

# No Stone Unturned, Part Six

H. Carvey 2002-08-14

## Introduction

This is an additional installment to the [No Stone Unturned](#) series, which was written to help clarify to NT/2K admins the steps they can take to determine the nature and purpose of suspicious files found on their systems. In [Part Five](#) of the series, our heroic system administrator found an unusual file on a compromised system. In this bonus installment, he attempts to determine the nature and purpose of that file.

## Part Six

Eliot sat at his desk. His arms were crossed in front of him on the desktop, and he'd pushed his seat back so he could rest his chin on his arms. Across from him on the desktop was a diskette. Eliot imagined that he could see through the outer plastic shell of the diskette, and visualize the magnetic film inside. He could see in his mind's eye the data that was actually on the diskette.

Just last week, Eliot had responded to a call from Elizabeth to come look at the HR Web server. It seems someone had accessed the server through an old vulnerability, and deposited the file on it. According to the EventLog entries on the Web server, a couple of unusual processes had been run. Additionally, several directories and files had been created on the system. Seeing these artifacts, Eliot had thought that perhaps they were all related, and had copied the files off of the Web server. He'd copied one particular file, `lb.exe`, to a diskette by itself. Another diskette contained the other files he'd found, along with a directory listing. The directory listing included several directories under `c:\program files>manual`, but only one of those directories, `winnt`, contained files:

Directory of C:\Program Files\Manual\WINNT

```
05/04/2002  05:06p      <DIR>          .
05/04/2002  05:06p      <DIR>          ..
04/28/1999  10:46p                33,183  bl.drv
05/04/2002  05:06p                18,342  cco.drv
05/04/2002  05:06p                18,342  cdl.drv
04/11/1999  06:40p                 6,381  cl0ne.dll
```

05/04/2002	05:06p	17,742	cli.drv
05/04/2002	05:06p	18,642	dbp.drv
05/04/2002	05:06p	18,942	dhc.drv
05/04/2002	05:21p	2,690	drx2.inf
05/04/2002	05:05p	2,836	dw.drv
05/04/2002	05:06p	18,642	fsd.drv
05/04/2002	05:06p	17,442	gjj.drv
05/04/2002	05:06p	17,742	gzy.drv
05/04/2002	05:06p	17,742	iib.drv
05/04/2002	05:06p	18,042	jee.drv
05/04/2002	05:06p	18,042	kvd.drv
05/04/2002	05:06p	17,742	kwc.drv
05/04/2002	05:06p	18,342	laq.drv
04/25/1999	04:39p	20,429	mn.drv
05/04/2002	05:06p	17,742	ozy.drv
12/29/1998	01:14a	1,988	proxylst.txt
05/04/2002	05:06p	18,042	rxo.drv
04/20/1999	05:22p	26,649	sc.drv
05/03/1999	11:12p	30,741	sl.drv
08/22/2001	08:38p	562,176	Statistics.exe
01/07/1999	03:20a	90,112	Sysinfo.dll
10/18/2000	05:23p	22,016	TeamScan32.exe
05/04/2002	05:06p	18,342	tqe.drv
05/04/2002	05:06p	18,342	uig.drv
05/04/2002	05:06p	18,942	ujk.drv
04/18/1999	09:50p	29,100	unicodbag.txt
01/15/1999	02:54p	0	unicod_look
01/15/1999	02:54p	0	unicod_ready
01/15/1999	02:30p	24	users2.txt
01/06/1999	10:54p	95	wm.drv
05/04/2002	05:06p	18,942	wou.drv
05/04/2002	05:06p	18,342	zjr.drv
05/04/2002	05:06p	18,342	zyp.drv
	37 File(s)	1,211,202	bytes

Many of the files in this directory were actually scripts of some kind, several of them just copies

of another script. Eliot didn't recognize the scripting language, though it reminded him of C or Perl code. He wondered what the mechanism for launching the scripts might be.

The file lb.exe seemed to be at the root of everything else that happened on the server. Eliot had decided to try and determine what this file was, as well as its purpose. Now the diskette sat before him, and he really had no idea where to start. Opening the diskette in Windows Explorer simply showed the icon for the file "AZzZ Team" in neon green lettering. He'd done a Google search for any reference to this "team", and found nothing.

Eliot reached back into his memory of dealing with suspicious files on Unix systems. Unix came with a wide range of utilities for "looking at" files, so he figured that all he needed to do was find tools for Windows systems with similar functionality. However, he also knew that Windows files had different characteristics from Unix files, especially executables and dynamic linked libraries (DLLs). So he'd need to find tools that revealed those aspects of the file as well.

His first thought was to fall back on the old standby, strings.exe. Eliot found a convenient copy of the utility at the [SysInternals](#) Web site, and ran it against lb.exe using the following command:

```
C:\lb_file>strings -a -n 4 lb.exe > strings4.dat
```

The command resulted in a .dat file almost 50K in size. That's quite a bit of data to go through, Eliot thought. He opened the file in Notepad and the first thing he saw at the top of the page was:

```
This program must be run under Win32
UPX0
UPX1
UPX2
$Id: UPX 0.62 Copyright (C) 1996-1999 Laszlo Molnar & Markus Oberhumer $
$Id: NRV 0.54 Copyright (C) 1996-1999 Markus F.X.J. Oberhumer $
$License: NRV for UPX is distributed under special license $
```

"Ah, a clue," Eliot thought with relief. This indicated that the executable file he was working with was actually compressed with [UPX](#), an executable file compressor. However, the version was older than anything Eliot had seen available. Eliot checked out the [UPX site at SourceForge.net](#), and the [oldest version](#) he could find was 1.0. He downloaded it and tried to run it against lb.exe to get a file listing, but instead got an error message referring to incompatible versions.

Not to be deterred by this apparent dead-end, Eliot continued browsing through the .dat file. Things went faster as he hit the "page down" key again and again, nonsense text scrolling by. He guessed that was to be expected: after all, the file was compressed and any usable data would be unreadable. When he finally reached the end of the .dat file, he found some readable text again:

```
<>Even the fool can break open it... But only clever will not do it...<> (c)
```

Ah, a message to the user, or anyone with the intestinal fortitude to open the file in a text editor. Interesting. A Google search on this phrase revealed nothing. Eliot was beginning to wonder if he'd ever get anywhere with this file.

Knowing that some individuals and companies who create programs for Windows will compile vendor, author, and version information into the resource section of an executable file (EXE, DLL, SYS, etc.), Eliot located a Perl script called [finfo.pl](#) that would extract that information, if present. For example, executable files provided by Microsoft contain this information. Running this script against lb.exe, Eliot didn't get any useful information. That was to be expected, he thought, since the file was compressed, but he had to try. Eliot felt that it was better to try something, regardless of the arguments against it, and document the result, rather than not try it.

With the clues he'd found so far, he decided that the best thing to do was to just execute the file on an isolated test system. He knew that he needed some means of "snapshotting" the system both before and after the installation, in order to determine what changes were made when the file was executed. Not only that, but he needed a way to quickly and accurately compare those snapshots. Less than an hour of searching led him to a utility called [InControl5](#). This utility does exactly what Eliot needed, and provides for output formats in HTML and comma-delimited text (.csv) for opening in Excel. But he also wanted to be sure that he'd covered all the bases, so he ran several other utilities on the system, including [service.pl](#), drivers.exe (from the Resource Kit), [pslist.exe](#), [listdlls.exe](#), [fport.exe](#), and netstat.exe. Eliot redirected the output of each of these commands to uniquely named files, so that he could run the commands again later, and compare the results.

So Eliot loaded lb.exe onto an isolated, non-networked test system, installed InControl5 from the incident response CD he'd created, and run the first half of the two-stage process. He also made sure to enable Process Tracking in the EventLog, in order to determine what processes ran and at

what privilege level. Once InControl5 completed its initial snapshot, he executed lb.exe and waited for about a minute. At first, there was quite a bit of drive activity, but once it quieted down, he completed InControl5's analysis process. He then re-ran the utilities he'd run previously in order to see if any new processes had been created, services installed, or ports opened.

Once he'd collected all of his data, Eliot opened up the Event Viewer to see what evidence the test had left behind. Exporting the contents of the Security EventLog to a .csv file that he could open in Excel made viewing the events much easier. He walked through the events and saw the event IDs of 592 for processes called lb.exe, statistics.exe, and TeamScan32.exe. Interestingly, the last two were executables that appeared in the directory listing he'd found on the Web server. Oddly enough, though, only statistics.exe appeared in the process listing obtained with pslist.exe. In fact, the output of fport.exe showed that this statistics.exe was using a port:

```
1028 Statistics -> 1053 TCP C:\Progra~1\Manual\WINNT\Statistics.exe
```

```
1028 Statistics -> 61080 TCP C:\Progra~1\Manual\WINNT\Statistics.exe
```

Since the test system was not on a network, Eliot figured that he had some time to peruse the data he'd collected already. One of the issues he'd heard about with new systems put on networks, and even honeypots, was that if they weren't employed and managed correctly, they could actually serve as an access point into the network. Eliot figured that the first place he would start was with the .csv output file from InControl5. He located the file, and double-clicked it, opening it in Excel. The first section he came across was the section that listed modifications to the Registry. Scrolling down, Eliot could see that keys had been added to the Registry, particularly:

```
HKEY_CURRENT_USER\Software\Mirc
```

```
HKEY_CLASSES_ROOT\ChatFile
```

```
HKEY_CLASSES_ROOT\irc
```

With few exceptions, these were the keys that had been added, with subkeys being added beneath them. Eliot knew from the reading he'd done that Trojan applications and other malware usually maintained persistence by creating an entry in the ubiquitous "Run" key, located under HKEY\_LOCAL\_MACHINE\Software\Microsoft\CurrentVersion\Windows. Scrolling through the spreadsheet, he found the key he was looking for. Looking to the right of that key in the same row, he found the data that had been added to that key:

ScanDetect32

c:\progra~1&gt;manual\winnt\statistics.exe

There it was, the Registry entry that would allow the application to be run every time the system was restarted. Eliot knew that this information was very useful, as it justified the time he put into Perl scripts that would remotely check the contents of that key on all systems in the domain, as well as update the permissions on that key. As an aside, Eliot decided to use the [keytime.pl](#) Perl script to verify the LastWrite time for the "Run" key. The response that the script returned corresponded to the time that he'd run the test, just as he'd expected.

Scrolling further down through the spreadsheet, Eliot reviewed the modifications that had been made to the file system. There wasn't anything unexpected there; in fact, every entry in the spreadsheet corresponded directly to the directory listing he'd obtained from the Web server.

At this point, Eliot knew from the Web server logs that lb.exe had been copied to the server using the directory transversal exploit to launch tftp.exe. From there, lb.exe was executed, and the Security EventLog entries from both the Web server and the test system showed the processes that were created when this happened. However, from what he'd been able to determine, UPX was simply a file compressor. Any executable compressed with UPX could be executed, or decompressed...but not decompressed and then immediately executed. Further research on the Internet led Eliot to believe that the version of UPX used to compress the lb.exe file was not only an older version, but had also been modified in some way to launch files in a specific sequence. During several hours of searching on Google, he kept coming across references to the [MyParty](#) worm. Eliot figured that the only other possibility would be an executable binder of some kind, like [EliteWrap](#), but he couldn't find any evidence of any [other binder](#) being used. Without any evidence to support his theory, Eliot figured that it was better left unsaid.

So, what to do next, he thought. He'd launched the file, and now had a bunch of files and a process to work with. Eliot figured that he would try some of the same things on the new files as he'd done on the original file, so he ran strings.exe and finfo.pl against statistics.exe and teamscan32.exe. The output from strings wasn't all that interesting, but it did show him that teamscan32.exe had been compressed with a newer version of UPX. Eliot used a more up-to-date version of UPX to decompress the file, and then reran strings.exe and finfo.pl against the resulting file.

The finfo.pl output from statistics.exe showed the following:

OriginalFilename	mirc32.exe
------------------	------------

```
InternalName          mIRC32
FileDescription       mIRC
FileVersion           5.82
Language              English (United States)
CompanyName           mIRC Co. Ltd.
LegalCopyright        Copyright © 1995-2000 mIRC Co. Ltd.
ProductName           mIRC
ProductVersion        5.82
```

The `finfo.pl` output from the decompressed `teamsan32.exe` revealed the following:

```
OriginalFilename     hidewndw.exe
InternalName         HideWindow
FileDescription      Hides/Reveals application windows
FileVersion          1.43
Language             English (United States)
CompanyName          Adrian Lopez
LegalCopyright       Copyright © 1996 Adrian Lopez; All rights reserved.
```

Now Eliot knew that he was getting somewhere. He knew that changing the names of files on a Windows system was an excellent method for hiding the file, and that's what had been done in this case. Anyone doing a Web search on a process or file called "statistics.exe" would find information about a database utility. A search for "teamsan32.exe" didn't reveal anything at all. However, by delving into the resource section of the file and retrieving the data placed there by the original author of the program, Eliot was lucky enough to find something very useful. He located the [author's Web site](#), and understood why the process only seemed to run for a short time: once the process was correctly invoked, it did its job (i.e., hide the IRC client window) and disappeared. Eliot then did a search for both applications, and found that this [wasn't the first instance](#) in which they had been used together to create a Trojan application.

Eliot now figured that he had a pretty good idea of what of the `lb.exe` file was all about. He was still in the dark about how the `statistics.exe` process had been launched by the `lb.exe` process, and he still had no idea what to call this particular IRC bot. During one of his Web searches for utilities, he'd run across a tool called [pmdump.exe](#) that would dump a process's memory contents to a file, and decided to run it against the `statistics.exe` process. What he ended up with was a 17.7 MB file, far too much to parse through by hand...so he ran `strings.exe` against the output of `pmdump`, and got a 1.55 MB file, which was a little easier to handle. He opened the output of

strings.exe in UltraEdit and started walking through the output a page at a time. He saw a lot of references to files and Registry keys, as well as the contents of scripts he'd seen in the WINNT directory along with statistics.exe and teamscan32.exe. He also saw something unusual:

```
Superkhaledfragilisticexpialidocious
```

Interesting, Eliot thought. Maybe someone had a sense of humor. However, given what he'd seen so far, both in the data he'd collected and in the research he'd done, it was pretty clear that this bot was controlled via an IRC channel. Someone would issue commands on the channel, and any bots connected to the channel would launch the appropriate scripts on the system. He also found e-mail addresses, and by doing a search using the IP address found in the output of netstat.exe (run after lb.exe was executed), he also found references to the IRC server the bot was trying to connect to. There were also some references to IRC channels:

```
#Russian_Cafe  
#russian_[a&m]_team  
#Russian_Love
```

More searches for information regarding "russian" netted Eliot with the following:

```
titlebar russiantopz  
    .run TeamScan32.exe "mIRC32 russiantopz"
```

The above looked like an excerpt from one of the scripts he'd seen. It looked as if this bot had originated in Russia, and was called "russiantopz". That was very interesting, Eliot thought, because his searches for a modified version of UPX mentioned the MyParty worm, which originated in Russia, as well.

By now, Eliot figured that he'd found out just about all he could about this IRC bot. He knew it was an IRC bot, and where it was from. He wasn't interested in parsing through each script to determine exactly what each one did, he wanted to get an e-mail sent off to Elizabeth to let her know what he'd found. He'd also have to talk to the network engineers about tracking outbound traffic to IRC servers. And thanks to this little exercise, he was confident that he had the necessary tools and skills to take a look at suspicious files and processes and determine their true nature. Between tools like [procdmp.pl](#) and those that he'd used to "look into" the statistics.exe file and process, he figured that he'd had just about all of the bases covered. Now, he thought, to burn them all to CD...

## Conclusion

As the No Stone Unturned series has shown, many of the basic troubleshooting, and even advanced "live" forensics techniques used in the Linux/\*nix world are applicable to the Windows world. Due to differences in architectures, some modifications or additional measures are required, but for the most part, the same principles apply.

### Relevant Links

[No Stone Unturned](#)

*H. Carvey*

[Reverse Engineering Malware](#)

*Lenny Zeltser*

[Hide Window 1.43](#)

*Adrian Lopez*

[PowerBOT Writeup](#)

*Dave Dittrich*

[Privacy Statement](#)

Copyright 2006, SecurityFocus