B0</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032B5</td>
<td>004032B5</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032B9</td>
<td>004032B9</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032BF</td>
<td>004032BF</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032C2</td>
<td>004032C2</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032C7</td>
<td>004032C7</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032CC</td>
<td>004032CC</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032D2</td>
<td>004032D2</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032D9</td>
<td>004032D9</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>000032E1</td>
<td>004032E1</td>
<td>NICK %s</td>
<td></td>
</tr>
<tr>
<td>000032EB</td>
<td>004032EB</td>
<td>NICK</td>
<td></td>
</tr>
<tr>
<td>000032F0</td>
<td>004032F0</td>
<td>%s %s</td>
<td></td>
</tr>
<tr>
<td>0000345C</td>
<td>0040345C</td>
<td>CmP</td>
<td></td>
</tr>
<tr>
<td>000036B0</td>
<td>004036B0</td>
<td>irc.chan</td>
<td></td>
</tr>
<tr>
<td>00003775</td>
<td>00403775</td>
<td>%s %s</td>
<td></td>
</tr>
</tbody>
</table>
Key fingerprint = AF19 FA27 2F94 998D FDB5 DE3D F8B5 06E4 A169 4E46

0000377B 0040377B 0  WHO %s
00003A45 00403A45 0 USERHOST %s
00003A52 00403A52 0 logged into %s(%s) as %s
00003B99 00403B99 0 nick.pre
00003BA2 00403BA2 0 %s=%04u
00003BAA 00403BAA 0 irc.user
00003BB3 00403BB3 0 irc.usereal
00003BBF 00403BBF 0 irc.real
00003BC8 00403BC8 0 irc.pass
00003BE0 00403BE0 0 tsend(): connection to %s:%u failed
00003C20 00403C20 0 USER %s localhost 0 :%s
00003C38 00403C38 0 NICK %s
000040BA 004040BA 0 PING
000040BF 004040BF 0 PRIVMSG
000040C7 004040C7 0 001
000040D0 004040D0 0 QUIT
000040D5 004040D5 0 352
000040D9 004040D9 0 302
000040DD 004040DD 0 303
000040E1 004040E1 0 005
000040E5 004040E5 0 PART
000040EA 004040EA 0 005
000040FD 004040FD 0 t&
00004100 00404100 0 trecv(): Disconnected from %s err:%u
0000424B 0040424B 0 NOTIC E
00004472 00404472 0 %s %s :%s
00004AD7 00404AD7 0 PING :ok
00004B28 00404B28 0 expect the worst
00004B39 00404B39 0 s_check: send error to %s disconnecting
00004B54 00404B54 0 irc.knick
00004B6B 00404B6B 0 servers
00004B74 00404B74 0 s_check: trying %s
00004DAA 00404DAA 0 Ph9K@
00004ED5 00404ED5 0 PhkK@
00004F41 00404F41 0 Sh$k@
00005052 00405052 0 %s.mode
0000505A 0040505A 0 MODE %s %s
00005078 00405078 0 SHRP@
000050DA 004050DA 0 Sh$%@
0000559F 0040559F 0 aop
000055A3 004055A3 0 mode %s +o %s
000055B2 004055B2 0 akick
000055B8 004055B8 0 mode %s +b %s %s
000055CA 004055CA 0 KICK %s %s
Key fingerprint = AF19 FA27 2F94 998D FDB5 DE3D F8B5 06E4 A169 4E46

00005760 00405760 0 irc.pre
00005781 00405781 0 Set an irc sock to preform %s command on
000057AB 004057AB 0 Type
000057B3 004057B3 0 %csklist
000057BC 004057BC 0 to view current sockets, then
000057DC 004057DC 0 %cdccsk
000057E4 004057E4 0 <#>
000058B4 004058B4 0 %s: dll loaded
000058C3 004058C3 0 %s: %d
000059E1 004059E1 0 said %s to %s
000059EF 004059EF 0 usage: %s <target> "text"
00005A74 00405A74 0 %s not on %s
00005B1B 00405B1B 0 PASS
00005B20 00405B20 0 %s logged in
00005B48 00405B48 0 sys: %s bot: %s
00005B52 00405B52 0 preformance counter not avail
00005C2B 00405C2B 0 usage: %s <cmd>
00005C3B 00405C3B 0 %s free'd
00005C45 00405C45 0 unable to free %s
00005C5D 00405C5D 0 later!
00005CB4 00405CB4 0 unable to %s errno:%u
00005D40 00405D40 0 service:%s user:%s inet connection:%c
00005DAE 00405DAE 0 ???
00005E1E 00405E1E 0 kill
00005E23 00405E23 0 %s-%s %s
00005F8F 00405F8F 0 %s: %s
00005F96 00405F96 0 %s: somefile
00006000 00406000 0 host: %s ip: %s
00006269 00406269 0 capGetDriverDescriptionA
00006282 00406282 0 9x
00006285 00406285 0 2k
00006298 00406298 0 XP
000062BB 004062BB 0 XP++
00006292 00406292 0 cpus:%u
0000629B 0040629B 0 CAM
000062A0 004062A0 0 WIN%s (u:%s)s%s mem:(%u/%u) %u%% %s %s
000065C6 004065C6 0 open
000065CB 004065CB 0 %s: %s (%u)
00006703 00406703 0 NICK
00006708 00406708 0 %s %s
00006754 00406754 0 %s bad args
000067D6 004067D6 0 aop
000067DA 004067DA 0 akick
000067E0 004067E0 0 OP
000067E3 004067E3 0 KICK
000067E8 004067E8 0 %s[%u] %s
000067F2 004067F2 0 %s removed
000067FD 004067FD 0 couldn't find %s
0000680D 0040680D 0 %s added
00006916 00406916 0 %s allready in list
0000692A 0040692A 0 usage: %s +/- <host>
000069EB 004069EB 0 jtram.conf
000069F6 004069F6 0 %s /t %s
000069FF 004069FF 0 jtr.home
00006A08 00406A08 0 %s\%s
00006A0E 00406A0E 0 %s: possibly failed: code %u
00006A2B 00406A2B 0 %s: possibly failed
00006A3F 00406A3F 0 %s: exec of %s failed err: %u
00006A90 00406A90 0 u.exf
00006CBC 00406CBC 0 jtr.id
00006CC3 00406CC3 0 %s: <url> <id>
00006E79 00406E79 0 %s
00006EBD 00406EBD 0 IRC
00006EC2 00406EC2 0 DCC
00006EC8 00406EC8 0 DATH
00006ED0 00406ED0 0 IATH
00006ED7 00406ED7 0 IREG
00006EDD 00406EDD 0 CLON
00006EE3 00406EE3 0 ICON
00006EE9 00406EE9 0 RNL
00006EEE 00406EEE 0 RBN
00006EF3 00406EF3 0 WSN
00006EF8 00406EF8 0 WCON
00006EEF 00406EEF 0 LSN
00006F03 00406F03 0 SSL
00006F08 00406F08 0 >>S
00006F40 00406F40 0 #%u [fd:%u] %s:%u [%s%s] last:%u
00006F63 00406F63 0 |\=> [n:%s fh:%s] (%s)
00006F7D 00406F7D 0 |
00006F82 00406F82 0 |--->[s] (%u) %s
00006F96 00406F96 0 | |-%[s%s] [%s]
00006FAD 00406FAD 0 |=> (%s) (%.8x)
00007360 00407360 0 %s <pass> <salt>
000073C8 004073C8 0 %s <nick> <chan>
00007435 00407435 0 !%s!
0000748B 0040748B 0 PING %s
000074C9 004074C9 0 mIRC v6.12 Khaled Mardam-Bey
000074E7 004074E7 0 VERSION %s
0000751C 0040751C 0 dcc.pass
00007525 00407525 0 temp add %s
0000766A 0040766A 0 %s%u-%s
00007672 00407672 0 wb
000077C8 004077C8 0 %s opened (%u)
000076A0 004076A0 0 %u bytes from %s in %u seconds saved to %s
000076CB 004076CB 0 (%s %s): incomplete! %u bytes
000076E9 004076E9 0 couldnt open %s err:%u
00007700 00407700 0 (%s) %s: %s
0000770C 0040770C 0 (%s) urlopen failed
00007720 00407720 0 (%s): inetopen failed
000077BE 004077BE 0 no file name in %s
000077D8 004077D8 0 %s created
000077E9 004077E9 0 %s %s to %s Ok
000077EE 004077EE 0 %0.2u/%0.2u/%0.2u %0.2u:%0.2u %15s %s
00007F09 00407F09 0 %s (err: %u)
00008085 00408085 0 err: %u
000080F8 004080F8 0 %s %s:ok
00008165 00408165 0 unable to %s %s (err: %u)
000081F5 004081F5 0 %s-%s %s
00008200 00408200 0 %s (%u.%u.%u.%u)
00008489 00408489 0 [%s][%s] %s
00008595 00408595 0 closing %u [%s:%u]
000085A8 004085A8 0 unable to close socket %u
000087E2 004087E2 0 using sock #%u %s:%u (%s)
000087FD 004087FD 0 Invalid sock
usage %s <socks #>
leaves %s
:o * * :%s
hmm
joins: %s
ACCEPT
resume
err: %u
DCC ACCEPT %s %s %s
dcc_resume: cant find port %s
send
dcc.dir
%s%s%s%s
unable to open (%s): %u
resuming dcc from %s to %s
DCC RESUME %s %s %u
?si
?ssl
?clone
?clones
?login
?uptime
?reboot
?status
?jump
?nick
?echo
?hush
?wget
?join
?op
?aop
?akick
?part
?dump
?set
?die
?md5p
?free
?raw
?update
?hostname
?fif
?!fif
?del
?pwd
?copy
?play
?move
?dir
?sums
?ls
?cd
?rmdir
?mkdir
?run
?exec
Key fingerprint = AF19 FA27 2F94 998D FDB5 DE3D F8B5 06E4 A169 4E46
Key fingerprint = AF19 FA27 2F94 998D FDB5 DE3D F8B5 06E4 A169 4E46

<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
<th>Arguments</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000EB61</td>
<td>0040EB61</td>
<td>%s</td>
<td></td>
</tr>
<tr>
<td>0000EB65</td>
<td>0040EB65</td>
<td>msg</td>
<td></td>
</tr>
<tr>
<td>0000EB80</td>
<td>0040EB80</td>
<td>%s: [NETWORK</td>
<td>all] %s &quot;&lt;parm&gt;&quot; ...</td>
</tr>
<tr>
<td>0000EE20</td>
<td>0040EE20</td>
<td>USER %s localhost 0 ::%s</td>
<td></td>
</tr>
<tr>
<td>0000EE38</td>
<td>0040EE38</td>
<td>NICK %s</td>
<td></td>
</tr>
<tr>
<td>0000F100</td>
<td>0040F100</td>
<td>md5</td>
<td></td>
</tr>
<tr>
<td>0000F140</td>
<td>0040F140</td>
<td>md5.c</td>
<td></td>
</tr>
<tr>
<td>0000F8F1</td>
<td>0040F8F1</td>
<td>buf != NULL</td>
<td></td>
</tr>
<tr>
<td>0000F99F</td>
<td>0040F99F</td>
<td>hash != NULL</td>
<td></td>
</tr>
<tr>
<td>0000FAC1</td>
<td>0040FAC1</td>
<td>abc</td>
<td></td>
</tr>
<tr>
<td>0000FAC5</td>
<td>0040FAC5</td>
<td>message digest</td>
<td></td>
</tr>
<tr>
<td>0000FAD4</td>
<td>0040FAD4</td>
<td>abcdefghijklmnopqrstuvwxyz</td>
<td></td>
</tr>
<tr>
<td>0000FB00</td>
<td>0040FB00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ABCD

0000EB61   0040EB61   0   %s
0000EB65   0040EB65   0   msg
0000EB80   0040EB80   0   %s: [NETWORK|all] %s "<parm>" ...
0000EE20   0040EE20   0   USER %s localhost 0 ::%s
0000EE38   0040EE38   0   NICK %s
0000F100   0040F100   0   md5
0000F140   0040F140   0   md5.c
0000F8F1   0040F8F1   0   buf != NULL
0000F99F   0040F99F   0   hash != NULL
0000FAC1   0040FAC1   0   abc
0000FAC5   0040FAC5   0   message digest
0000FAD4   0040FAD4   0   abcdefghijklmnopqrstuvwxyz
0000FB00   0040FB00   0

ABCD

0000EB61   0040EB61   0   %s
0000EB65   0040EB65   0   msg
0000EB80   0040EB80   0   %s: [NETWORK|all] %s "<parm>" ...
0000EE20   0040EE20   0   USER %s localhost 0 ::%s
0000EE38   0040EE38   0   NICK %s
0000F100   0040F100   0   md5
0000F140   0040F140   0   md5.c
0000F8F1   0040F8F1   0   buf != NULL
0000F99F   0040F99F   0   hash != NULL
0000FAC1   0040FAC1   0   abc
0000FAC5   0040FAC5   0   message digest
0000FAD4   0040FAD4   0   abcdefghijklmnopqrstuvwxyz
0000FB00   0040FB00   0

ABCD

0000EB61   0040EB61   0   %s
0000EB65   0040EB65   0   msg
0000EB80   0040EB80   0   %s: [NETWORK|all] %s "<parm>" ...
0000EE20   0040EE20   0   USER %s localhost 0 ::%s
0000EE38   0040EE38   0   NICK %s
0000F100   0040F100   0   md5
0000F140   0040F140   0   md5.c
0000F8F1   0040F8F1   0   buf != NULL
0000F99F   0040F99F   0   hash != NULL
0000FAC1   0040FAC1   0   abc
0000FAC5   0040FAC5   0   message digest
0000FAD4   0040FAD4   0   abcdefghijklmnopqrstuvwxyz
0000FB00   0040FB00   0

ABCD

0000EB61   0040EB61   0   %s
0000EB65   0040EB65   0   msg
0000EB80   0040EB80   0   %s: [NETWORK|all] %s "<parm>" ...
0000EE20   0040EE20   0   USER %s localhost 0 ::%s
0000EE38   0040EE38   0   NICK %s
0000F100   0040F100   0   md5
0000F140   0040F140   0   md5.c
0000F8F1   0040F8F1   0   buf != NULL
0000F99F   0040F99F   0   hash != NULL
0000FAC1   0040FAC1   0   abc
0000FAC5   0040FAC5   0   message digest
0000FAD4   0040FAD4   0   abcdefghijklmnopqrstuvwxyz
0000FB00   0040FB00   0

ABCD
Key fingerprint = AF19 FA27 2F94 998D FDB5 DE3D F8B5 06E4 A169 4E46

Hashes built-in:

- SHA-512
- SHA-384
- SHA-256

Block Chaining Modes:

- CFB
- OFB
- CTR

PRNG:

- Yarrow

PK Algs:

- RSA
- DH
- ECC

Compiler:

- WIN32 platform detected.
- GCC compiler detected.

Various others: BASE64 MPI HMAC

Microsoft Base Cryptographic Provider v1.0
Key fingerprint = AF19 FA27 2F94 998D FDB5 DE3D F8B5 06E4 A169 4E46
0011BB32 0051BB32 0 Process32First
0011BB46 0051BB46 0 Process32Next
0011BB56 0051BB56 0 QueryPerformanceFrequency
0011BB72 0051BB72 0 ReadFile
0011BB7E 0051BB7E 0 ReleaseMutex
0011BB8E 0051BB8E 0 RemoveDirectoryA
0011BB92 0051BB92 0 SetConsoleCtrlHandler
0011BBBA 0051BBBA 0 SetCurrentDirectoryA
0011BBDA 0051BBDA 0 SetFilePointer
0011BBE6 0051BBE6 0 SetUnhandledExceptionFilter
0011BC06 0051BC06 0 Sleep
0011BC0E 0051BC0E 0 TerminateProcess
0011BC22 0051BC22 0 WaitForSingleObject
0011BC3A 0051BC3A 0 WriteFile
0011BC46 0051BC46 0 _itoa
0011BC4E 0051BC4E 0 _stat
0011BC56 0051BC56 0 _strdup
0011BC62 0051BC62 0 _stricmp
0011BC6E 0051BC6E 0 __getmainargs
0011BC7E 0051BC7E 0 __p__envir
0011BC8E 0051BC8E 0 __p__fmode
0011BC9E 0051BC9E 0 __set_app_type
0011BCB2 0051BCB2 0 __beginthread
0011BCC2 0051BCC2 0 __cexit
0011BCE6 0051BCE6 0 __errno
0011BCEA 0051BCEA 0 __fileno
0011BCE6 0051BCE6 0 __io
0011BCFA 0051BCFA 0 __setmode
0011BD06 0051BD06 0 __vsnprintf
0011BD16 0051BD16 0 abort
0011BD1E 0051BD1E 0 atexit
0011BD2A 0051BD2A 0 atoi
0011BD32 0051BD32 0 clock
0011BD3A 0051BD3A 0 fclose
0011BD46 0051BD46 0 fflush
0011BD52 0051BD52 0 fgets
0011BD5A 0051BD5A 0 fopen
0011BD62 0051BD62 0 fprintf
0011BD6E 0051BD6E 0 fread
0011BD76 0051BD76 0 free
0011BD7E 0051BD7E 0 fwrite
0011BD8A 0051BD8A 0 malloc
0011BD96 0051BD96 0 memcopy
0011BDA2 0051BDA2 0 memset
0011BDAE 0051BDAE 0 printf
0011BDFA 0051BDFA 0 rand
0011BDBA 0051BDBA 0 raise
0011BDC2 0051BDC2 0 rand
0011BDBA 0051BDBA 0 realloc
0011BDD6 0051BDD6 0 setvbuf
0011BDE2 0051BDE2 0 signal
0011BDEE 0051BDEE 0 sprintf
0011BFAE 0051BFAE 0 srand
0011BE02 0051BE02 0 strcat
0011BE0E 0051BE0E 0 strchr
0011BEEA 0051BEEA 0 strcmp
0011BE26 0051BE26 0 strcpy
0011BE32 0051BE32 0 strerror
0011BE3E  0051BE3E  0  strcat
0011BE4A  0051BE4A  0  strcmp
0011BE56  0051BE56  0  strncpy
0011BE62  0051BE62  0  strstr
0011BE6E  0051BE6E  0  time
0011BE76  0051BE76  0  toupper
0011BE82  0051BE82  0  ShellExecuteA
0011BE92  0051BE92  0  DispatchMessageA
0011BEA6  0051BEA6  0  ExitWindowsEx
0011BEB6  0051BEB6  0  GetMessageA
0011BEC6  0051BEC6  0  PeekMessageA
0011BED6  0051BED6  0  GetFileVersionInfoA
0011BEEE  0051BEEE  0  VerQueryValueA
0011BF02  0051BF02  0  InternetCloseHandle
0011BF1A  0051BF1A  0  InternetGetConnectedState
0011BF36  0051BF36  0  InternetOpenA
0011BF46  0051BF46  0  InternetOpenUrlA
0011BF5A  0051BF5A  0  InternetReadFile
0011BF6E  0051BF6E  0  WSAGetLastError
0011BF82  0051BF82  0  WSAStartup
0011BF92  0051BF92  0  __WSAFDIsSet
0011BFA2  0051BFA2  0  accept
0011BFB2  0051BFB2  0  bind
0011BFC6  0051BFC6  0  closesocket
0011BFD6  0051BFD6  0  connect
0011BFE2  0051BFE2  0  gethostbyaddr
0011BFF2  0051BFF2  0  gethostbyname
0011CO02  0051CO02  0  gethostname
0011CO12  0051CO12  0  getsockname
0011CO22  0051CO22  0  htonl
0011CO2A  0051CO2A  0  htons
0011CO32  0051CO32  0  inet_addr
0011CO3E  0051CO3E  0  inet_ntoa
0011CO4A  0051CO4A  0  ioctlsocket
0011CO5A  0051CO5A  0  listen
0011CO66  0051CO66  0  ntohl
0011CO6E  0051CO6E  0  recv
0011CO76  0051CO76  0  select
0011CO82  0051CO82  0  send
0011CO8A  0051CO8A  0  sendto
0011CO96  0051CO96  0  setsockopt
0011COA6  0051COA6  0  shutdown
0011COB2  0051COB2  0  socket
0011C0FC  0051C0FC  0  ADVAPI32.DLL
0011C1FC  0051C1FC  0  KERNEL32.dll
0011C21C  0051C21C  0  msvcrt.dll
0011C2E0  0051C2E0  0  msvcrt.dll
0011C2F0  0051C2F0  0  SHELL32.DLL
0011C30C  0051C30C  0  USER32.dll
0011C320  0051C320  0  VERSION.dll
0011C340  0051C340  0  WININET.DLL
0011C3B4  0051C3B4  0  W32_32.DLL
0011D071  0051D071  0  VirtualAlloc
0011D07E  0051D07E  0  VirtualFree
0011D441  0051D441  0  kernel32.dll
0011D44E  0051D44E  0  ExitProcess
0011D45A  0051D45A  0  user32.dll
0011D465 0051D465 0 MessageBoxA
0011D471 0051D471 0 wssprintfA
0011D47B 0051D47B 0 LOADER ERROR
0011D488 0051D488 0 The procedure entry point %s could not be located in the dynamic link library %s
0011D4D9 0051D4D9 0 The ordinal %u could not be located in the dynamic link library %s
0011DF6C 0051DF6C 0 kernel32.dll
0011DF7B 0051DF7B 0 GetProcAddress
0011DF8C 0051DF8C 0 GetModuleHandleA
0011DF9F 0051DF9F 0 LoadLibraryA
0011E074 0051E074 0 advapi32.dll
0011E081 0051E081 0 msvcrtd.dll
0011E08C 0051E08C 0 msvcrtd.dll
0011E097 0051E097 0 shell32.dll
0011E0A3 0051E0A3 0 user32.dll
0011E0AE 0051E0AE 0 version.dll
0011E0BA 0051E0BA 0 wininet.dll
0011E0C6 0051E0C6 0 ws2_32.dll
0011E113 0051E113 0 AdjustTokenPrivileges
0011E12B 0051E12B 0 _itoa
0011E133 0051E133 0 __getmainargs
0011E143 0051E143 0 ShellExecuteA
0011E153 0051E153 0 DispatchMessageA
0011E166 0051E166 0 GetFileVersionInfoA
0011E17C 0051E17C 0 InternetCloseHandle
0011E192 0051E192 0 WSAGetLastError
001200C8 005200C8 0 advapi32.dll
001200D7 005200D7 0 AdjustTokenPrivileges
001200EF 005200EF 0 CloseServiceHandle
00120104 00520104 0 CreateServiceA
00120115 00520115 0 CryptAcquireContextA
0012012C 0052012C 0 CryptGenRandom
0012013D 0052013D 0 CryptReleaseContext
00120153 00520153 0 GetUserNameA
00120162 00520162 0 LookupPrivilegeValueA
0012017A 0052017A 0 OpenProcessToken
0012018D 0052018D 0 OpenSCManagerA
0012019E 0052019E 0 RegCloseKey
001201AC 005201AC 0 RegCreateKeyExA
001201BE 005201BE 0 RegSetValueExA
001201CF 005201CF 0 RegisterServiceCtrlHandlerA
001201ED 005201ED 0 SetServiceStatus
00120200 00520200 0 StartServiceCtrlDispatcherA
0012021C 0052021C 0 kernel32.dll
0012022B 0052022B 0 AddAtomA
00120236 00520236 0 CloseHandle
00120244 00520244 0 CopyFileA
00120250 00520250 0 CreateDirectoryA
00120263 00520263 0 CreateFileA
00120271 00520271 0 CreateMutexA
00120280 00520280 0 CreatePipe
0012028D 0052028D 0 CreateProcessA
0012029E 0052029E 0 CreateToolhelp32Snapshot
001202B9 005202B9 0 DeleteFileA
001202C7 005202C7 0 DuplicateHandle
001202D9 005202D9 0 EnterCriticalSection
001202F0 005202F0 0 ExitProcess
<table>
<thead>
<tr>
<th>Address</th>
<th>Bytes</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>001202FE</td>
<td>005202FE</td>
<td>ExitThread</td>
</tr>
<tr>
<td>0012030B</td>
<td>0052030B</td>
<td>FileTimeToSystemTime</td>
</tr>
<tr>
<td>00120322</td>
<td>00520322</td>
<td>FindAtomA</td>
</tr>
<tr>
<td>0012032E</td>
<td>0052032E</td>
<td>FindClose</td>
</tr>
<tr>
<td>0012033A</td>
<td>0052033A</td>
<td>FindFirstFileA</td>
</tr>
<tr>
<td>0012034B</td>
<td>0052034B</td>
<td>FindNextFileA</td>
</tr>
<tr>
<td>0012035B</td>
<td>0052035B</td>
<td>FreeLibrary</td>
</tr>
<tr>
<td>00120369</td>
<td>00520369</td>
<td>GetAtomNameA</td>
</tr>
<tr>
<td>00120378</td>
<td>00520378</td>
<td>GetCommandLineA</td>
</tr>
<tr>
<td>0012038A</td>
<td>0052038A</td>
<td>GetCurrentDirectoryA</td>
</tr>
<tr>
<td>00120391</td>
<td>00520391</td>
<td>GetCurrentProcess</td>
</tr>
<tr>
<td>0012039F</td>
<td>0052039F</td>
<td>GetCurrentThreadId</td>
</tr>
<tr>
<td>001203A1</td>
<td>005203A1</td>
<td>GetExitCodeProcess</td>
</tr>
<tr>
<td>001203D9</td>
<td>005203D9</td>
<td>GetFullPathNameA</td>
</tr>
<tr>
<td>001203F0</td>
<td>005203F0</td>
<td>GetLastError</td>
</tr>
<tr>
<td>001203F1</td>
<td>005203F1</td>
<td>GetModuleFileNameA</td>
</tr>
<tr>
<td>0012040A</td>
<td>0052040A</td>
<td>GetProcAddress</td>
</tr>
<tr>
<td>00120410</td>
<td>00520410</td>
<td>GetSystemDirectoryA</td>
</tr>
<tr>
<td>0012041F</td>
<td>0052041F</td>
<td>GetSystemInfo</td>
</tr>
<tr>
<td>00120425</td>
<td>00520425</td>
<td>GetTempPathA</td>
</tr>
<tr>
<td>0012042F</td>
<td>0052042F</td>
<td>GetLastError</td>
</tr>
<tr>
<td>00120434</td>
<td>00520434</td>
<td>msvcrt.dll</td>
</tr>
<tr>
<td>0012043E</td>
<td>0052043E</td>
<td>_itoa</td>
</tr>
<tr>
<td>00120446</td>
<td>00520446</td>
<td>_mbsdup</td>
</tr>
<tr>
<td>00120457</td>
<td>00520457</td>
<td>_strcmpi</td>
</tr>
<tr>
<td>0012045F</td>
<td>0052045F</td>
<td>_beginthread</td>
</tr>
<tr>
<td>00120463</td>
<td>00520463</td>
<td>__p__environ</td>
</tr>
<tr>
<td>00120473</td>
<td>00520473</td>
<td>__p__fmode</td>
</tr>
<tr>
<td>00120489</td>
<td>00520489</td>
<td>__set_app_type</td>
</tr>
<tr>
<td>0012049E</td>
<td>0052049E</td>
<td>_beginthread</td>
</tr>
<tr>
<td>00120505</td>
<td>00520505</td>
<td>ExitThread</td>
</tr>
<tr>
<td>0012055F</td>
<td>0052055F</td>
<td>ExitThread</td>
</tr>
<tr>
<td>0012057B</td>
<td>0052057B</td>
<td>ExitThread</td>
</tr>
<tr>
<td>0012058B</td>
<td>0052058B</td>
<td>ReleaseMutex</td>
</tr>
<tr>
<td>00120595</td>
<td>00520595</td>
<td>RemoveDirectoryA</td>
</tr>
<tr>
<td>001205A8</td>
<td>005205A8</td>
<td>SetConsoleCtrlHandler</td>
</tr>
<tr>
<td>001205C0</td>
<td>005205C0</td>
<td>SetCurrentDirectoryA</td>
</tr>
<tr>
<td>001205D7</td>
<td>005205D7</td>
<td>SetFilePointer</td>
</tr>
<tr>
<td>001205E8</td>
<td>005205E8</td>
<td>SetUnhandledExceptionFilter</td>
</tr>
<tr>
<td>00120606</td>
<td>00520606</td>
<td>Sleep</td>
</tr>
<tr>
<td>00120621</td>
<td>00520621</td>
<td>WaitForSingleObject</td>
</tr>
<tr>
<td>00120637</td>
<td>00520637</td>
<td>WriteFile</td>
</tr>
<tr>
<td>00120641</td>
<td>00520641</td>
<td>msvcrt.dll</td>
</tr>
<tr>
<td>0012064E</td>
<td>0052064E</td>
<td>_itoa</td>
</tr>
<tr>
<td>00120656</td>
<td>00520656</td>
<td>__mbsdup</td>
</tr>
<tr>
<td>00120668</td>
<td>00520668</td>
<td>__strcmpi</td>
</tr>
<tr>
<td>00120671</td>
<td>00520671</td>
<td>msvcrt.dll</td>
</tr>
<tr>
<td>0012067E</td>
<td>0052067E</td>
<td>__getmainargs</td>
</tr>
<tr>
<td>0012068E</td>
<td>0052068E</td>
<td>__p__environ</td>
</tr>
<tr>
<td>0012069D</td>
<td>0052069D</td>
<td>__p__fmode</td>
</tr>
<tr>
<td>001206AA</td>
<td>005206AA</td>
<td>__set_app_type</td>
</tr>
<tr>
<td>001206BB</td>
<td>005206BB</td>
<td>__beginthread</td>
</tr>
<tr>
<td>Address</td>
<td>Offset</td>
<td>Function</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>-------------------</td>
</tr>
<tr>
<td>001206CA</td>
<td>005206CA</td>
<td>_cexit</td>
</tr>
<tr>
<td>001206D3</td>
<td>005206D3</td>
<td>_errno</td>
</tr>
<tr>
<td>001206DC</td>
<td>005206DC</td>
<td>_fileno</td>
</tr>
<tr>
<td>001206E6</td>
<td>005206E6</td>
<td>_iob</td>
</tr>
<tr>
<td>001206ED</td>
<td>005206ED</td>
<td>_onexit</td>
</tr>
<tr>
<td>001206F7</td>
<td>005206F7</td>
<td>_setmode</td>
</tr>
<tr>
<td>00120702</td>
<td>00520702</td>
<td>_vsnprintf</td>
</tr>
<tr>
<td>0012070F</td>
<td>0052070F</td>
<td>abort</td>
</tr>
<tr>
<td>00120717</td>
<td>00520717</td>
<td>atexit</td>
</tr>
<tr>
<td>00120720</td>
<td>00520720</td>
<td>atoi</td>
</tr>
<tr>
<td>00120727</td>
<td>00520727</td>
<td>clock</td>
</tr>
<tr>
<td>0012072F</td>
<td>0052072F</td>
<td>fclose</td>
</tr>
<tr>
<td>00120738</td>
<td>00520738</td>
<td>fflush</td>
</tr>
<tr>
<td>00120741</td>
<td>00520741</td>
<td>fgets</td>
</tr>
<tr>
<td>00120749</td>
<td>00520749</td>
<td>fopen</td>
</tr>
<tr>
<td>00120751</td>
<td>00520751</td>
<td>fprintf</td>
</tr>
<tr>
<td>0012075B</td>
<td>0052075B</td>
<td>fread</td>
</tr>
<tr>
<td>00120763</td>
<td>00520763</td>
<td>free</td>
</tr>
<tr>
<td>0012076A</td>
<td>0052076A</td>
<td>fwrite</td>
</tr>
<tr>
<td>00120773</td>
<td>00520773</td>
<td>malloc</td>
</tr>
<tr>
<td>0012077C</td>
<td>0052077C</td>
<td>memccpy</td>
</tr>
<tr>
<td>00120785</td>
<td>00520785</td>
<td>memset</td>
</tr>
<tr>
<td>0012078E</td>
<td>0052078E</td>
<td>printf</td>
</tr>
<tr>
<td>00120797</td>
<td>00520797</td>
<td>raise</td>
</tr>
<tr>
<td>0012079F</td>
<td>0052079F</td>
<td>rand</td>
</tr>
<tr>
<td>001207A6</td>
<td>005207A6</td>
<td>realloc</td>
</tr>
<tr>
<td>001207B0</td>
<td>005207B0</td>
<td>setvbuf</td>
</tr>
<tr>
<td>001207BA</td>
<td>005207BA</td>
<td>signal</td>
</tr>
<tr>
<td>001207C3</td>
<td>005207C3</td>
<td>sprintf</td>
</tr>
<tr>
<td>001207CD</td>
<td>005207CD</td>
<td>srand</td>
</tr>
<tr>
<td>001207D5</td>
<td>005207D5</td>
<td>_mbscat</td>
</tr>
<tr>
<td>001207DF</td>
<td>005207DF</td>
<td>strchr</td>
</tr>
<tr>
<td>001207E8</td>
<td>005207E8</td>
<td>strcmp</td>
</tr>
<tr>
<td>001207F1</td>
<td>005207F1</td>
<td>_mbscpy</td>
</tr>
<tr>
<td>001207FB</td>
<td>005207FB</td>
<td>strerror</td>
</tr>
<tr>
<td>00120806</td>
<td>00520806</td>
<td>strncmp</td>
</tr>
<tr>
<td>00120810</td>
<td>00520810</td>
<td>strncmp</td>
</tr>
<tr>
<td>0012081A</td>
<td>0052081A</td>
<td>strlen</td>
</tr>
<tr>
<td>00120824</td>
<td>00520824</td>
<td>strstr</td>
</tr>
<tr>
<td>0012082D</td>
<td>0052082D</td>
<td>time</td>
</tr>
<tr>
<td>00120834</td>
<td>00520834</td>
<td>toupper</td>
</tr>
<tr>
<td>0012083C</td>
<td>0052083C</td>
<td>shell32.dll</td>
</tr>
<tr>
<td>0012084A</td>
<td>0052084A</td>
<td>ShellExecuteA</td>
</tr>
<tr>
<td>00120858</td>
<td>00520858</td>
<td>USER32.dll</td>
</tr>
<tr>
<td>00120865</td>
<td>00520865</td>
<td>DispatchMessageA</td>
</tr>
<tr>
<td>00120878</td>
<td>00520878</td>
<td>ExitWindowsEx</td>
</tr>
<tr>
<td>00120888</td>
<td>00520888</td>
<td>GetMessageA</td>
</tr>
<tr>
<td>00120896</td>
<td>00520896</td>
<td>PeekMessageA</td>
</tr>
<tr>
<td>001208A3</td>
<td>005208A3</td>
<td>version.dll</td>
</tr>
<tr>
<td>001208B1</td>
<td>005208B1</td>
<td>GetFileVersionInfoA</td>
</tr>
<tr>
<td>001208C7</td>
<td>005208C7</td>
<td>VerQueryValueA</td>
</tr>
<tr>
<td>001208D6</td>
<td>005208D6</td>
<td>wininet.dll</td>
</tr>
<tr>
<td>001208E4</td>
<td>005208E4</td>
<td>InternetCloseHandle</td>
</tr>
<tr>
<td>001208FA</td>
<td>005208FA</td>
<td>InternetGetConnectedState</td>
</tr>
<tr>
<td>00120916</td>
<td>00520916</td>
<td>InternetOpenA</td>
</tr>
<tr>
<td>00120926</td>
<td>00520926</td>
<td>InternetOpenUrlA</td>
</tr>
<tr>
<td>00120939</td>
<td>00520939</td>
<td>InternetReadFile</td>
</tr>
<tr>
<td>0012094A</td>
<td>0052094A</td>
<td>ws2_32.dll</td>
</tr>
<tr>
<td>Address</td>
<td>ID Address</td>
<td>Size</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>00120957</td>
<td>00520957</td>
<td>0</td>
</tr>
<tr>
<td>00120969</td>
<td>00520969</td>
<td>0</td>
</tr>
<tr>
<td>00120976</td>
<td>00520976</td>
<td>0</td>
</tr>
<tr>
<td>00120983</td>
<td>00520983</td>
<td>0</td>
</tr>
<tr>
<td>00120992</td>
<td>00520992</td>
<td>0</td>
</tr>
<tr>
<td>0012099B</td>
<td>0052099B</td>
<td>0</td>
</tr>
<tr>
<td>001209A2</td>
<td>005209A2</td>
<td>0</td>
</tr>
<tr>
<td>001209B0</td>
<td>005209B0</td>
<td>0</td>
</tr>
<tr>
<td>001209BA</td>
<td>005209BA</td>
<td>0</td>
</tr>
<tr>
<td>001209CA</td>
<td>005209CA</td>
<td>0</td>
</tr>
<tr>
<td>001209DA</td>
<td>005209DA</td>
<td>0</td>
</tr>
<tr>
<td>001209E8</td>
<td>005209E8</td>
<td>0</td>
</tr>
<tr>
<td>001209F6</td>
<td>005209F6</td>
<td>0</td>
</tr>
<tr>
<td>001209FE</td>
<td>005209FE</td>
<td>0</td>
</tr>
<tr>
<td>00120A06</td>
<td>00520A06</td>
<td>0</td>
</tr>
<tr>
<td>00120A12</td>
<td>00520A12</td>
<td>0</td>
</tr>
<tr>
<td>00120A1E</td>
<td>00520A1E</td>
<td>0</td>
</tr>
<tr>
<td>00120A2C</td>
<td>00520A2C</td>
<td>0</td>
</tr>
<tr>
<td>00120A35</td>
<td>00520A35</td>
<td>0</td>
</tr>
<tr>
<td>00120A3D</td>
<td>00520A3D</td>
<td>0</td>
</tr>
<tr>
<td>00120A44</td>
<td>00520A44</td>
<td>0</td>
</tr>
<tr>
<td>00120A4D</td>
<td>00520A4D</td>
<td>0</td>
</tr>
<tr>
<td>00120A54</td>
<td>00520A54</td>
<td>0</td>
</tr>
<tr>
<td>00120A5D</td>
<td>00520A5D</td>
<td>0</td>
</tr>
<tr>
<td>00120A6A</td>
<td>00520A6A</td>
<td>0</td>
</tr>
<tr>
<td>00120A75</td>
<td>00520A75</td>
<td>0</td>
</tr>
</tbody>
</table>
Behavioral Analysis

Preparation for Infection

First I use md5sum to get the md5 hash of the msrll.exe malware file by running the following command:

```
c:\malware\exe>md5sum msrll.exe
84acfe96a98590813413122c12c11aaa *msrll.exe
```

The reason I did this first is so I have a baseline to compare to after executing the msrll.exe malware. If the malware modifies msrll.exe or copies itself somewhere else we will be able to verify if the file is the exact same as the original.

I then ran Regshot to get a snap shot of the registry on the clean system. Fig. 1 shows the options used to get the 1st snap shot.

![Regshot 1.61e5 Final](image)

I click 1st shot and select shot and save to get a copy of the registry while the system is clean to compare later with the infected version.

Next I open FileMon, RegMon, and TDIMon and stop them from capturing and clear the display.

Process Explorer is then opened to show the processes running on the system.

On the VMware virtual Red Hat 9 system I start the sniffer by typing in the following
command

```
snort –vd | tee /tmp/sniffer1.log
```

This command will use snort to capture network traffic to the file sniffer1.log in the tmp directory.

**Infection**

Now everything is ready to infect the VMware Windows XP SP1 system. I start capturing on FileMon, RegMon, and TDIMon. I then execute msrll.exe malware by double clicking on it. After waiting about 30 seconds I view the processes using Process Explorer and see msrll.exe running under the parent process explorer. By highlighting the process and hitting the del key I kill the malware process.

Quickly I stop capturing on FileMon, RegMon, and TDIMon. Also, I switch over to the VMware Red Hat 9 system and hit CTRL+C to end the snort capture.

I run Regshot again this time clicking on 1\textsuperscript{st} shot and selecting load. I browse to the saved registry file from the first time I ran Regshot on the clean system. Next I click on 2\textsuperscript{nd} shot and select shot and save. Now we can click on compare and see the results shown in Fig. 2.

![Fig. 2](image-url)
In examining the interesting results from the Regshot compare it looks like a service was created by the name of “Rll enhanced drive”. This service was confirmed to exist by opening the services window on the system and locating the service with that name. It was setup to start automatically on every boot up and ran with logon of local system. It was running the executable located in c:\windows\system32\mfm\msrll.exe. I verified the file existed there by navigating explorer to that location where I found msrll.exe and jtram.conf files.

I ran md5sum on msrll.exe in the c:\windows\system32\mfm location and compared the hash to the original msrll.exe file hash. The hashes matched so the file was an exact copy of the one I ran earlier. I then opened jtram.conf in UltraEdit-32. The jtram.conf file was not understandable and was probably encrypted.

I examined the FileMon log and found some interesting entries listed in Fig. 3.

Fig. 3

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>File</th>
<th>Action</th>
<th>Success Options</th>
<th>Success Error Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:51:07</td>
<td>msrll.exe:256</td>
<td>CREATE</td>
<td>C:\WINDOWS\System32\mfm</td>
<td>SUCCESS</td>
<td>Create Directory Access: All</td>
</tr>
<tr>
<td>2:51:08</td>
<td>msrll.exe:256</td>
<td>CREATE</td>
<td>C:\WINDOWS\System32\mfm\msrll.exe</td>
<td>SUCCESS</td>
<td>Overwrite If Sequential Access: All</td>
</tr>
<tr>
<td>2:51:08</td>
<td>msrll.exe:256</td>
<td>WRITE</td>
<td>C:\WINDOWS\System32\mfm\msrll.exe</td>
<td>SUCCESS</td>
<td>Offset: 0 Length: 41984</td>
</tr>
<tr>
<td>2:51:29</td>
<td>msrll.exe:952</td>
<td>CREATE</td>
<td>C:\WINDOWS\system32\mfm\jtram.conf</td>
<td>SUCCESS</td>
<td>Overwrite If Access: All</td>
</tr>
</tbody>
</table>

The FileMon log shows the creation of the directory mfm in c:\windows\system32 along with the creation of the files msrll.exe and jtram.conf.

Examination of the RegMon log files didn’t reveal much more than what Regshot did. One interesting entry found in the RegMon log is listed in Fig. 4.

Fig. 4

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>File</th>
<th>Action</th>
<th>Success Options</th>
<th>Success Error Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>659.29827310</td>
<td>msrll.exe:256</td>
<td>SetValue</td>
<td>HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed</td>
<td>SUCCESS</td>
<td>E0 7F F7 64 D4 66 A1 09 ...</td>
</tr>
</tbody>
</table>

This entry has something to do with cryptography.

Examination of TDIMon shows TCP activity being setup on the system. Entries in the TDIMon log show ports 2200 and 113 in use. TDIMon interesting log entries are displayed in Fig. 5.

Fig. 5

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>File</th>
<th>Action</th>
<th>Success Options</th>
<th>Success Error Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:51:27</td>
<td>msrll.exe:952</td>
<td>IRP_MJ_CREATE</td>
<td>TCP:0.0.0.0:2200</td>
<td>SUCCESS</td>
<td>Address Open</td>
</tr>
<tr>
<td>2:51:27</td>
<td>msrll.exe:952</td>
<td>TDI_SET_EVENT_HANDLER</td>
<td>TCP:0.0.0.0:2200</td>
<td>SUCCESS</td>
<td>Error Event</td>
</tr>
<tr>
<td>2:51:27</td>
<td>msrll.exe:952</td>
<td>TDI_SET_EVENT_HANDLER</td>
<td>TCP:0.0.0.0:2200</td>
<td>SUCCESS</td>
<td>Disconnect Event</td>
</tr>
<tr>
<td>2:51:27</td>
<td>msrll.exe:952</td>
<td>TDI_SET_EVENT_HANDLER</td>
<td>TCP:0.0.0.0:2200</td>
<td>SUCCESS</td>
<td>Receive Event</td>
</tr>
<tr>
<td>2:51:27</td>
<td>msrll.exe:952</td>
<td>TDI_SET_EVENT_HANDLER</td>
<td>TCP:0.0.0.0:2200</td>
<td>SUCCESS</td>
<td>Expedited Receive</td>
</tr>
<tr>
<td>2:51:27</td>
<td>msrll.exe:952</td>
<td>TDI_SET_EVENT_HANDLER</td>
<td>TCP:0.0.0.0:2200</td>
<td>SUCCESS</td>
<td>Chained Receive</td>
</tr>
</tbody>
</table>
I also ran AutoRuns just to check if any auto run registry entry had been added, but all looked well there.

Next I examined the snort capture file sniffer1.log. This capture file was completely empty. So it appears that no network traffic took place during the initial infection of the system.
I decided to run the service created “Rll Enhanced Drive”, but before I do I restart snort for another network capture with the command:

```
snort –vd | tee /tmp/sniffer2.log
```

I also start Process Explorer to monitor what processes are running.

Now I start the service and leave it run another 30 seconds. While I’m waiting for the 30 seconds I examine Process Explorer and see msrll.exe process running under parent service services.exe. By double clicking on the msrll.exe process in Process Explorer I get a window with various tabs to display information about the process. I select the TCP/IP tab and see that it is listening on ports 113 and 2200. After about 30 seconds pass I kill the process and switch to the VMware Red Hat 9 system and end the snort capture.

The listening ports 113 and 2200 correlates to the TDIMon log entries. This malware is listening on these ports.

Examination of the snort capture sniffer2.log shows the interesting entries displayed in Fig. 6.

```
Fig. 6
```

```
UDP TTL:128 TOS:0x0 ID:786 IpLen:20 DgmLen:66
Len: 38
00 0D 01 00 00 01 00 00 00 00 00 00 0B 63 6F 6C  .............col
6C 65 63 74 69 76 65 37 04 7A 78 79 30 03 63 6F  lective7.zxy0.co
6D 00 00 01 00 01                                m.....
```

This packet shows a DNS request coming from 192.168.62.129 port 1026 to 192.168.62.1 port 53 for the domain name collective7.zxy0.com. This DNS request is not answered however because 192.168.62.1 is the host computer that is running VMware and is not running a DNS server. 192.168.62.129 is the VMware virtual Windows XP SP1 system that is the infected system.

Since I have no DNS server I add to the hosts file on the infected system the entry listed in Fig. 7. The hosts file on the infected system is in c:\windows\system32\drivers\etc\hosts.

```
Fig. 7
```

```
192.168.62.128 collective7.zxy0.com
```
I put the address 192.168.62.128 in because I want to redirect any traffic going from the infected host 192.168.62.129 to my VMware Red Hat 9 system 192.168.62.128 so I can capture the requests the infected system is sending to collective7.zxy0.com.

I start another snort capture on the Red Hat 9 system using the following command:

```
snort –vd | tee /tmp/sniffer3.log
```

I also start Process Explorer to monitor the msrll.exe process.

I restart the “Rll Enhanced Drive” service and wait about 30 seconds. While waiting the 30 seconds I use Process Explorer to view the process msrll.exe TCP/IP activity. I observe it listening on ports 113 and 2200 like before, but during the wait I see it send a connection request to 192.168.62.128:6667 and then stop. I then see another connection request to 192.168.62.128:9999 and then stop. And another connection request to 192.168.62.128:8080 and then stop. The 30 seconds are past so I kill the msrll.exe process using Process Explorer and stop the snort capture on the Red Hat 9 system.

Examination of the snort capture file sniffer3.log I found the interesting entries listed in Fig. 8.

---

<table>
<thead>
<tr>
<th>Time</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Protocol</th>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
---

Fig. 8
Here there are connection requests from the infected system to the Red Hat 9 system on ports 6667, 9999, and 8080. The obvious port is 6667 which possibly could be an irc connection attempt. Port 9999 I’m not sure of, but listed as Distinct in the IANA list. Port 8080 is usually an alternate port for port 80 or proxy. These connection attempts listed in the snort capture correlate to the observed connection attempts in Process Explorer’s TCP/IP section for the msrll.exe process.

Since port 6667 is normally irc port I decide to run an irc server on the Red Hat 9 system that the infected system already thinks is the collective7.zxy0.com. Also the observed connection attempts to 192.168.62.128 on ports 9999 and 8080 prompt me to run netcat to record the requests on the Red Hat 9 system.

I start the irc server which is configured to listen on port 6667 by issuing the following commands on the Red Hat 9 system:

```
su – ircd
./ircd
exit
```

I then start the irc client in another session on the Red Hat 9 system by issuing the command:

```
irc
```

I also start two more sessions on the Red Hat 9 system and issue the following commands:

```
nc –l –p 9999 >/tmp/port9999.txt
nc –l –p 8080 >/tmp/port8080.txt
```

These commands start netcat listening on ports 9999 and 8080. Whatever is sent to these ports will be written to the appropriate file port9999.txt or port8080.txt in the /tmp directory. This is in the hope to capture what type of requests are being sent to these ports.

Snort capture is also started again with the following command:

```
snort –vd | tee /tmp/sniffer4.log
```

Again we start the msrll.exe malware by starting the “Rll Enhanced Drive” service and wait about 30 seconds. After 30 seconds I kill the msrll.exe process and stopped the snort capture along with the two netcat listeners on the Red Hat 9 system.

First I examine the snort capture sniffer4.log and find the interesting sections displayed
in Fig. 9.

Fig. 9

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Port</th>
<th>Sequence</th>
<th>Acknowledgment</th>
<th>Window</th>
<th>TCP Options</th>
<th>Raw Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/10-10:55:25</td>
<td>907418</td>
<td>192.168.62.129</td>
<td>192.168.62.128</td>
<td>6667</td>
<td>0x992158CC</td>
<td>0xB8F21707</td>
<td>0xFAA3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TCP TTL:128 TOS:0x0 ID:1449 IpLen:20 DgmLen:53 DF

**AP** Seq: 0xB88F210F6 Win: 0xFAA3 TcpLen: 20

4A 4F 49 4E 20 23 6D 69 6C 73 20 3A 0A JOIN #mils :

---


TCP TTL:128 TOS:0x0 ID:1444 IpLen:20 DgmLen:97 DF

**AP** Seq: 0x99215880 Ack: 0xB8F210F6 Win: 0xFAA7D TcpLen: 20

55 53 45 52 20 56 52 55 74 6F 6E 6E 49 7A 65 77 USER VRUtonnizew

46 20 6C 6F 63 61 6C 68 74 72 65 0 :M

74 65 62 70 6E 4E 49 43 4B 20 4B 4C 48 74 tebpnl.NICK KLHt

74 64 75 66 72 61 4B 44 0A tdudraKD.

---

Here it can be seen that the malware joined the channel #mils.

Next I examined the netcat capture files port9999.txt and port8080.txt and found them to be empty.

In the irc client I join the #mils channel by issuing the command:

```
/join #mils
```

This puts my client in the #mils channel.

I then run the malware again by starting the “Rll Enhanced Drive” service and wait in the irc client for the malware to join the #mils channel. After a few minutes it does join the #mils channel verifying that irc is in play with this malware.

I then try connecting to port 2200 on the infected system from the Red Hat 9 system using netcat command:

```
nc 192.168.62.129 2200
```

Netcat connects and a prompt appears #: that is awaiting my command. At this point I don’t really know what it wants so I just hit enter then CTL+C to drop the connection.

I try to connect to port 113 on the infected system from the Red Hat 9 system by issuing the following command:
nc 192.168.62.129 113

Netcat connects but no prompt just a blinking cursor. I hit enter and nothing happens so I type x and press enter. This disconnects me with the following text:

x : USERID : UNIX : GPRdvDe

At this point I am done with my behavioral analysis since I cannot invoke anymore behavior.

The summary of what is known at this point is that the malware installs itself into the c:\windows\system32\mfm directory as msrll.exe and an encrypted configuration file jtram.conf. It sets up a Windows service called “Rll Enhanced Drive” that runs at Windows startup the msrll.exe file in the mfm directory. The malware then attempts to connect to a server on the internet called collective7.zxy0.com on port 6667, 9999, and 8080. The connection to port 6667 is an irc connection and the malware joins the channel #mils. The malware also listens on ports 2200 and 113 which can be connected to using netcat. The listener on port 2200 displays a prompt #: and the port 113 doesn’t display any prompt.
Code Analysis

The code analysis starts with examining the BinText listing of the msrll.exe file. Fig. A on page 11 displays the BinText output for the msrll.exe file. Here the segment .aspack leads me to believe that the executable has been packed with aspack. The rest of the strings are not understandable which also indicates that the file is packed.

PE-Sniffer is run on the msrll.exe file, but the scans do not reveal the packing technique used. I then load the file in PEInfo which displays the section aspack further indicating that aspack was used.

I load aspackdie with the msrll.exe and a file is created called unpacked.exe and seems to have unpacked the msrll.exe. A quick examination of the strings using BinText reveals more understandable strings. It looks like it worked, but an execute test should be done. I copy the file to the c:\windows\system32\mfm directory and rename the original msrll.exe to msrll.exe.org and then rename unpacked.exe to msrll.exe. I start the service while monitoring the irc channel #mils with the irc client on the Red Hat system and sure enough the malware joins the channel after a few minutes. It looks now like I have an unpacked version of the malware that still works.

Examination of the interesting strings reveals what looks like commands that start with a question mark (?). Commands like ?si, ?jolt, ?uptime, and ?login. Some of these commands look like denial of service attacks while others seem to be giving information about the current state of the system. The command ?login leads me to believe that some sort of authentication needs to be done.

While running the malware I tried some of the commands in the irc connection to the #mils channel. Every command I typed didn’t elicit any response from the malware. I then try the following in the irc client:

```
?login
?login malware
? login a b
```

None of these generated a response from the malware. I tried these same commands using the netcat connection to port 2200 on the infected system from the Red Hat 9 system and got similar results.

The Search for Authentication Code

I’m going to cut to the chase here since there were many failed attempts at finding the authentication code. I used mainly the interesting strings found by BinText, see Fig. B on page 11, IDA Pro and OllyDbg to search the assembly instructions for possible areas where the malware is processing the authentication (?login).
I stumbled on the following string:

% bad pass from "\%s\"@\%s

This looked like a place in the code you would go if your login failed.

I loaded the unpacked version of msrll.exe into IDA Pro and performed a text search looking for the string “bad pass”. The search had two hits, one at memory address 0040BB52 and another at memory address 0040BC6F displayed in Fig. 10.

![Fig. 10](image_url)

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0040BB52</td>
<td><code>asBadPassFromS db %s bad pass from &quot;%s&quot;@%s: DATA XREF sub_40BB6B+104to</code></td>
</tr>
<tr>
<td>0040BC6F</td>
<td><code>push offset asBadPassFromS: &quot;%s bad pass from \&quot;%s\&quot;@%s&quot;</code></td>
</tr>
</tbody>
</table>

The instruction located at address 0040BC6F looked to be where bad pass would be pushed to the stack preparing to display on the screen possibly. This became the area of the code I focused on. I wanted to find how you ended up at this instruction and where the decision was made to branch to this section of the code. To find this branch I used IDA Pro to view a flowchart of the malware instructions. Clicking on view + graphs + flow chart I was able to view the code in a more visual way. I searched the flow chart manually until I found the instruction set that matched what was at address 0040BC6F. This was in a section labeled loc_40BC5A seen in Fig. 11. Going up the flowchart the instructions that decide what branch to jump to is in the section labeled 0040BBD6. In this section you can see the following instructions:

```
test   eax, eax
jz     short loc_40BC5A
```

It’s easy to see in the flowchart that if the result of the test is true we are going to jump to the section of code that contains the bad pass string which is where I’m assuming I don’t want to go if I want to get authenticated. If the result is false the code jumps to address 0040BBEB which might be where you go if you get successfully authenticated.
I switch over to the code view and search for address 004BBD6 since this is the label for the beginning of the instructions that include the test and jz instructions. I find that the test instruction is at address 0040BBE7 which is where I decide I want to set a breakpoint, but not in IDA Pro. For this job I load msrll.exe into OllyDbg.

In OllyDbg I go to address 0040BBE7 and press F2 to set a breakpoint. Next I start ircd and irc on the Red Hat 9 system. In irc I join the #mils channel. Back in OllyDbg I hit F9 to run msrll.exe malware. Then I switch back to irc and wait for the malware to join the #mils channel. After a few minutes I see the malware join the #mils channel. I want to trigger the break point with the ?login command so I try the following commands:

```
?login malware
?login a b
```

Neither of these triggered the break point. I examined the code around 0040BBE7. The beginning of this section started at address 0040BBD6 which you can see in Fig. 11 above. Following the code I couldn't find any hints as to what the problem was.
I decided to connect using netcat on the Red Hat 9 system to port 2200 on the infected system. I issued the following command:

```
nc 192.168.62.129 2200
```

Netcat connected and I was prompted with a #:_. Next I issued the following commands:

```
?login malware <enter>
<enter>
<enter>
x <enter>
```

At the point I typed (x <enter>) I triggered the breakpoint in OllyDbg. Now I knew I was sitting at the TEST EAX, EAX instruction. From the flowchart research I knew I needed the result to be false. The EAX register contained 00000000 as its value which means the next instruction, JE SHORT msrll.0040BC5A, would be taken. This would be a true condition. So I right clicked on the EAX register and selected increment changing the EAX register’s value to 00000001. This would now result in a false condition. I pressed F9 and continued running the malware.

Back on the Red Hat 9 system all I had was a cursor, but I was still connected. I typed in the following command:

```
?uptime <enter>
```

This returned the following line:

```
sys: 01h 49m 04s bot; 33m 09s
```

I was now authenticated, but just to make sure I issued the following command:

```
?status <enter>
```

This also returned information. It looks like it worked. I want to try all the commands to see what happens, but before I do that I’m going to patch the malware to bypass the authentication.

The instruction (JE SHORT, msrll.0040BC5A) needs to be replaced with a NOP so that this jump cannot happen since taking the jump is a true condition and I want a false condition to get authenticated.

To patch the malware I clicked on the JE SHORT, msrll.0040BC5A instruction in OllyDbg and hit the space bar. This opened an assembler window with an entry box that I typed in NOP. I made sure the “Fill with NOP’s” box was checked in order to fill the
replaced instruction space properly. I then click assemble and cancel. This replaced the JE instruction with two NOP instructions. To save this patched version of the malware I right clicked on the assembler pane, select “Copy to executable”, select “All modifications”, clicked “Copy all”, and then a new disassembler pane opened. I right click on the pane and select “Save file”. This prompted me for the file name to save as. I used the name msrll-patched.exe and saved it.

Now I had a patched version of the program. I saved this version in place of the current msrll.exe in the c:\windows\system32\mfm directory.

**Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>?insmod</td>
<td>Install loadable modules</td>
<td>?insmod: &lt;mod name&gt;</td>
</tr>
<tr>
<td>?rmmod</td>
<td>Remove loadable modules</td>
<td>?rmmod: &lt;mod name&gt;</td>
</tr>
<tr>
<td>?lsmod</td>
<td>List loadable modules</td>
<td>(No response)</td>
</tr>
<tr>
<td>?ping</td>
<td>ping DoS attack</td>
<td>?ping &lt;ip&gt; &lt;total secs&gt; &lt;p size&gt; &lt;delay&gt; [port]</td>
</tr>
<tr>
<td>?udp</td>
<td>udp DoS attack</td>
<td>?udp &lt;ip&gt; &lt;total secs&gt; &lt;p size&gt; &lt;delay&gt; [port]</td>
</tr>
<tr>
<td>?syn</td>
<td>syn DoS attack</td>
<td>?syn &lt;ip&gt; &lt;port&gt; &lt;t_time&gt; &lt;delay&gt;</td>
</tr>
<tr>
<td>?smurf</td>
<td>smurf DoS attack</td>
<td>?smurf &lt;ip&gt; &lt;p size&gt; &lt;duration&gt; &lt;delay&gt;</td>
</tr>
<tr>
<td>?jolt</td>
<td>jolt DoS attack</td>
<td>?jolt &lt;ip&gt; &lt;duration&gt; &lt;delay&gt;</td>
</tr>
<tr>
<td>?si</td>
<td>Displays system information</td>
<td>WINXP (u:James) mem:(52/127) 58% Genuine Intel (null)</td>
</tr>
<tr>
<td>?ssl</td>
<td>Something to do with ssl</td>
<td>?ssl: -1</td>
</tr>
<tr>
<td>?login</td>
<td>Login command must be</td>
<td>(No response on unsuccessful login)</td>
</tr>
<tr>
<td>login</td>
<td><code>?</code></td>
<td>login &lt;enter&gt; user &lt;enter&gt; password &lt;enter&gt;</td>
</tr>
<tr>
<td>?uptime</td>
<td>Displays system and bot uptime stats</td>
<td>sys: 58m 16s bot: 27m 14s</td>
</tr>
<tr>
<td>?reboot</td>
<td>Reboots the system</td>
<td>later!</td>
</tr>
<tr>
<td>?status</td>
<td>Displays status info</td>
<td>service: N user: James inet connection: Y contype: Lan reboot privs: Y</td>
</tr>
<tr>
<td>?jump</td>
<td><code>?</code></td>
<td>(No response)</td>
</tr>
<tr>
<td>?nick</td>
<td>Change nick on irc channel that bot is connected to, but you must first select the irc sock to perform the command on</td>
<td>Set an irc sock to perform ?nick command on Type .sklist to view current sockets, then .dccsk &lt;#&gt;</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>?echo</td>
<td>echos what you type after ?echo to the screen - if you typed ?echo hello &lt;enter&gt; the response would be what is in the response column to the right</td>
<td>hello</td>
</tr>
<tr>
<td>?hush</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>?wget</td>
<td>When you type ?wget &lt;enter&gt; there is no response, but if you type ?wget x &lt;enter&gt; the response will be what is in the response column to the right</td>
<td>no file name in x</td>
</tr>
<tr>
<td>?join</td>
<td>irc join command to join other channels</td>
<td>(No output to screen - need to us the ?sklist and ?dccsk commands to find and connect to an irc sock first)</td>
</tr>
<tr>
<td>?op</td>
<td>irc command to become channel operator</td>
<td>?op bad args</td>
</tr>
<tr>
<td>?aop</td>
<td>Not sure what this does, but you can add or remove a host by the ?aop + &lt;host&gt; or ?aop - &lt;host&gt;</td>
<td>usage: ?aop +/- &lt;host&gt;</td>
</tr>
<tr>
<td>?akick</td>
<td>Not sure what this one does either, but has similar syntax as ?aop</td>
<td>usage: ?akick +/- &lt;host&gt;</td>
</tr>
<tr>
<td>?part</td>
<td>irc command to leave a channel</td>
<td>(No output to screen)</td>
</tr>
<tr>
<td>?dump</td>
<td>?</td>
<td>(No response)</td>
</tr>
<tr>
<td>?set</td>
<td>shows the jtram.conf contents plus can change settings by issuing the ?set &lt;setting&gt; &lt;value&gt;</td>
<td>set jtr.bin msrll.exe set jtr.home mfm set bot.port 2200 set jtr.id run5 set servers collective7.zxy0.com,collective7.zxy0.com:9999!,collective7.zxy0.com:8080 set irc.quit set dcc.pass $1$KZLPLKDF$W8kl8Ju1X8DOHZsmlp9qq0</td>
</tr>
<tr>
<td>?die</td>
<td>Kills msrll.exe process</td>
<td>(No output to screen)</td>
</tr>
<tr>
<td>?md5p</td>
<td>Displays the salt and md5 hash of whatever is typed in as &lt;pass&gt; parameter</td>
<td>?md5p &lt;pass&gt; &lt;salt&gt;</td>
</tr>
<tr>
<td>?free</td>
<td>?</td>
<td>usage: ?free &lt;cmd&gt;</td>
</tr>
<tr>
<td>?raw</td>
<td>?</td>
<td>(No output to screen)</td>
</tr>
<tr>
<td>?update</td>
<td>Possibily a command to update bot</td>
<td>?update &lt;url&gt; &lt;id&gt;</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Output</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td><code>?hostname</code></td>
<td>Displays hostname and ip address</td>
<td>host: xxx.localdomain ip: 192.168.62.129</td>
</tr>
<tr>
<td><code>?fif</code></td>
<td>Delete a file</td>
<td>(No response)</td>
</tr>
<tr>
<td><code>?mdif</code></td>
<td>Display a directory</td>
<td>(No response)</td>
</tr>
<tr>
<td><code>?pwd</code></td>
<td>Display the current directory</td>
<td>c:\windows\system32\mfm</td>
</tr>
<tr>
<td><code>?play</code></td>
<td>Copy a file</td>
<td>(null): somefile</td>
</tr>
<tr>
<td><code>?move</code></td>
<td>Move a file</td>
<td></td>
</tr>
<tr>
<td><code>?dir</code></td>
<td>Display directory</td>
<td>(Displays directory listing of current directory)</td>
</tr>
<tr>
<td><code>?sums</code></td>
<td>Display the md5 hashes for all files in the current directory</td>
<td>(Displays a file listing with it's md5 hash value next to it)</td>
</tr>
<tr>
<td><code>?ls</code></td>
<td>Displays directory listing of current directory</td>
<td>(Displays directory listing of current directory)</td>
</tr>
<tr>
<td><code>?cd</code></td>
<td>Changes directory</td>
<td></td>
</tr>
<tr>
<td><code>?rmkdir</code></td>
<td>Removes a directory</td>
<td></td>
</tr>
<tr>
<td><code>?mkdir</code></td>
<td>Makes a new directory</td>
<td></td>
</tr>
<tr>
<td><code>?run</code></td>
<td>Run a program (hidden) syntax: ?run c:\windows\system32\notepad.exe &lt;enter&gt;</td>
<td>(Example: ?run c:\windows\system32\notepad.exe) ?run: ran ok (4022304)</td>
</tr>
<tr>
<td><code>?exec</code></td>
<td>irc command exec</td>
<td>(No response)</td>
</tr>
<tr>
<td><code>?ps</code></td>
<td>Display all processes running and their PID's</td>
<td>(lists active processes running on infected system)</td>
</tr>
<tr>
<td><code>?kill</code></td>
<td>Kill a process - ?kill 1448 &lt;enter&gt;</td>
<td>pid 1448 killed</td>
</tr>
<tr>
<td><code>?killall</code></td>
<td>?</td>
<td>(No response)</td>
</tr>
<tr>
<td><code>?crash</code></td>
<td>Crashes system</td>
<td>(No output to screen)</td>
</tr>
<tr>
<td><code>?dcc</code></td>
<td>irc command dcc direct connections to remote clients</td>
<td>(No response)</td>
</tr>
<tr>
<td><code>?get</code></td>
<td>?</td>
<td>(No response)</td>
</tr>
<tr>
<td><code>?say</code></td>
<td>irc say command - I believe this message would be said non-privately</td>
<td>usage: ?say &lt;target&gt; &quot;text&quot;</td>
</tr>
<tr>
<td><code>?msg</code></td>
<td>irc command to send private message to nick or list of nicks</td>
<td>usage: ?msg &lt;target&gt; &quot;text&quot;</td>
</tr>
<tr>
<td><code>?kb</code></td>
<td>?</td>
<td>?kb &lt;nick&gt; &lt;chan&gt;</td>
</tr>
<tr>
<td><code>?sklist</code></td>
<td>Display current socks</td>
<td>(A display numbering the different socks and connection information like ip address, nick, and irc chan)</td>
</tr>
<tr>
<td><code>?unset</code></td>
<td>Unsets a set command refer to ?set command</td>
<td>(Example: ?unset pass &lt;enter&gt;) (This will remove the set pass parameter)</td>
</tr>
<tr>
<td><code>?uattr</code></td>
<td>?</td>
<td>usage: ?uattr &lt;nick&gt; &lt;chan&gt;</td>
</tr>
<tr>
<td><code>?dccsk</code></td>
<td>Set the irc sock to use</td>
<td>usage: ?dccsk &lt;socks #&gt;</td>
</tr>
</tbody>
</table>
### Analysis Wrap-Up

Once the malware is executed on a system, it will copy itself to the c:\%systemroot%\system32\mfm directory. It will then create jtram.conf file in the same directory. The jtram.conf file contains the encrypted configuration settings of the bot. The malware sets up a service, Rll enhanced drive, which starts automatically when the system is booted and runs with local system authority. The bot then attempts to connect to the irc server collective7.zxy0.com first on port 6667 then port 9999 and then port 8080. Once the bot is connected to the irc server it joins the #mils channel with a randomly generated nick. At this point the bot is awaiting orders from the bot commander/creator.

Analysis shows this bot is capable of receiving a connection on port 2200 using telnet or netcat. Connecting to this port presents a prompt #:_ awaiting authentication using the ?login command. Authentication allows you to execute numerous commands. These commands can setup denial of service attacks, run programs hidden to the user, update the bot, send irc related commands to the irc server from the bot system, get information on the infected system resources and configuration, kill processes, transfer files and change the bot configuration.

This bot army might have been created to sell or trade for something in return, to attack a specific website, to speed the spread of a future virus or worm, to steal financial information, to harvest email address, to spam, and the list goes on. This bot appears to have upgradeability build in so its purpose could change.

The first defensive tactic is to use a firewall to block outgoing traffic on port 6667, 9999, and 8080. If other outgoing ports are not being used they should be blocked as well to prevent the bot from reporting in. Next block incoming traffic to port 2200 as well as any other ports that are not required. Now you’re left with the existing infected systems to detect and clean. To find the infected systems first run antivirus software, but if that doesn’t detect it then you could run a port scanner on the network like nmap and look for systems listening on port 2200 or other odd ports. You could manually go to each one of the suspect systems and kill the msrll.exe process and remove the c:\%systemroot%\system32\mfm directory and files from the system. You’d also have the rll enhanced drive service to deal with by at least setting it to manual startup instead of automatic. You could also script the removal of the files since nothing should live in the mfm directory. The following batch file presents as an example. It would not matter if this ran on a system that wasn’t infected; it would just not delete...
anything.

```batch
@echo off
cls
echo Ready to delete mfm directory...
pause
c:
del /Q c:\windows\system32\mfm\*.*
del /Q c:\winnt\system32\mfm\*.*
attrib –r c:\windows\system32\mfm
attrib –r c:\winnt\system32\mfm
rd c:\windows\system32\mfm
rd c:\winnt\system32\mfm
```

Additional things you could do to prevent future attacks is to install a personal firewall on each system that can detect when unauthorized applications try to communicate on the network or Internet. Keep the antivirus software up to date. Monitor on regular bases the listening ports on each system. Create Snort IDS signatures to detect this activity.
List of Resources

Software Tools

vmware Workstation Product Web Site. 8 Dec. 2004

Symantec Ghost Product Web Site. 8 Dec. 2004
<http://sea.symantec.com/content/product.cfm?productid=9>

Free Software Foundation, Inc. md5sum Web Site. 8 Dec 2004
<http://www.gnu.org/software/textutils/textutils.html>

Foundstone, Inc. BinText Web Site. 8 Dec. 2004
<http://www.foundstone.com/proddesc/bintext.htm>

TiANWEi. Regshot Download Web Site. 8 Dec. 2004
<http://www.pcworld.com/downloads/file_description/0,fid,19540,00.asp>

Russinovich, Mark and Cogswell, Bryce. Filemon Web Site. 8 Dec. 2004
<http://www.sysinternals.com/ntw2k/source/filemon.shtml>

Russinovich, Mark and Cogswell, Bryce. Regmon Web Site. 8 Dec. 2004
<http://www.sysinternals.com/ntw2k/source/regmon.shtml>

Russinovich, Mark. TDIMon Web Site. 8 Dec. 2004
<http://www.sysinternals.com/ntw2k/freeware/tdimon.shtml>

Russinovich, Mark. Process Explorer Web Site. 8 Dec. 2004
<http://www.sysinternals.com/ntw2k/freeware/procexp.shtml>

Russinovich, Mark and Cogswell, Bryce. Autoruns Web Site. 8 Dec. 2004
<http://www.sysinternals.com/ntw2k/freeware/autoruns.shtml>

UltraEdit Web Site. 8 Dec. 2004
<http://www.ultraedit.com/index.php?name=Content&pa=showpage&pid=10>


y0da. ASPACKDIE Web Site. 8 Dec. 2004 <http://scifi.pages.at/yoda9k/proggies.htm>


Yuschuk, Oleh. OllyDbg Web Site. 8 Dec. 2004 <http://home.t-online.de/home/ollydbg/>


# Upcoming SANS Forensics Training

<table>
<thead>
<tr>
<th>Training Event</th>
<th>Location and Time</th>
<th>Start Date - End Date</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Gulf Region Spring 2021</td>
<td>Virtual - Gulf Standard Time, United Arab Emirates</td>
<td>Mar 13, 2021 - Mar 18, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Cyber Security West: March 2021</td>
<td></td>
<td>Mar 15, 2021 - Mar 20, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Riyadh March 2021</td>
<td>Virtual - Gulf Standard Time, Kingdom Of Saudi Arabia</td>
<td>Mar 20, 2021 - Apr 01, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS 2021</td>
<td></td>
<td>Mar 22, 2021 - Mar 27, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Secure Australia 2021</td>
<td>Canberra, Australia</td>
<td>Mar 22, 2021 - Mar 27, 2021</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Secure Australia 2021 Live Online</td>
<td>Virtual - Australian Eastern Daylight Time, Australia</td>
<td>Mar 22, 2021 - Mar 27, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Munich March 2021</td>
<td>Virtual - Central European Time, Germany</td>
<td>Mar 22, 2021 - Mar 27, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>USMC-MARFORCYBER / UKI - Live Online (FOR508)</td>
<td>Columbia, MD</td>
<td>Apr 05, 2021 - Apr 10, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Cyber Security Mountain: April 2021</td>
<td></td>
<td>Apr 05, 2021 - Apr 10, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>MPO/W Columbia CMIP FY21 (FOR508) 5-day format</td>
<td>Columbia, MD</td>
<td>Apr 05, 2021 - Apr 09, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Autumn Australia 2021 - Live Online</td>
<td>Virtual - Australian Eastern Standard Time, Australia</td>
<td>Apr 12, 2021 - Apr 17, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS London April 2021</td>
<td>Virtual - British Summer Time, United Kingdom</td>
<td>Apr 12, 2021 - Apr 17, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Fort Gordon Cyber Protection Brigade (CPB/ARCYBER) (FOR508)</td>
<td>Augusta, GA</td>
<td>Apr 12, 2021 - Apr 17, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Cyber Security East: April 2021</td>
<td></td>
<td>Apr 12, 2021 - Apr 17, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Autumn Australia 2021</td>
<td>Sydney, Australia</td>
<td>Apr 12, 2021 - Apr 17, 2021</td>
<td>Live Event</td>
</tr>
<tr>
<td>USMC-MARFORCYBER / UKI - Live Online (FOR578)</td>
<td>Columbia, MD</td>
<td>Apr 12, 2021 - Apr 17, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Hurlburt DCO/H 21-04</td>
<td>Hurlburt Field, FL</td>
<td>Apr 19, 2021 - Apr 24, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Secure India 2021</td>
<td>Virtual - India Standard Time, India</td>
<td>Apr 19, 2021 - Apr 24, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Stay Sharp: DFIR 2021</td>
<td></td>
<td>Apr 19, 2021 - Apr 21, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Rocky Mountain Spring: Virtual Edition 2021</td>
<td></td>
<td>Apr 26, 2021 - May 01, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Baltimore Spring: Virtual Edition 2021</td>
<td></td>
<td>Apr 26, 2021 - May 01, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Brussels April 2021</td>
<td>Virtual - Central European Summer Time, Belgium</td>
<td>Apr 26, 2021 - May 01, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS DFIRCON Spring 2021</td>
<td></td>
<td>May 03, 2021 - May 08, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>Fort Gordon Cyber Protection Brigade (CPB/ARCYBER)</td>
<td>Augusta, GA</td>
<td>May 03, 2021 - May 08, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Security West 2021</td>
<td></td>
<td>May 10, 2021 - May 15, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Amsterdam May 2021</td>
<td>Virtual - Central European Summer Time, Netherlands</td>
<td>May 17, 2021 - May 22, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Stockholm May 2021</td>
<td>Virtual - Central European Summer Time, Sweden</td>
<td>May 31, 2021 - Jun 05, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Cyber Security Central: June 2021</td>
<td></td>
<td>Jun 07, 2021 - Jun 12, 2021</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS FOR500 In Italian June 2021</td>
<td></td>
<td>Jun 07, 2021 - Jun 12, 2021</td>
<td>CyberCon</td>
</tr>
</tbody>
</table>