



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

## Interested in learning more?

Check out the list of upcoming events offering  
"Reverse-Engineering Malware: Malware Analysis Tools and Techniques (Forensics)"  
at <http://www.giac.org/registration/grem>

# **Malicious Code Analysis: The msrll.exe Case**

Reverse Engineering Malware  
(GREM)

Practical Assignment

Version 1.0

Yves Lafrance  
ILOT XII

December 2, 2004

© SANS Institute 2000 - 2005, Author retains full rights

## Table of Contents

<a href="#"><u>Abstract</u></a>	1
<a href="#"><u>Document Conventions</u></a>	1
<a href="#"><u>Introduction</u></a>	2
<a href="#"><u>Laboratory Setup</u></a>	3
<a href="#"><u>Hardware Settings</u></a>	3
<a href="#"><u>Network Settings</u></a>	3
<a href="#"><u>Host Computer</u></a>	4
<a href="#"><u>'Target 1' Computer</u></a>	5
<a href="#"><u>'Target 2' Computer</u></a>	5
<a href="#"><u>'Monitor / Utility' Computer</u></a>	5
<a href="#"><u>Software</u></a>	6
<a href="#"><u>Host Computer</u></a>	6
<a href="#"><u>Target 1 Computer</u></a>	6
<a href="#"><u>Target 2 Computer</u></a>	7
<a href="#"><u>'Monitor / Utility' Computer</u></a>	7
<a href="#"><u>Properties of the Malware Specimen</u></a>	9
<a href="#"><u>File Information</u></a>	9
<a href="#"><u>Embedded Strings</u></a>	9
<a href="#"><u>Behavioral Analysis</u></a>	11
<a href="#"><u>Msrll.exe 'first run'</u></a>	11
<a href="#"><u>File Monitoring</u></a>	11
<a href="#"><u>Registry Monitoring</u></a>	11
<a href="#"><u>Cryptography</u></a>	12
<a href="#"><u>Confirming Observations</u></a>	12
<a href="#"><u>Behavior Once Installed</u></a>	13
<a href="#"><u>Agent Behavior</u></a>	14
<a href="#"><u>'msrll.exe' Startup</u></a>	14
<a href="#"><u>'msrll.exe' Auxiliary Files</u></a>	14
<a href="#"><u>Environment Information</u></a>	15
<a href="#"><u>Network Activities</u></a>	15
<a href="#"><u>Alternate Ports (8080 and 9999)</u></a>	15
<a href="#"><u>Successful IRC Connection</u></a>	16
<a href="#"><u>Port 2200</u></a>	17
<a href="#"><u>Agent Commands</u></a>	17
<a href="#"><u>Code Analysis</u></a>	19
<a href="#"><u>Unpacking 'msrll.exe'</u></a>	19
<a href="#"><u>"Authentication Bypass"</u></a>	19
<a href="#"><u>Finding Commands</u></a>	20
<a href="#"><u>Finding command routines</u></a>	21
<a href="#"><u>Analysis Wrap-up</u></a>	22
<a href="#"><u>Program Capabilities</u></a>	22
<a href="#"><u>Protect against it!</u></a>	22
<a href="#"><u>Detect it!</u></a>	22
<a href="#"><u>Contain it!</u></a>	22
<a href="#"><u>Eradicate it!</u></a>	22

<a href="#">Appendices</a>	24
<a href="#">Appendix A – Checklist Examples</a>	24
<a href="#">Malicious Code Loading</a>	24
<a href="#">Transferring Results Files</a>	24
<a href="#">Appendix B - Embedded Strings</a>	25
<a href="#">Appendix C - Regshot Comparison – User Account</a>	33
<a href="#">Appendix D - Agent Control Commands</a>	36
<a href="#">Appendix E - Connection Sequences</a>	41
<a href="#">References</a>	42

## List of Figures

<a href="#">Figure 1 – Laboratory Setup for Malicious Code Analysis</a>	3
<a href="#">Figure 2 – msrll.exe Behavior</a>	13

## List of Tables

<a href="#">Table 1 – Msrll.exe - Program Sections</a>	9
<a href="#">Table 2 – File Monitoring</a>	11
<a href="#">Table 3 – Registry Monitoring</a>	12
<a href="#">Table 4 - Cryptography</a>	12
<a href="#">Table 5 – RegShot - Files</a>	13
<a href="#">Table 6 – Folder Creation – Name Collision</a>	14
<a href="#">Table 7 – Auxiliary Files</a>	15
<a href="#">Table 8 - DNS Request</a>	15
<a href="#">Table 9 – Port 8080 Activity</a>	16
<a href="#">Table 10 - IRC Connection</a>	17
<a href="#">Table 11 - AsPack Section Name</a>	19
<a href="#">Table 12 - Authentication Error Messages</a>	19
<a href="#">Table 13 - Authentication Bypass</a>	20
<a href="#">Table 14 - Command Names &amp; Command Subroutines Call</a>	21

## Abstract

---

This document presents observations made about one piece of malicious code. It follows the investigating method taught in the 'Reverse Engineering Malware' course.

The first section presents the setup used to investigate. This is followed by the first observations regarding the code to analyze. The next two sections describe the findings about the program's behavior and code analysis. Through these sections, readers may see how pieces of information influence the analysis process.

The document concludes with a wrap-up of the knowledge gained during the analysis.

Appendices contain detailed information gathered about 'msrll.exe'.

## Document Conventions

---

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented in this way include the following:

<code>command</code>	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
<code>filename</code>	Filenames, paths, and directory names are represented in this style.
<code>computer output</code>	The results of a command and other computer output are in this style
<code>URL</code>	Web URL's are shown in this style.
<i>Quotation</i>	A citation or quotation from a book or web site is in this style.

## Introduction

---

Examining malicious code is a fascinating experience. Like many specialized tasks, it requires both the use of tools and knowledge. But it also requires some understanding of people. Analyzing someone else's program leads to trying to understand their motives and way of thinking.

The first part of this document describes the laboratory used to pursue the analysis. This includes the hardware and the software used as well as the network settings. The environment being set, the next sections describe one specific malicious code analysis.

The subsequent sections describe observations made during the experiments conducted to understand this piece of code. These observations were gathered using two methods. The first one consists in deploying tools to observe the interaction of the malicious code specimen, both inside and outside an infected computer. The goal of the second method is to gain access to the code of the analyzed program and to understand it (at least parts of it).

Even though these sections are presented sequentially, the process of understanding malicious code requires combining behavioral and code analyses together; the result of one approach gives pertinent clues to use with the other one. It is possible to go deeper in the analysis combining the knowledge gained from the two methods.

It is important to remember why such an analysis is performed. The overall goals are to contribute to the incident handling process. Malicious code analysis can be very handy especially for the 'containment' and 'eradication' phases. To be cost effective, analyses have to stop when the analyst has reasonable grounds to believe that the process has delivered the required information.

© SANS Institute 2000 - 2005

## Laboratory Setup

The laboratory used to perform this malicious code analysis is inspired from the setup proposed in the "Reverse Engineering Malware" course. It is based on virtual computers running on a single "real" one.

### Hardware Settings

The computer supporting the laboratory environment is configured with a 1.8 Ghz Pentium 4M CPU and 786 Mbytes of memory. It runs under a Windows XP-SP2 operating system. The current document refers to this computer as the 'host computer'.

VMware is used to provide the virtual computer environment. It is described in the 'hardware section' because it simulates hardware on which operating systems run as well as providing the network environments. Settings are adjusted to implement maximum isolation between the real computer and the virtual ones. These include deactivating the 'cut and paste' capabilities between the host computer and the other ones.

### Network Settings

The figure below illustrates how the host computer is configured to create the environment used to securely study malicious programs (malware). The objective of this configuration is to provide the connectivity required to perform analyses as well as maintaining isolation to make sure that malicious code does not "escape" from the laboratory environment.

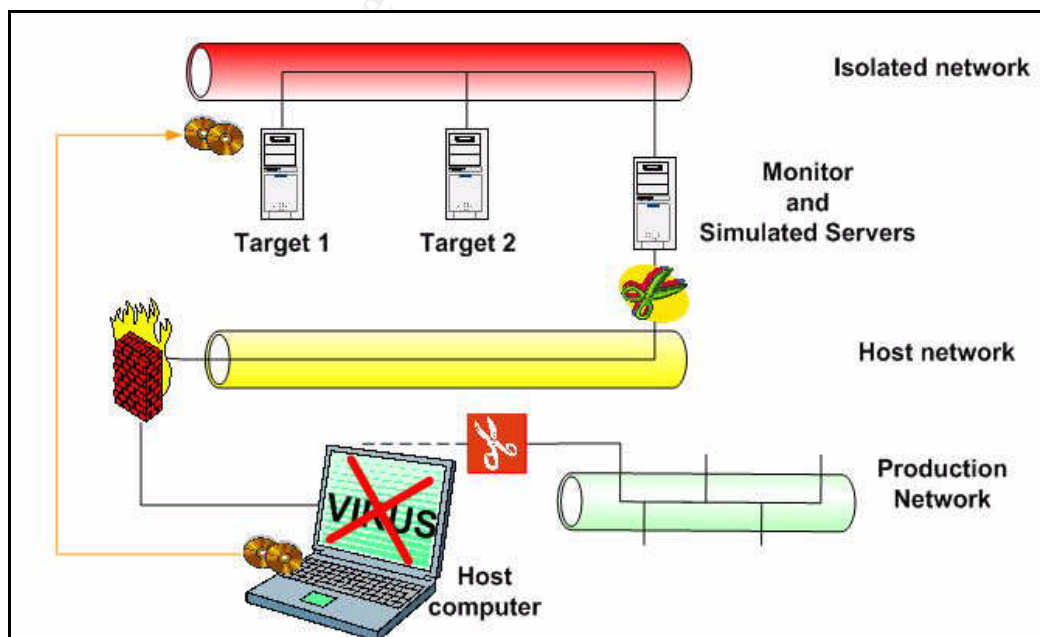





Figure 1 – Laboratory Setup for Malicious Code Analysis



Keeping risk mitigation in mind, the laboratory is designed with three different networks. A fourth network is 'simulated' using another VMware function.

- Production network (green)**
- ❑ This network is used only to transfer files with the 'host computer'.
  - ❑ The network cable is physically disconnected when working with malicious code. 
- Host network (yellow)**
- ❑ This network, provided by the VMware software, is used to supply connectivity between the host computer and the virtual ones.
  - ❑ This network is only used to share information between the isolated network and the host computer (mostly to retrieve files resulting from the analysis).
  - ❑ Computers targeted to be 'infected' by the malicious code are not connected to this network.
  - ❑ An FTP server is used to perform the file transfer directed to the host computer.
  - ❑ The network interface on the host computer is only activated (using the virtual machine settings) when required. 
- Isolated network (red)**
- ❑ The isolated network is the only network where malicious traffic is allowed to happen.
  - ❑ The host computer is not linked to this network, to reduce contamination risks. The FTP server also uses this network to relay files from the 'target computers' to the host computer.
  - ❑ Once again, this server is only activated when a file transfer is required.
- ISO disk images (orange)**
- 
- ❑ VMware provides a function to feed ISO image files to a virtual computer to reproduce a CD-ROM disk. This function is used to 'feed' the first target computer with the malicious code to analyze.
  - ❑ This provides a network with an independent (and read-only) way to communicate from the host computer to the first target computer.
  - ❑ This method results in a 'one-way' communication to protect the host computer from malicious infection.

Now that the communications paths have been configured, each computer has distinct functions:

## Host Computer

---

The host computer controls the environment of the virtual computers that are used to perform the malicious code analysis. It supplies the virtual networks and

resources to run each virtual computer.

The primary concern about the host computer is its protection. This computer must be 'sealed' to guarantee that malicious code won't leak into the production environment. Several methods are used to enforce the safety measures.

- ❑ **Up-to-date operating system maintenance** to reduce the vulnerability level of the host computer.
- ❑ **Up-to-date Anti Virus software** to detect any known malicious code that could be stored or active on the host computer.
- ❑ **Firewall** tightly configured on each network interface (virtual and real)
- ❑ **ISO images** to provide the malicious code to the first target computer in a one-way communication
- ❑ **Procedures and Checklists** to make sure that all safety measures are enforced correctly to protect the host computer. Checklists are reminders to protect the integrity of the laboratory environment. Appendix A presents examples of checklists used every time a file exchange is made to or from the host computer with the virtual environment.

### **'Target 1' Computer**

---

The first target computer is used to:

- ❑ Check the properties of the malicious specimen
- ❑ Perform the behavioral analysis of the specimen
- ❑ Perform the code analysis
- ❑ Store the file of each observation

The computer was configured with Windows 2000 Professional - Service Pack 4. Other software applications used on this target computer are described in the software section. This computer may also be called 'target', 'infected computer' 'bot' or 'zombie computer'. A zombie is a computer acting under the control of another one operated by a person with malicious intentions.

### **'Target 2' Computer**

---

The general role of the second target computer is to be infected by the first one over the isolated network. It operates under Windows 2000 Professional - Service Pack 4 with no additional protection. In the present analysis, it is used as a second member of a 'zombie army'.

### **'Monitor / Utility' Computer**

---

This computer, running under Redhat Linux version 9, supplies commodities to perform the malicious code analysis.

It is used to:

- ❑ Simulate servers if required by the code infecting the target computers
- ❑ Support tools to monitor the malicious code behavior over the 'isolated network'
- ❑ As a base for the analyst to interact over the network with the malicious code running on targets to 'stimulate' malicious code and observe its reactions
- ❑ To transfer files containing the observations to the host computer where analyses and report writing are performed.

## Software

---

Software applications used to perform the analysis are presented grouped under the computers in which they are installed.

### Host Computer

---

Vmware v 4.5.2 As described in the hardware section, this software simulates the hardware and network environments to perform tasks on 'virtual computers and networks'. URL: <http://www.vmware.com/>

Notepad Text editor provided with Microsoft Windows XP.  
URL: <http://www.microsoft.com/>

Excel Spreadsheet used to browse logs in a convenient way.  
URL: <http://www.microsoft.com/>

MagicISO ISO image editor to create ISO images to be read by the target computers. MagicISO is a commercial product. A function restricted trial version is available. URL: <http://www.magiciso.com/index.htm>

### 'Target 1' Computer

---

Windows 2000Pro Partially patched Operating system to operate the computer. Patch level is Service Pack 4. URL: <http://www.microsoft.com/>

WinZip Used to read compressed files and extract their content.  
URL: <http://www.winzip.com/>

MD5sum Computes a unique hash based on the content of a file. This hash provides a sure way to determine if a file is an exact copy of another one.  
Windows version URL: <http://www.etree.org/md5com.html>

BinText Used to extract text strings from an executable file.  
URL: <http://www.foundstone.com/>

PEinfo	Used to view PE headers and embedded strings from an executable program. This program, created by Tom Liston, was installed from the 'Reverse Engineering Malware course' CD.
RegShot	A tool to compare to 'states' of a computer. This is used to look at modifications performed on a computer by a malicious program. URL: <a href="http://regshot.ist.md/">http://regshot.ist.md/</a>
RegMon	Logs actions made on a computer regarding the Registry. URL: <a href="http://www.sysinternals.com/ntw2k/source/regmon.shtml">http://www.sysinternals.com/ntw2k/source/regmon.shtml</a>
FileMon	Logs actions made on a computer regarding the file system. URL: <a href="http://www.sysinternals.com/ntw2k/source/filemon.shtml">http://www.sysinternals.com/ntw2k/source/filemon.shtml</a>
TDIMon	Logs network actions made on a computer regarding the network (TCP and UDP) activities. URL: <a href="http://www.sysinternals.com/ntw2k/freeware/tdimon.shtml">http://www.sysinternals.com/ntw2k/freeware/tdimon.shtml</a>
Process Explorer	Tracking program to gather information about processes running on a computer. Can also be used to kill processes that may be difficult to stop with task manager. URL: <a href="http://www.sysinternals.com/ntw2k/freeware/procexp.shtml">http://www.sysinternals.com/ntw2k/freeware/procexp.shtml</a>
IDA pro	Freeware version of the IDA pro Disassembler. Used to disassemble the malicious code to look at it in a more comprehensive way. URL: <a href="http://www.datarescue.be/downloadfreeware.htm">http://www.datarescue.be/downloadfreeware.htm</a>
OllyDebug	Free debugger including a disassembler. Used to trace a program with the objective of understanding its behavior. URL: <a href="http://home.t-online.de/home/Ollydbg/">http://home.t-online.de/home/Ollydbg/</a>
AsPackDie	Unpacking tool to unpack files packed using 'AsPack' method. URL: <a href="http://protools.anticrack.de/unpackers.htm">http://protools.anticrack.de/unpackers.htm</a>

## **'Target 2' Computer**

---

Software applications used on this computer are identical to those used on 'Target 1' computer.

## **'Monitor / Utility' Computer**

---

RedHat Linux v. 9 The Linux Operating system platform is used to provide a different operating environment from the target computers. The difference in technology helps to reduce malicious code propagation risks. The Linux virtual machine image supplied on the course CD was used for the laboratory elaboration. URL: <http://www.redhat.com>

NetCat Network utility used to read and write across network connections. Used to simulate software listening on specific network ports. URL: <http://www.atstake.com/research/tools/>

Snort Open source network intrusion detection system. Used in the laboratory environment to capture network traffic. URL: <http://www.snort.org/>

IRCd IRC server used to interact with malicious code infecting target computers. URL: <http://ircd-hybrid.com/>

Vsftpd FTP server used to transfer files from the 'target computers' and Linux computer to the host computer. Installed from the RedHat CD image disks. URL: <http://vsftpd.beasts.org/>

---

© SANS Institute 2000 - 2005

## Properties of the Malware Specimen

When harvesting files to perform malware analysis, it is important to gather as much information as possible. Information such as:

- ❑ Computer where the file(s) are taken from
- ❑ How the computer received the file (email, network, WEB, unknown source, etc...)
- ❑ Was the file embedded into a compressed file? If so, information about the compressed file would have been gathered as well.

It is also important to preserve as much of the computer's environment (registry, files) as possible in case the analysis process requires further extraction to pursue the investigation. In the present case, these circumstances are unknown and will not be described.

### File Information

The following information is presented as if there were only a single file:

File name: msrll.exe  
File date and time: 2004-05-10 16:29  
File size: 41 984 bytes  
Runs on: Windows operating systems  
File MD5 signature: 84acfe96a98590813413122c12c11aaa \*msrll.exe

This information is essential to correctly identify the malicious code during analysis as well as information exchanged between people investigating the same case.

### Embedded Strings

A first attempt to retrieve strings embedded (using `BinText` tool) in the 'msrll.exe' file showed that the program was packed (compressed). The tool retrieved only a few significant text strings. However, the first strings seemed to represent program sections. This was confirmed using the `PEInfo` tool.

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

**Table 1 – Msrll.exe - Program Sections**

Once the program was unpacked, a large amount of text strings were found. Appendix B shows the complete text string list. For details about the unpacking process, refer to the Code Analysis chapter.

A glance at the string list showed some clues that may be used in the analysis process:

- ❑ Many strings begin with a question mark ('?') followed by strings that may represent some kind of commands. Here are a few examples:  
`?login, ?uptime, ?reboot, ?status, ?jump, ?nick, ?echo, ...`
  - ❑ Strings including '%s' and '%u' seem to be answers or error messages
  - ❑ Some strings seem to refer to IRC parameters: Here are some examples:  
`irc.user irc.usereal irc.real irc.pass, ...`
  - ❑ Other strings have the same format but refer to something named 'jtr':  
`jtr.bin, jtr.home, jtr.id, jtr.%u%s.iso, jtr.*,`
  - ❑ One string refers to an Internet domain name and communication ports:  
`collective7.zxy0.com, collective7.zxy0.com:9999!, collective7.zxy0.com:8080`
- 

© SANS Institute 2000 - 2005, All rights reserved.

## Behavioral Analysis

This section describes the observed behavior when the `msrll.exe` file was activated. It is divided into two parts. The first one addresses the observed behavior when the program is run for the first time. The second one describes the behavior of the code when 'comfortably installed' on a computer.

### *Msrl.exe 'first run'*

A first attempt was made to run the '`msrll.exe`' program that was made using a user account. This attempt showed almost no results. This was later confirmed using the `RegShot` program, other tools requiring an administrative right to start. Appendix C presents `RegShot` comparison results. Every other experiment was made using an administrator account.

Several tools were activated to observe the program's behavior when it was run for the first time. Most of them were active on the target computer (`RegShot`, `FileMon`, `RegMon` and `TDImon`). The `snort` program was also used on the Linux computer to capture eventual traffic emanating from the target computer. Following are the most important observations about the malware's first run.

### File Monitoring

When first launched, the program's objective regarding the file system appeared to hide itself in the system files and 'disappear' from its original location. To confirm this hypothesis, a few extracts were taken from the file monitor (`FileMon`).

#### Create 'mfm' folder

```
msrll.exe:560    CREATE C:\WINNT\system32\mfm  SUCCESS Options: Create
Directory Access: All
```

#### Copy itself in 'mfm' folder

```
msrll.exe:560    CREATE C:\WINNT\system32\mfm\msrll.exe SUCCESS
Options: OverwriteIf Sequential Access: All
```

```
msrll.exe:560 WRITE C:\WINNT\system32\mfm\msrll.exe      SUCCESS
Offset: 0 Length: 41984
```

#### Set attributes about it's file

```
msrll.exe:560    SET INFORMATION C:\WINNT\system32\mfm\msrll.exe
SUCCESS
```

#### Delete the original file

```
msrll.exe:560    DELETE C:\Resultats\msrll.exe SUCCESS
```

**Table 2 – File Monitoring**

### Registry Monitoring



On the registry level, 'msrll.exe' explores many registry values, possibly to learn about the environment in which it runs. Registry monitoring (using RegMon) also reveals how the programs arrange the system settings to start automatically when the computer is started. A log extract from RegMon shows how 'msrll.exe' uses Windows Services.exe to register as a service started up at boot time:

```
SERVICES.EXE:212 OpenKey HKLM\System\CurrentControlSet\Services SUCCESS Key:
0xE1C88660
SERVICES.EXE:212 CreateKey HKLM\System\CurrentControlSet\Services\mfm SUCCESS Key:
0xE1E92760
SERVICES.EXE:212 CloseKey HKLM\System\CurrentControlSet\Services SUCCESS Key:
0xE1C88660
SERVICES.EXE:212 SetValue HKLM\System\CurrentControlSet\Services\mfm\Type SUCCESS
0x120
SERVICES.EXE:212 SetValue HKLM\System\CurrentControlSet\Services\mfm\Start SUCCESS
0x2
SERVICES.EXE:212 SetValue HKLM\System\CurrentControlSet\Services\mfm\ErrorControl
SUCCESS 0x2
SERVICES.EXE:212 SetValue HKLM\System\CurrentControlSet\Services\mfm\ImagePath
SUCCESS "C:\WINNT\system32\mfm\msrll.exe"
SERVICES.EXE:212 SetValue HKLM\System\CurrentControlSet\Services\mfm\DisplayName
SUCCESS "Rll enhanced drive"
SERVICES.EXE:212 CreateKey HKLM\System\CurrentControlSet\Services\mfm\Security
SUCCESS Key: 0xE1C88660
SERVICES.EXE:212 SetValue
HKLM\System\CurrentControlSet\Services\mfm\Security\Security SUCCESS 01 00 14 80
A0 00 00 00 ...
SERVICES.EXE:212 CloseKey HKLM\System\CurrentControlSet\Services\mfm\Security
SUCCESS Key: 0xE1C88660
SERVICES.EXE:212 SetValue HKLM\System\CurrentControlSet\Services\mfm\ObjectName
SUCCESS "LocalSystem"
```

**Table 3 – Registry Monitoring**

## Cryptography

The msrll.exe program shows many traces regarding cryptography. Embedded strings in the unpacked file show references to crypto routines (such as Blowfish, Rijndael, XTEA, Twofish, SHA-512, SHA1, MD5, etc). Registry monitoring also shows activity regarding cryptography:

```
msrll.exe:504 CreateKey HKLM\SOFTWARE\Microsoft\Cryptography\RNG SUCCESS Key:
0xE139DA80
msrll.exe:504 SetValue HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed SUCCESS E3
0E 6E 3A 5E 8A EA 63 ...
```

**Table 4 - Cryptography**

## Confirming Observations

Most observations made dynamically with Filemon and Regmon tools are confirmed using RegShot. This tool was used to compare the computer's state before and after the 'msrll.exe' program had completed its installation. An extract from the comparison log shows both the created and deleted files:

Files added:2
-----
C:\WINNT\system32\mf\mjtram.conf
C:\WINNT\system32\mf\msrll.exe
-----
Files deleted:1
-----
C:\Resultats\TP\msrll.exe

Table 5 – RegShot - Files

### ***Behavior Once Installed***

Before going further, it appears appropriate to define some terms used to describe the program's behavior.

A 'zombie' or 'zombie computer' is a computer that is under someone else's control without the knowledge of its legitimate user or administrator. This computer is generally used as an intermediary to perform 'Denial of Service' attacks (DoS).<sup>1</sup> A 'zombie army' is a collection of many computers under the same control. To succeed with a DoS attack, malicious persons need to synchronize the behavior of many computers into attacking the same target at the same time.

A 'zombie agent' or 'agent' is a program that runs on a computer to automate a specific task. In this document, the interest for this kind of program is its capability to react to orders that are issued to it, generally from an outside source, and which influence the computer's actions. Agents are also referred as 'bots', which is short for 'robot'.

The following figure facilitates the understanding of the 'msrll.exe' program detailed in the next pages.

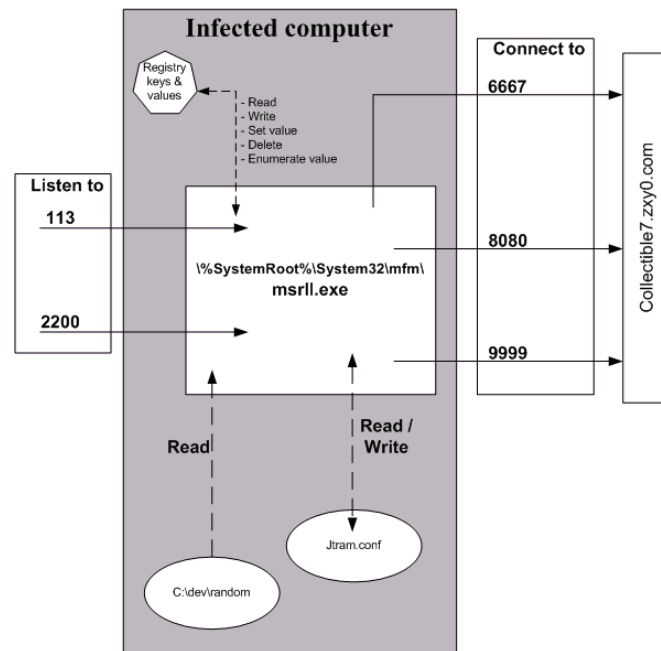


Figure 2 – msrll.exe Behavior

## Agent Behavior

Once installed, the original program terminates itself and starts the newly installed file. This program may also be called an 'agent'. Its behavior, such as trying to create the directory where it has already been copied, shows that it is probably the same file. This is confirmed using MD5sum, which generates an identical hash.

```
Try to create 'mf' folder
msrll.exe:1044      CREATE C:\WINNT\system32\mf NAME COLLISION Options:
                   Create Directory Access: All
```

Table 6 – Folder Creation – Name Collision

## 'msrll.exe' Startup

The 'msell.exe' program may also be started as a service when the system is booted. It can be started as a regular program if activated under the Windows explorer or a command shell.

The 'msrll.exe' process cannot be killed using Windows task manager. The 'process explorer' tool can both investigate and kill the process.

The 'msrll.exe' process disables GUI safemode by rebooting the computer if an attempt is made to enter into this mode. Text safemode is still functional.

## 'msrll.exe' Auxiliary Files

One of the first noticeable tasks performed by 'msrll.exe' is to create the `C:\WINNT\system32\mf\mjtram.conf` file. To create this file, it looks for `c:\dev\random`. If the file is absent, the program seems to rely on itself to generate the random strings.

On Unix/Linux systems, the `/dev/random` file is used to generate random strings. Its counterpart (as a file) does not exist on Windows systems. If the file name `c:\dev\random` is created and filled with some strings in it, the program reads it like a real random string generator.

The following FileMon log extracts corroborate this hypothesis:

**File not found:**

```
1029 19:47:36 msrll.exe:972 OPEN C:\dev\random PATH NOT FOUND Options:
Open Access: All
1030 19:47:36 msrll.exe:972 OPEN C:\dev\random PATH NOT FOUND Options:
Open Access: All
1031 19:47:36 msrll.exe:972 WRITE C:\WINNT\system32\mf\mjtram.conf
SUCCESS Offset: 0 Length: 53
```

**File present:**

```
2673 20:02:36 msrll.exe:252 OPEN C:\dev\random SUCCESS Options: Open
Access: All
2674 20:02:36 msrll.exe:252 READ C:\dev\random SUCCESS Offset: 0
Length: 16
2675 20:02:36 msrll.exe:252 CLOSE C:\dev\random SUCCESS
2676 20:02:36 msrll.exe:252 WRITE C:\WINNT\system32\mf\mjtram.conf
SUCCESS Offset: 0 Length: 53
```

**Table 7 – Auxiliary Files**

The `C:\winnt\system32\mf\mjtram.conf` file is read and recreated every time the 'msrll.exe' program is started. It is then updated (or re-created if renamed or deleted) every hour. Update time depends on the 'msrll.exe' starting time (boot time). File content varies every time. The file is updated even if the agent is not able to reach its server to receive orders (if server is unreachable or if targeted ports (IRC (6667), 8080 and 9999) do not answer (port RESET)). It is also possible to force the file to be updated using the '?dump' command (refer to the 'agent control' section).

## Environment Information

File and registry monitoring shows 'interest' from the 'msrll.exe' program about the computer environment. It reads and even modifies registry values regarding 'Internet Explorer', and 'Document and settings keys and files. It acts the same with cryptography keys and 'DLL' files.

## Network Activities

Network activities were monitored using the TDIMon tool on the infected computer and Snort on the Linux computer. Snort can trace any traffic on the network while TDIMon can reveal listening processes not traceable with Snort if

not solicited.

'msrll.exe' begins its network activities by looking for a 'master site'. This is done using DNS request for the 'collective7.zxy0.com' domain.

#### Snort trace – DNS request

```
11/12-15:03:42.437367 192.168.116.110:1029 -> 192.168.116.1:53
UDP TTL:128 TOS:0x0 ID:186 IpLen:20 DgmLen:66 Len: 38
E4 AD 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.....
```

**Table 8 - DNS Request**

To give the malicious code access to a fake collective7.zxy0.com server, the host table was edited to point this name to the linux computer's interface.

In reaction, 'msrll.exe' then starts to listen on port 113 and port 2200. It then tries to connect to the collective7.zxy0.com server on port 6667, an IRC port.

#### Alternate Ports (8080 and 9999)

If the IRC server is not available, the agent ('msrll.exe') tries to connect to port 8080. If successful, it sends some identification strings to it. The agent appears to wait for a command for 5 seconds then closes the connection. Experiments did not succeed in provoking a reaction from the agent with this connection.

#### Snort trace – connection on port 8080

```
11/22-07:27:46.730978 192.168.116.128:1030 -> 192.168.116.129:8080
TCP TTL:128 TOS:0x0 ID:82 IpLen:20 DgmLen:92 DF
***AP*** Seq: 0x8C0C546D Ack: 0x4A43DBB Win: 0xFAF0 TcpLen: 20
55 53 45 52 20 6F 43 6A 4A 7A 4C 42 20 6C 6F 63 USER oCjJzLB loc
61 6C 68 6F 73 74 20 30 20 3A 61 6B 70 4F 72 66 alhost 0 :akpOrf
76 0A 4E 49 43 4B 20 6D 55 6C 73 6B 59 74 69 62 v.NICK mUlskYtib
47 4D 61 0A GMa.
```

#### Netcat capture on port 8080

```
USER ymPSNwcKHRf localhost 0:fdGWTMtySwEDlRWmvPvPSvvfWocGZn
NICK iUHyQPxrWh
```

**Table 9 – Port 8080 Activity**

If not successful on port 8080, the agent repeats the same process on port 9999.

#### Successful IRC Connection

If the 'master computer' has an IRC server listening, the agent connects itself to an IRC channel named '#mils'. Experiments did not succeed in provoking a reaction from the agent with this connection.

## Snort trace – IRC connection request

```
11/23-04:00:16.987346 192.168.116.128:1026 -> 192.168.116.129:6667
TCP TTL:128 TOS:0x0 ID:23 IpLen:20 DgmLen:48 DF
*****S* Seq: 0xEBCC1B9A Ack: 0x0 Win: 0xFAF0 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
6D 00 00 01 00 01
```

### Snort trace – IRC server tries to authenticate the connecting computer

[illegible]

## Snort trace – IRC connection to server

```

11/23-04:00:17.169372 192.168.116.128:1026 -> 192.168.116.129:6667
TCP TTL:128 TOS:0x0 ID:26 IpLen:20 DgmLen:129 DF
***AP*** Seq: 0xEBCC1B9B Ack: 0xC01189CD Win: 0xFAC2 TcpLen: 20
55 53 45 52 20 49 47 64 67 46 63 46 45 6B 57 20 USER IGdgFcFEkW
6C 6F 63 61 6C 68 6F 73 74 20 30 20 3A 50 57 67 localhost 0 :PWg
79 54 68 50 48 6D 72 6F 67 47 6D 64 68 6A 51 5A yThPHmrogGmdhjQZ
4D 77 48 44 78 6E 50 43 52 47 41 74 6B 4D 46 65 MwHDxpPCRGAatkMFe
4C 69 6D 6A 77 70 0A 4E 49 43 4B 20 6E 61 45 63 Limjwn.NICK naEc
43 6B 77 4D 46 41 76 74 0A CkwMFAvt.

```

## Snort trace – IRC infected computer joins #mils channel

```
11/23-04:00:17.989294 192.168.116.128:1026 -> 192.168.116.129:6667
TCP TTL:128 TOS:0x0 ID:23 IpLen:20 DgmLen:48 DF
*****S* Seq: 0xEBCC1B9A Ack: 0x0 Win: 0xFAF0 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
6D 00 00 01 00 01
```

### Table 10 - IRC Connection

The agent reacts to an IRC `'PING'` request from the server by issuing a `'PONG'` answer. This shows the agent's capability to wait for strings on this connection and to take action based on the input supplied.

## Port 2200

---

With original code of 'msrll.exe', a connection to port 2200 on the target computer using a telnet, is answered by a two character string '#:' prompt. You may enter some character strings on the first two lines. If they do not correspond to something expected, the communication is closed.

A modified version of the agent (see 'code analysis section') was patched to bypass this barrier. It was possible to enter commands and analyze the agent reactions.

If attempting to connect another computer a second telnet on port 2200 while the first one is still active, the agent issues a message to the first session: '\*\*\* bot.port: connect from <second computer IP>'. MSRLL does not answer to the second session.

## Agent Commands

---

Once the "authentication process" is deactivated, a session begins by entering string on the first two lines after the '#:' prompt. String entered on first line is displayed on the connection on port 2200 by the '?sklist' command.

Experiments identified four command types.

- ❑ **Information** commands return information about the agent. These commands are: '?si, ?status, ?echo and ?uptime'. The '?set' without parameters may also be considered as an informative command.
- ❑ **Agent controls** are used to control the agent's actions
- ❑ **Socket commands** are used to manage the agent's IRC communications
- ❑ **Attack commands** instruct the agent to perform an attack on the specified target. Attack commands are: '?ping, ?udp, ?jolt, ?syn and ?smurf'.

**Funny detail:** Embedded strings show commands beginning with a question mark. However, the agent accepts commands if they begin with other characters (except numbers or letters) such as: ., ; \_ : - = + / \ # | ? % ( ) etc...

Appendix D presents the complete command list and the experiment results for each of them. Some commands kept their secrets. This is the case for the '?lsmode, ?insmode and ?rmode' commands. However, it is possible that they represent the remains of a piece of code made to run in a Unix/linux environment. Their names look like names for this kind of system.

## Code Analysis

As mentioned earlier, the code analysis process was performed in collaboration with the behavioral analysis. This section presents the results of the work done using tools like 'BindText, PEInfo, AsPackDie, OllyDbg'.

### Unpacking 'msrll.exe'

Clues to find out the program packer used to pack the 'msrll.exe' program were given by the BindText and PeInfo tools. As the following extract shows, a program section was named as a packer program ('AsPack'):

```
Section Name:      .aspack
VirtualAddress:    0051D000
VirtualSize:       00002000 (8192)
SizeOfRawData:     00002000 (8192)
PointerToRawData:  0011D000
Section characteristics:
    Contains initialized data
    Default alignment (16 bytes)
    Is readable
    Is writeable
```

Table 11 - AsPack Section Name

The 'AsPackDie' program was used to successfully unpack the program. Some tests showed that the behavior of the unpack version was similar to the original file. From that moment, the unpacked code was use to conduct the analysis.

Some experiments with both 'IDA pro' and 'OllyDbg' showed that 'OllyDbg' alone could be used to complete the analysis. The following code segments were extracted using 'OllyDbg'.

### "Authentication Bypass"

A clue to find this piece of code was at:

```
0040BB52 . 25 73 20 62 61>ASCII "%s bad pass from"
0040BB62 . 20 22 25 73 22>ASCII " "%s"@%s",0
```

Table 12 - Authentication Error Messages

These strings suggested that the following code performed some kind of 'authentication'.

To bypass the "authentication requirement" that controls access on port 2200, NOP operations are used to replace the 'JUMP' operation at address 0040BBE9



(JE SHORT msrll.0040BC5A). To gain access to the channel, any strings will be accepted on the first two lines (hit 'return' between the two chains). On the third line, commands are accepted. The first character string (first line) seems to be used as a "name". The '?sklist' command's result shows this string as identifying the connection on port 2200 by the agent.

Original code		
0040BBDE	.52	PUSH EDX ;  Arg1
0040BBDF	.E8 8E9CFFFF	CALL msrll.0040587
0040BBE4	.83C4 10	ADD ESP,10
0040BBE7	.85C0	TEST EAX,EAX
0040BBE9	.74 6F	JE SHORT MSRLL.0040BC5A
0040BBEB	.83EC 0C	SUB ESP,0C
Modified code		
0040BBDE	.52	PUSH EDX ;  Arg1
0040BBDF	.E8 8E9CFFFF	CALLmsrll.0040587
0040BBE4	.83C4 10	ADD ESP,10
0040BBE7	.85C0	TEST EAX,EAX
0040BBE9	.90	NOP
0040BBEA	.90	NOP
0040BBEB	.83EC 0C	SUB ESP,0C

Table 13 - Authentication Bypass

## Finding Commands

To gather the list of all possible commands for connection on port 2200, a first search was performed to find out program calls with names related to the TCP network system call.<sup>2</sup> Appendix E shows a table for some of these calls as well as a trace of network connections made by 'msrll.exe'.

After several experiments with 'OllyDbg', it was found that setting a breakpoint at the address 004089F8 (in red in the next 'OllyDbg' extract) gives the best opportunity to look at every command name accepted by 'msrll.exe'. At that point, the EAX register (easily readable in the 'OllyDbg' 'Register pane') contains the value of one valid command. The program runs through a loop until the subroutine finds a match or runs out of possibilities. Issuing commands in the telnet session with port 2200 to make the program going into that loop, an analyst can find every string acceptable by the agent.

'OllyDbg' extract			
004089F7	. 50	PUSH EAX	;  Arg1
004089F8	. E8 A6440000	CALL msrll.0040CEA3	; \msrll.0040CEA3
004089FD	. 83C4 10	ADD ESP,10	
00408A00	. 85C0	TEST EAX,EAX	
00408A02	. 74 59	JE SHORT msrll.00408A5D	
00408A04	. 8B43 08	MOV EAX,DWORD PTR DS:[EBX+8]	
00408A07	. A9 10000000	TEST EAX,10	
00408A0C	. 75 4F	JNZ SHORT msrll.00408A5D	
00408A0E	. A9 00000200	TEST EAX,20000	
00408A13	. 74 15	JE SHORT msrll.00408A2A	
00408A15	. 6A 00	PUSH 0	
00408A17	. 68 EC884000	PUSH msrll.004088EC	; ASCII "hmm"
00408A1C	. FF75 10	PUSH DWORD PTR SS:[EBP+10]	
00408A1F	. FFB6 84040000	PUSH DWORD PTR DS:[ESI+484]	
00408A25	. FF53 04	CALL DWORD PTR DS:[EBX+4]	
00408A28	. EB 29	JMP SHORT msrll.00408A53	
00408A2A	> 83EC 0C	SUB ESP,0C	
00408A2D	. FF75 10	PUSH DWORD PTR SS:[EBP+10]	
00408A30	. E8 3C1C0000	CALL msrll.0040A671	
00408A35	. 6A 00	PUSH 0	
00408A37	. 68 EC884000	PUSH msrll.004088EC	; ASCII "hmm"
00408A3C	. FF75 10	PUSH DWORD PTR SS:[EBP+10]	
00408A3F	. FFB6 84040000	PUSH DWORD PTR DS:[ESI+484]	
00408A45	. FF53 04	CALL DWORD PTR DS:[EBX+4]	
00408A48	. 83C4 14	ADD ESP,14	

Table 14 - Command Names &amp; Command Subroutines Call

### ***Finding command routines***

Once the code validating commands founded, it is relatively simple to find any subroutine associated with a specific command. Setting a breakpoint at address 00408A45 (in yellow in the previous code extract) pauses the 'msrll.exe' agent just before it jump to the subroutine corresponding to the selected command. Executing a program step in 'OllyDbg' (F7 key) makes the program go to the subroutine entry point. For example, using this method on the '?smurf' command shows that the subroutine executing it is located at 00402284.

---

## Analysis Wrap-up

---

### *Program Capabilities*

---

To review the capabilities of the 'msrll.exe' program, we learned:

- ❑ The program cannot install itself if run under a user privilege account.
- ❑ The program hides itself on a computer in a directory named 'mfm' under `c:\%systemroot%\System32\`.
- ❑ It normally runs as a service, but can be started a regular program (the `?status` command informs the malicious person controlling this agent of the current running mode of the program).
- ❑ The agent is configurable using the `?set` and `?dump` commands.
- ❑ The objective of this malicious program is to take control of computers to perform Denial of Service attacks (DoS).
- ❑ It reports to the site controlling it using an IRC channel.
- ❑ Ports 8080 and 9999 appear to be backup communication channels.
- ❑ Ports 113 is used to facilitate the computer authentication on the IRC server but is not mandatory.
- ❑ 'msrll.exe' accepts commands on port 2200 once an authentication is accepted. To be efficient, a malicious person would probably use a program to issue commands to an army of such infected computers (zombies). Another possibility to control such an army is to issue commands on the IRC channel. This possibility was not confirmed.

### *Protect against it!*

---

The simplest way to protect against 'msrll.exe' is to operate the computer using a user account for day-to-day work.

### *Detect it!*

---

In its Current form, it is possible to detect this malware by looking for the presence of an 'mfm' directory under `c:\%systemroot%\System32\`. This directory should contains two files: 'msrll.exe' and 'jtram.conf'.

### *Contain it!*

---

To contain the action of this agent, it is possible to block outgoing ports 6667, 8080 and 9999 trying to reach the 'collective7.zxy0.com' domain. In addition, blocking the incoming access to ports 2200 is mandatory. Access to port 113 could also be blocked.

### *Eradicate it!*

---

The minimal action to remove this malware is to delete all 'msrll.exe' files. If no anti virus can detect this program and erase every trace of it, a script could

be constructed to erase the files and Registry entries made by the malicious code.

It would be interesting to go deeper into this program analysis, but it appears that the information gathered is sufficient to deal with this threat.

© SANS Institute 2000 - 2005, Author retains full rights.

## Appendices

### Appendix A – Checklist Examples

#### Malicious Code Loading

Action	Done (check)	Comments
Host computer disconnected from the production network		
Host computer disconnected from the 'host network'		
First Target computer reset Fresh copy of computer image OR VMware Snapshot revert		
Anti-virus active detection turned OFF		
ISO image creation		Files included in the ISO image:
Anti-virus active detection turned ON		
Files loaded on first target computer		Files destination (folder):
CD-ROM drive deactivated on first target computer		

#### Transferring Results Files

Action	Done (check)	Comments
Host computer disconnected from the network		
Host computer disconnected from the 'host network'		
Anti-virus active detection turned ON		
Activate FTP server on 'monitor' computer		
Transfer files from target computer(s) to 'monitor' computer. (targets 'put' files)		File list:

Transfer files from 'monitor' computer to host computer (host computer 'get' files)		File destination folder:
Perform anti-virus detection on destination folder		

© SANS Institute 2000 - 2005, Author retains full rights

## Appendix B - Embedded Strings

!This program cannot be run	?ping
in DOS mode.	?smurf
[AspackDie!]	?jolt
.text	PONG :%s
.data	0h (@
.idata	%s!%s@%s
.aspack	%s!%s
.adata	SVh=+@
?insmod	irc.nick
?rmmod	NICK %s
?lsmod	NETWORK=
%s: <mod name>	irc.pre
%s: mod list full	__%s__
%s: err: %u	__%s__
mod_init	__%s__
mod_free	NICK %s
%s: cannot init %s	%s %s
%s: %s loaded (%u)	irc.chan
%s: mod already loaded	%s %s
%s:%s err %u	WHO %s
%s:%s not found	PPhV,@
%s: unloading %s	USERHOST %s
[%u]: %s hinst:%x	logged into %s(%s) as %s
unloading %s	<\$hE:@
%s: invalid_addr: %s	PhR:@
%s%s [port]	nick.pre
finished %s	%s-%04u
%s <ip> <port> <t_time>	irc.user
<delay>	irc.usereal
sockopt: %u	irc.real
sendto err: %u	irc.pass
sockraw: %u	tsend(): connection to
syn: done	%s:%u failed
%s <ip> <duration> <delay>	USER %s localhost 0 :%s
sendto: %u	NICK %s
jolt2: done	Ph <@
%s <ip> <p size> <duration>	PRIVMSG
<delay>	trecv(): Disconnected from
Err: %u	%s err:%u
smurf done	NOTICE
PhV#@	%s %s :%s
&err: %u	Ph}D@

```

MODE %s -o+b %s *@%s
C'PSWh
Sh'G@
MODE %s -bo %s %s
Sh'G@
%s.key
Ph'G@
sk#%u %s is dead!
s_check: %s dead?
pinging...
PING :ok
s_check: send error to %s
disconnecting
expect the worst
s_check: killing socket %s
irc.knick
jtr.%u%s.iso
ison %s
servers
s_check: trying %s
Ph9K@
PhkK@
ShtK@
uYVh|K@
%s.mode
MODE %s %s
ShRP@
Sh$I@
PShZP@
mode %s +o %s
akick
mode %s +b %s %s
KICK %s %s
irc.pre
Set an irc sock to preform
%s command on
Type
%csklist
to view current sockets,
then
%cdccsk
<#>
%s: dll loaded
%s: %d
RhHY@
RhHY@

said %s to %s
usage: %s <target> "text"
%s not on %s
usage: %s <nick> <chan>
%s logged in
Sh [@
sys: %s bot: %s
preformance counter not
avail
usage: %s <cmd>
%s free'd
unable to free %s
0h+\@
later!
unable to %s errno:%u
service:%c user:%s inet
connection:%c contype:%s
reboot privs:%c
Ph@]@
%-5u %s
%s: %s
%s: somefile
PhHY@
host: %s ip: %s
capGetDriverDescriptionA
cpus:%u
WIN%s (u:%s)%s%s
mem:(%u/%u) %u%% %s %s
%s: %s (%u)
%s %s
%s bad args
3hTg@
akick
%s[%u] %s
%s removed
couldnt find %s
%s added
%s allready in list
usage: %s +/- <host>
7h*h@
jtram.conf
%s /t %s
jtr.home
%s\s
%s: possibly failed: code
%u

```



```

%s: possibly failed
%s: exec of %s failed err:
%u
u.exf
Ph+j@
Ph?j@
jtr.id
%s: <url> <id>
IREG
CLON
ICON
WCON
#%u [fd:%u] %s:%u [%s%s]
last:%u
|\> [n:%s fh:%s] (%s)
|---[%s] (%u) %s
|      |---[%s%s] [%s]
|=> (%s) (%.8x)
B$PRhco@
%s <pass> <salt>
%s <nick> <chan>
PING %s
mIRC v6.12 Khaled Mardam-
Bey
VERSION %s
dcc.pass
temp add %s
$h%u@
%s%u-%s
%s opened (%u)
%u bytes from %s in %u
seconds saved to %s
(%s %s): incomplete! %u
bytes
couldnt open %s err:%u
(%s) %s: %s
(%s) urlopen failed
(%s): inetopen failed
Whjv@
Ph w@
no file name in %s
%s created
%s %s to %s Ok
3hI~@
%0.2u/%0.2u/%0.2u
%0.2u:%0.2u %15s %s

%s (err: %u)
ShHY@
err: %u
%s %s :ok
unable to %s %s (err: %u)
ShHY@
%-16s %s
%-16s (%u.%u.%u.%u)
[%s][%s] %s
closing %u [%s:%u]
unable to close socket %u
using sock #%u %s:%u (%s)
Invalid sock
usage %s <socks #>
leaves %s
:0 * * :%s
joins: %s
ACCEPT
resume
err: %u
DCC ACCEPT %s %s %s
dcc_resume: cant find port
%s
dcc.dir
%s\%s\%s\%s
unable to open (%s): %u
resuming dcc from %s to %s
DCC RESUME %s %s %u
?clone
?clones
?login
?uptime
?reboot
?status
?jump
?nick
?echo
?hush
?wget
?join
?akick
?part
?dump
?md5p
?free
?update

```

?hostname	send of %s completed (%u
?!fif	bytes), %u seconds %u cps
?play	cant open %s (err:%u)
?copy	pwd:{%s}
?move	DCC SEND %s %u %u %u
?sums	%s %s
?rmdir	%s exited with code %u
?mkdir	%s\%s
?exec	%s: %s
?kill	exec: Error:%u pwd:%s
?killall	cmd:%s
?crash	dcc.pass
?sklist	bot.port
?unset	%s bad pass from "%s"@%s
?uattr	%s: connect from %s
?dccsk	jtr.bin
?killsk	msrll.exe
VERSION*	jtr.home
IDENT	jtr.id
%ud %02uh %02um %02us	irc.quit
%02uh %02um %02us	servers
%um %02us	collective7.zxy0.com,collec
jtram.conf	tive7.zxy0.com:9999!,collec
jtr.*	tive7.zxy0.com:8080
DiCHFc2ioiVmb3cb4zZ7zWZH1oM	irc.chan
=	#mils
conf_dump: wrote %u lines	\$1\$KZLPLKDF\$W8kl8Jr1X8DOHZs
get of %s incomplete at %u	mIp9qq0
bytes	\$1\$KZLPLKDF\$55isA1ITvamR7bj
get of %s completed (%u	AdBziX.
bytes), %u seconds %u cps	SSL_get_error
error while writing to %s	SSL_load_error_strings
(%u)	SSL_library_init
chdir: %s -> %s (%u)	SSLv3_client_method
dcc_wait: get of %s from %s	SSL_set_connect_state
timed out	SSL_CTX_new
dcc_wait: closing [#%u]	SSL_new
%s:%u (%s)	SSL_set_fd
%4s #%.2u %s %ucps %u%	SSL_connect
[sk#%u] %s	SSL_write
%u Send(s) %u Get(s) (%u	SSL_read
transfer(s) total) UP:%ucps	SSL_shutdown
DOWN:%ucps Total:%ucps	SSL_free
PRQh0	SSL_CTX_free
send of %s incomplete at %u	kernel32.dll
bytes	QueryPerformanceCounter

QueryPerformanceFrequency	#4EVgx
RegisterServiceProcess	\$5FWhy
jtram.conf	#4EVgx
irc.user	\$5FWhy
%s : USERID : UNIX : %s	#4EVgx
QUIT :FUCK %u	\$5FWhy
Killed!? Arrg! [%u]	gN]HU
QUIT :%s	desired_keysize != NULL
SeShutdownPrivilege	ctr.c
%s\s	ctr != NULL
%s\s\s	key != NULL
Rll enhanced drive	count != NULL
software\microsoft\windows\	ct != NULL
currentversion\run	pt != NULL
/d "%s"	ABCDEFGHIJKLMNOPQRSTUVWXYZa
< u&	bcdefghijklmnopqrstuvwxyz01
./0123456789ABCDEFGHIJKLMNO	23456789+/?
PQRSTUVWXYZabcdefghijklmnop	456789;:<=
qrstuvwxyz	!"#\$%&'()*+,-./0123
usage %s: server[:port]	base64.c
amount	outlen != NULL
%s: %s	out != NULL
%s %s %s <PARAM>	in != NULL
%s: [NETWORK all] %s	_ARGCHK '%s' failure on
<"parm"> ...	line %d of file %s
USER %s localhost 0 :%s	crypt.c
NICK %s	name != NULL
PSVh	cipher != NULL
md5.c	hash != NULL
md != NULL	prng != NULL
buf != NULL	LibTomCrypt 0.83
hash != NULL	Endianness: little (32-bit
message digest	words)
abcdefghijklmnopqrstuvwxyz	Clean stack: disabled
ABCDEFGHIJKLMNOPQRSTUVWXYZa	Ciphers built-in:
bcdefghijklmnopqrstuvwxyz01	Blowfish
23456789	RC2
1,23457E+79	RC5
sprng	RC6
sprng.c	Serpent
buf != NULL	Safer+
rc6.c	Safer
skey != NULL	Rijndael
key != NULL	XTEA
ct != NULL	Twofish
pt != NULL	CAST5

```

Noekeon
Hashes built-in:
SHA-512
SHA-384
SHA-256
TIGER
SHA1
MD5
MD4
MD2
Block Chaining Modes:
CFB
OFB
CTR
PRNG:
Yarrow
SPRNG
RC4
PK Algs:
RSA
DH
ECC
KR
Compiler:
WIN32 platform detected.
GCC compiler detected.
Various others:  BASE64
MPI  HMAC
/dev/random
Microsoft Base
Cryptographic Provider v1.0
bits.c
buf != NULL
t9VWS
prng != NULL
"<"tx< tf<      t"
"< tV<      t"
"< tJ<      tF"
#NOM?
<ip> <total secs> <p size>
<delay>
modem
Lan
Proxy
none
m220 1.0 #2730 Mar 16

11:47:38 2004
unable to %s %s (err: %u)
unable to kill %s (%u)
%s killed (pid:%u)
AVICAP32.dll
unable to kill %u (%u)
pid %u killed
error!
ran ok
MODE %s +o %s
set %s %s
Mozilla/4.0
Accept: */*
<DIR>
Could not copy %s to %s
%s copied to %s
0123456789abcdef
%s unset
unable to unset %s
(%s) %s
%s %s
libssl32.dll
libeay32.dll
<die|join|part|raw|msg>
AdjustTokenPrivileges
CloseServiceHandle
CreateServiceA
CryptAcquireContextA
CryptGenRandom
CryptReleaseContext
GetUserNameA
LookupPrivilegeValueA
OpenProcessToken
OpenSCManagerA
RegCloseKey
RegCreateKeyExA
RegSetValueExA
RegisterServiceCtrlHandlerA
SetServiceStatus
StartServiceCtrlDispatcherA
AddAtomA
CloseHandle
CopyFileA
CreateDirectoryA
CreateFileA
CreateMutexA

```

CreatePipe	SetConsoleCtrlHandler
CreateProcessA	SetCurrentDirectoryA
CreateToolhelp32Snapshot	SetFilePointer
DeleteFileA	SetUnhandledExceptionFilter
DuplicateHandle	Sleep
EnterCriticalSection	TerminateProcess
ExitProcess	WaitForSingleObject
ExitThread	WriteFile
FileTimeToSystemTime	_itoa
FindAtomA	_stat
FindClose	_strdup
FindFirstFileA	_stricmp
FindNextFileA	__getmainargs
FreeLibrary	__p__environ
GetAtomNameA	__p__fmode
GetCommandLineA	__set_app_type
GetCurrentDirectoryA	_beginthread
GetCurrentProcess	_cexit
GetCurrentThreadId	_errno
GetExitCodeProcess	_fileno
GetFileSize	_onexit
GetFullPathNameA	_setmode
GetLastError	_vsnprintf
GetModuleFileNameA	abort
GetModuleHandleA	atexit
GetProcAddress	clock
GetStartupInfoA	fclose
GetSystemDirectoryA	fflush
GetSystemInfo	fgets
GetTempPathA	fopen
GetTickCount	fprintf
GetVersionExA	fread
GlobalMemoryStatus	fwrite
InitializeCriticalSection	malloc
IsBadReadPtr	memcpy
LeaveCriticalSection	memset
LoadLibraryA	printf
MoveFileA	raise
OpenProcess	realloc
PeekNamedPipe	setvbuf
Process32First	signal
Process32Next	sprintf
QueryPerformanceFrequency	srand
ReadFile	strcat
ReleaseMutex	strchr
RemoveDirectoryA	strcmp

strcpy	SHELL32.DLL
strerror	USER32.dll
strncat	VERSION.dll
strncmp	WININET.DLL
strncpy	WS2_32.DLL
strstr	VirtualAlloc
toupper	VirtualFree
ShellExecuteA	kernel32.dll
DispatchMessageA	ExitProcess
ExitWindowsEx	user32.dll
GetMessageA	MessageBoxA
PeekMessageA	wsprintfA
GetFileVersionInfoA	LOADER ERROR
VerQueryValueA	The procedure entry point
InternetCloseHandle	%s could not be located in
InternetGetConnectedState	the dynamic link library %s
InternetOpenA	The ordinal %u could not be
InternetOpenUrlA	located in the dynamic link
InternetReadFile	library %s
WSAGetLastError	MW dNW
WSASocketA	(08@P
WSAStartup	D41 M
__WSAFDIsSet	; ; F, s
accept	, ; F0s
closesocket	; F4s
connect	D\$\$W3
gethostbyaddr	kernel32.dll
gethostbyname	GetProcAddress
gethostname	GetModuleHandleA
getsockname	LoadLibraryA
htonl	advapi32.dll
htons	msvcrt.dll
inet_addr	msvcrt.dll
inet_ntoa	shell32.dll
ioctlsocket	user32.dll
listen	version.dll
ntohl	wininet.dll
select	ws2_32.dll
sendto	AdjustTokenPrivileges
setsockopt	_itoa
shutdown	__getmainargs
socket	ShellExecuteA
ADVAPI32.DLL	DispatchMessageA
KERNEL32.dll	GetFileVersionInfoA
msvcrt.dll	InternetCloseHandle
msvcrt.dll	WSAGetLastError

© SANS Institute 2000 - 2005, Author retains full rights.

## Appendix C - Regshot Comparison – User Account

---

REGSHOT LOG 1.61e5

Comments:

Datetime:2004/11/12 15:00:11 , 2004/11/12 15:01:59

Computer:TARGET1-2000PRO , TARGET1-2000PRO

Username:User , User

-----  
Keys added:3  
-----

HKEY\_USERS\S-1-5-21-1993962763-1767777339-839522115-  
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Streams\7

HKEY\_USERS\S-1-5-21-1993962763-1767777339-839522115-  
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Streams\8

HKEY\_USERS\S-1-5-21-1993962763-1767777339-839522115-  
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Streams\9

-----  
Values added:12  
-----

HKEY\_USERS\S-1-5-21-1993962763-1767777339-839522115-  
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\StreamMRU\7: 14 00 1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B  
30 30 9D 19 00 23 43 3A 5C 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 31 84 25 00 31 00 00 00 00 00 51 31 E6 A9  
31 00 50 72 6F 67 72 61 6D 20 46 69 6C 65 73 00 50 52 4F 47  
52 41 7E 31 00 17 00 31 00 00 00 00 00 7B 31 E5 9D 10 00 52  
65 67 53 68 6F 74 00 00 00 00

HKEY\_USERS\S-1-5-21-1993962763-1767777339-839522115-  
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\StreamMRU\8: 14 00 1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B  
30 30 9D 19 00 23 43 3A 5C 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 31 84 25 00 31 00 00 00 00 00 51 31 E6 A9  
31 00 50 72 6F 67 72 61 6D 20 46 69 6C 65 73 00 50 52 4F 47  
52 41 7E 31 00 17 00 31 00 00 00 00 00 51 31 E9 A8 10 00 46  
69 6C 65 6D 6F 6E 00 00 00 00

HKEY\_USERS\S-1-5-21-1993962763-1767777339-839522115-  
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\StreamMRU\9: 14 00 1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B  
30 30 9D 19 00 23 43 3A 5C 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 31 84 25 00 31 00 00 00 00 00 51 31 E6 A9



```
31 00 50 72 6F 67 72 61 6D 20 46 69 6C 65 73 00 50 52 4F 47
52 41 7E 31 00 00 00
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Stre
ams\7\CabView: 5C 00 00 00 00 00 00 00 01 00 00 00 FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF 42 00 00 00 42 00 00
00 54 03 00 00 5F 02 00 00 01 00 00 00 00 00 00 00 80 60 08
00 00 00 00 00 56 0F 51 71 07 00 00 00 E0 D0 57 00 73 35 CF
11 AE 69 08 00 2B 2E 12 62 01 00 00 00 00 00 00 00
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Stre
ams\7\ViewView2: 1C 00 00 00 01 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 01 00 00 00 FF FF FF FF F0 F0 F0 F0 14 00
03 00 30 00 00 00 00 00 00 00 00 00 00 00 00 00
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Stre
ams\8\CabView: 5C 00 00 00 00 00 00 00 01 00 00 00 FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF 42 00 00 00 42 00 00
00 54 03 00 00 5F 02 00 00 01 00 00 00 00 00 00 00 80 60 08
00 00 00 00 00 56 0F 51 71 07 00 00 00 E0 D0 57 00 73 35 CF
11 AE 69 08 00 2B 2E 12 62 01 00 00 00 00 00 00 00
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Stre
ams\8\ViewView2: 1C 00 00 00 01 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 01 00 00 00 FF FF FF FF F0 F0 F0 F0 14 00
03 00 30 00 00 00 00 00 00 00 00 00 00 00 00 00
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Stre
ams\9\CabView: 5C 00 00 00 00 00 00 00 01 00 00 00 FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF 42 00 00 00 42 00 00
00 54 03 00 00 5F 02 00 00 01 00 00 00 00 00 00 00 80 60 08
00 00 00 00 00 56 0F 51 71 07 00 00 00 E0 D0 57 00 73 35 CF
11 AE 69 08 00 2B 2E 12 62 01 00 00 00 00 00 00 00
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Stre
ams\9\ViewView2: 1C 00 00 00 01 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 01 00 00 00 FF FF FF FF F0 F0 F0 F0 14 00
03 00 30 00 00 00 00 00 00 00 00 00 00 00 00 00
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\User
Assist\{75048700-EF1F-11D0-9888-
006097DEACF9}\Count\HRZR_EHACNGU:P:\Cebtenz
Svyrf\Svyrzba\Svyrzba.rkr: 00 00 00 00 06 00 00 00 90 6F 1E
4D ED D6 C4 01
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\User
```

```

Assist\{75048700-EF1F-11D0-9888-
006097DEACF9}\Count\HRZR_EHACNGU:P:\Erfhygngf\zfeyy.rkr: 00
00 00 00 06 00 00 00 E0 64 1C 5A ED D6 C4 01
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Run\Rll
enhanced drive: "C:\WINNT\system32\mfm\msrll.exe"

```

```

-----
Values modified:3
-----

```

```

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\RNG\Seed:
A5 D9 87 86 2D 43 17 B7 7E 47 92 89 13 CB DE 45 A0 17 B3 65
31 7C 8B DB 69 8E 8D 68 61 45 BA E2 E9 A7 D1 7E E3 A4 63 7D
DB 84 28 B6 3F 06 49 15 F7 0D 99 3F E7 5C D7 E6 17 EB 6E 3D
F3 C6 79 6B 31 87 B2 11 24 3F 5A B3 87 AC BD 53 59 90 A3 A8
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\RNG\Seed:
B0 96 66 8E D8 84 85 BF D0 7D 9C BF E8 F6 F6 12 70 0D B9 3B
1C 2D DF 9B F8 CB F4 1D A3 EE B1 77 FB E5 D7 75 1A 2D CD 65
02 2B 0D 29 92 AF E8 44 23 1B 38 CB F1 95 A0 6B 48 89 29 C5
66 B5 9F 1C 33 CB A9 58 67 CA 1F 9C 4D 0A DD D9 44 09 F0 96
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Stre
amMRU\MRUListEx: 06 00 00 00 05 00 00 00 04 00 00 00 03 00
00 00 02 00 00 00 01 00 00 00 00 00 00 00 FF FF FF FF
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\Stre
amMRU\MRUListEx: 09 00 00 00 01 00 00 00 08 00 00 00 07 00
00 00 06 00 00 00 05 00 00 00 04 00 00 00 03 00 00 00 02 00
00 00 00 00 00 00 FF FF FF FF
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\User
Assist\{75048700-EF1F-11D0-9888-
006097DEACF9}\Count\HRZR_EHACNGU: 00 00 00 00 0F 00 00 00 B0
50 8D 41 ED D6 C4 01
HKEY_USERS\S-1-5-21-1993962763-1767777339-839522115-
1000\Software\Microsoft\Windows\CurrentVersion\Explorer\User
Assist\{75048700-EF1F-11D0-9888-
006097DEACF9}\Count\HRZR_EHACNGU: 00 00 00 00 11 00 00 00 E0
64 1C 5A ED D6 C4 01

```

```

-----
Total changes:19
-----

```

## Appendix D - Agent Control Commands

Command	Type	Parameter(s)	Effect
'Authentication process'			After displaying a prompt ('#:'), MSRL.EXE seems to wait for <ID> and <password> on the first two lines. String entered on first line is displayed on the connection on port 2200 by the '?sklist' command.
?uptime	Info		Time since system was booted and since the agent (msrll.exe) has been started
?si	Info		Returns information about infected host:  OS-version (u:user) mem: (used/available) CPU-utilization CPU-type CPU-speed  Ex: WIN2k (u:SYSTEM) mem: (175/255) 31% GenuineIntel Intel(R) Pentium(R) 4 Mobile CPU 1.80GHz
?echo	Info	<text>	Echo back the text string
?status	Info		Issues information about the 'zombie computer' status Runs as a service?    Started under which user ??inet connection??    Connection type Reboot privilege?  Ex: service:Y user:SYSTEM inet connection:Y contype: Lan    reboot privs:Y
?con			???????
?ping	Attack	<ip> <total secs> <p size> <delay> [port]	Issues a 'ping attack' from the zombie computer
?udp	Attack	<ip> <total secs> <p size> <delay> [port]	Issues a 'UDP attack' from the zombie computer
?jolt	Attack	<ip> <duration> <delay>	Issues a 'JOLT attack' from the zombie computer

?smurf	Attack	<ip> <p_size> <duration> <delay>	Issues a 'SMURF attack' from the zombie computer
?syn	Attack	<ip> <port> <t_time> <delay>	Issue a 'SYN attack' from the zombie computer
?reboot	Agent Control		Reboots the zombie computer Returns 'later!' as command confirmation
?cd	Agent Control	<directory name>	Issues a "CHANGE DIR command' on the zombie computer to control the agent's current directory
?pwd	Agent Control		Returns the agent's current directory
?dir or ?ls	Agent Control	[filename]	Issues a "dir command' on the zombie computer, returning the directory list to the agent's current directory on the zombie computer. Filename is optional, file name format does not appear to be limited to 8.3 format. Can use a wild card (*). The single-replacement character (?) is also supported but only if not repeated (ex: fil?.txt but not fi???.txt)
?mkdir	Agent Control	<directory name>	Creates a directory on the zombie computer.
?rmdir	Agent Control	<directory name>	Deletes a directory on the zombie computer.
?copy	Agent Control	<filename1> <filename2>	Copies a file (local to the zombie computer) on the zombie computer.
?play	Agent Control	<filename>	Displays the specified file content.
?move	Agent Control	[dir] <filename1> [dir] <filename2>	Moves a file from one directory to another. Can also be used to rename a file.
?del	Agent Control	<filename>	Deletes a file on the zombie computer. The command has problems addressing long file names. Can use a wild card (*). The single-replacement character (?) is also supported but only if not repeated (ex: fil?.txt but not fi???.txt)
?hostname	Agent Control		Returns agent computer's hostname and IP address  Ex: Host:Target1-2000pro IP: 192.168.134.130
?sums	Agent Control		Returns a MD5 signature of every file in the current directory.

?run	Agent Control	<filename>   <command>	Executes the file or the command on the zombie computer.  Returns a confirmation ending with a return code The return code seems to be identical for every command issued : (4151744)
?exec	Agent Control	<filename>   <command>	Similar to '?run' except that the program is not visible on the desktop Does not return confirmation.
?ps	Agent Control		Returns a list of running processes, each one preceded by its process ID (PID).
?kill	Agent Control	<pid>	Terminates the indicated process (using the pid) on the zombie computer.  Returns two confirmation lines: PID <pid> killed <Program name> exited with code <return_code>
?killall	Agent Control	<process name>	Terminates the indicated process (using its name) on the zombie computer.  Returns confirmation: <Program name> killed (PID:<pid>)
?set	Agent Control	<var> <value>	Used without parameters, shows the 'environment' ruling the agent comportment.  Used with parameters, modifies a value for the given variable.  Ex: 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:99 99!,collective7.zxy0.com:8080 set irc.chan #mils set pass \$1\$KZLPLKdf\$W8kl8Jr1X8DOHZsmIp9qq0 set dcc.pass \$1\$KZLPLKdf\$55isA1ITvamR7bjAdBziX.
?unset	Agent Control	<var>	Deletes the variable (and the associated value) from the agent's environment.
?dump	Agent Control		Writes the environment variables into the jtram.conf file in a ciphered form. (see ?set command)
?md5p	Agent Control	<pass> <salt>	Produces a text string with beginning with '\$1\$' followed with the 'salt' passed as second parameter, followed with the '\$' and ends with the ciphered password.  Ex: \$1\$malware\$ppk9WdZ7tMT7skiakRnpm
?free	???	<command>	Disables the specified command. Consequently, the command cannot be used until the agent is restarted on the zombie computer.

?die	Agent Control		Terminates msrll.exe program on the zombie computer
?crash	?		Makes the agent quit the IRC server. In addition, the agent no longer answers to commands issued on port 2200. However, the agent maintains the connection opened on this port.
?update	Agent Control	<url> <id>	Seems to command a file download intended to replace the current version of the agent.
?ssl	Agent Control		Seems to return a status code. Ex: SSL: -1
?wget	Agent Control	<URL> <filename>	Makes the agent go to the specified URL to download a file
?sklist	Socket command		Issues a list of active agent sockets(IP address:port number)  Ex: #1 [fd:424] collective7.zxy0.com:6667 [IRC IATH IREG ICON RNL] last:117  > [n:LyQEMLktC fh:LyQEMLktC!~exYqqrej@192.168.116.128] (WFnet)    ---[#mils] (2) +tn    -[LyQEMLktC] [192.168.116.128]    -[@root] [127.0.0.1] #2 [fd:436] 192.168.116.129:0 [DCC ICON RNL] ] last:0  => (_R.E.M._) (00000021)  Maybe [IRC IATH IREG ICON RNL] and [DCC ICON RNL ] represents authorized commands on displayed sockets
?dccsk	Socket command	<socket #>	Opens a "channel" to relay commands issued on the 2200 port to the selected socket. The socket is specified using the numbers displayed following the '#' character within text returned by the '?sklist' command.

© SANS

?killsk	Socket command	<socket #>	<p>Terminate communication on the selected socket.</p> <p>Based on the sockets showed as result of the ?sklist command, here are two samples of the ?killsk command</p> <p>Killing socket when using the IRC socket (#1) disconnects the agent from IRC server :  Answer  Closing 0 [:2200]</p> <p>Killing socket when using backdoor socket (#2) disconnect the current session with the agent on port 2200:  Answer:  Closing 2 [192.168.116.129:0]  *** leaves R.E.M.</p>
<p>"Socket commands" require a 'socket' to be selected using the sklist and dccsk commands prior to being used.</p>			
?get	Socket command		
?hush	Socket command		<p>This command did not return results. However, 'Hush' is an IRC command. This command is normally issued by an IRC operator (not by a channel operator) to suspend a user from sending visible messages to the server.</p>
?uattr	Socket command	<nick> <chan>	Command had no apparent effect.
?op	Socket command		Command had no apparent effect.
?aop	Socket command		Command had no apparent effect.
?dcc	Socket command		<p>Command had no apparent effect. On IRC, the 'DCC' command is used to initiate a 'direct chat channel' between two users connected on the same IRC network.</p>
?say	Socket command	<target> "text"	Sends a private message in IRC channel to the nickname specified as target.
?kb	Socket command	<nick> <chan>	Command had no apparent effect.
?msg	Socket command	<target> "text"	Sends a private message in IRC channel to the nickname specified as target. Apparently identical to the '?say' command.
?raw	Socket command		
?join	Socket command	<IRC channel>	Makes the zombie computer join the specified IRC channel.
?part	Socket command	<IRC channel>	Makes the zombie computer quit the specified IRC channel.
?akick	Socket command		

?login	?		
?clone	?	<server> [:port] <amount>	
?clones	?	[NETWORK   all] <die   join   part   raw   msg> <"parm"> ... .	
?jump	?		
?nick	?	<password>	Makes the zombie computer changes the nickname used on the IRC server.
?insmod	?	<mod name>	
?rmmod	?		
?lsmod	?		
?fif	?		???? Find File ????
?!fif	?		

© SANS Institute 2000 - 2005, Author



## Appendix E - Connection Sequences

Seq.	Function	Location	Caller	Port	Known addresses		
					Function	Address	RETN add.
1	Listen	0040E4D1	0040C400	2200			
2	Connect	0040D8BC	004050I6	6667			
3	Listen	0040E4D1	00405025	113	<b>Listen</b>	0040E4D1	0040E52B
4	CloseSocket	0040E400	0040E015			0040B1D6	0040B2AE
5	Connect	0040D8BC	004050I6	9999			
6	CloseSocket	0040E400	00404DC7	9999	<b>CloseSocket</b>	004019EF	00401A2C
7	Connect	0040D8BC	004050I6	8080		00401F5C	00401FBB
8	CloseSocket	0040E400	00404DC7	8080		0040222D	00402257
9	Connect	0040D8BC	004050I6			004025C3	004025DD
10	CloseSocket	0040E400	0040C6A0			00402704	00402752
11	CloseSocket	0040E400	00404DC7	113		0040D88A	004008CD
12	ShutDown	0040C672	0040DF62	6667		0040DC0D	0040E326
13	CloseSocket	0040E400	0040E015			0040E400	0040E420
14	Connect	0040D8BC	004050I6	9999			
15	Listen	0040E4D1	00405025	9999	<b>Connect</b>	0040D8BC	004008CD
16	CloseSocket	0040E400	00404DC7				
17	Connect	0040D8BC	004050I6	8080	<b>ShutDown</b>	004086A8	004086D2
18	CloseSocket	0040E400	00404DC7			0040A471	0040A4AF
19	Connect	0040D8BC	004050I6			0040AE6A	0040AEAA
20	CloseSocket	0040E400	0040C6A0			0040BC92	0040BCC8
21	CloseSocket	0040E400	00404DC7			0040C672	0040C67D
22	ShutDown	0040C672	0040DF62				
23	CloseSocket	0040E400	0040E015				
24	Connect	0040D8BC	004050I6				

© SANS

## References

---

<sup>1</sup> Tulloch, Mitch. Microsoft Encyclopedia of Security. Redmond: Microsoft Press 2003 p.385

<sup>2</sup> Wright, Gary R. & Stevens, W. Richards. TCP/IP Illustrated Volume 2: The Implementation. Addison-Wesley 1995 p.449

© SANS Institute 2000 - 2005, Author retains full rights.