Global Information Assurance Certification Paper

Binary Analysis, Forensics and Legal Issues

Michael Wyman
GCFA Version 1.3 Practical
SANS CDI San Antonio TX
January 25-30 2003

Analysis of an Unknown binary

For the initial analysis, I used a Dell Laptop running the Redhat 7.3 Linux operating system, kernel version 2.4.18-3. A binary called binary_v1.3.zip was downloaded from The Sans Institute. The zip file was downloaded to a USB flash device commonly known as a "pen drive". Using a pen drive makes it easier to move the file between computers and operating systems.

To make it even easier to work with the binary, I borrowed a trick from Greg Owen's forensics practical <http://www.giac.org/practical/Greg_Owen_GCFA.zip>, and created a file system within a file, which was mounted using the loop option, making it appear as a hard drive partition. The noatime option was used in the mount command to ensure that the access times of the files were not modified. The commands used are shown below:

```
[root@localhost root]# cd /mnt/sda1
[root@localhost sda1]# dd if=/dev/zero of=vault.fs bs=1024 count=5000
5000+0 records in
5000+0 records out
[root@localhost sda1]# losetup /dev/loop1 vault.fs
[root@localhost sda1]# mkfs.ext3 /dev/loop1
mke2fs 1.27 (8-Mar-2002)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1256 inodes, 5000 blocks
250 blocks (5.00%) reserved for the super user
First data block=1
1 block group
8192 blocks per group, 8192 fragments per group
1256 inodes per group

Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 32 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[root@localhost sda1]# mkdir /mnt/vault
[root@localhost sda1]# mount -t ext3 -o loop,noexec,noatime /mnt/sda1/vault.fs /mnt/vault
```

The first command, cd /mnt/sda1, simply changes the working directory to the mount point of the pen drive. Next, the “dd” command was given to create a 5000 kb file full of zeros. Zeros were written to the file to ensure that all previous data occupying the same space on the pen drive, was effectively erased. Next, the command “losetup /dev/loop1 vault.fs” was issued to make the vault.fs file appear as a filesystem to the operating system. Finally, the filesystem was created within the vault.fs file by issuing “mkfs.ext3 /dev/loop1”. Once the vault.fs file contained a valid filesystem, it could be mounted using the mount command specifying the noexec and noatime options. This was done to ensure that the access times of the files examined were not altered, and to prevent the accidental execution of any programs that may reside on mounted filesystem.

Binary Details:

Once the vault was mounted, binary_v1.3.zip was copied to the vault and examined by executing the unzip command with the "-v", or verbose option.

```
[root@localhost vault]# unzip -v binary_v1.3.zip
Archive:  binary_v1.3.zip
 Length   Method    Size  Ratio   Date   Time   CRC-32    Name
--------  ------  ------- -----   ----   ----   ------    ----
   26793  Defl:N     5567  79%  02-20-03 12:45  d185fd18  target2.exe
--------          -------  ---                            -------
   26793             5567  79%                            1 file
```

Unfortunately, no MD5 checksum was included in the zip file. It would have been nice to have a MD5 checksum of the file as found on the compromised system to ensure the files were identical. The zip archive contains a file called target2.exe dated 2/20/03 with a time of 12:45. The file is 26793 bytes expanded.

Next, the file was extracted by invoking unzip with the "-X" option to preserve the Userid and Groupid of the file. The user and group are listed as root (UID 0). One explanation for this could be that the person who zipped up the file may have inadvertently modified the file attributes. Another possibility would be that the file came from a Windows 95/98/ME system, which always sets the UID and GID to 0. To examine the MAC times, the stat command was run against the extracted file, target2.exe. We see below that the Access and Modify times are the same. We can not determine when this file was last executed from the evidence we have seen so far. It is possible that the file was never executed.

```
[root@localhost vault]# unzip -X binary_v1.3.zip
Archive:  binary_v1.3.zip
  inflating: target2.exe
[root@localhost vault]# stat target2.exe
  File: "target2.exe"
  Size: 26793           Blocks: 56         IO Block: -4611692409338720256 Regular File
Device: 702h/1794d      Inode: 13          Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: Thu Feb 20 12:45:48 2003
Modify: Thu Feb 20 12:45:48 2003
Change: Sat Jun 21 03:06:36 2003
```

The .exe extension strongly indicates this is a Windows executable and the file command confirms this fact.

```
[root@localhost vault]# file target2.exe
target2.exe: MS-DOS executable (EXE), OS/2 or MS Windows
```

The MD5 checksum of the expanded file is:

```
$ md5sum.exe target2.exe
848903a92843895f3ba7fb77f02f9bf1 *target2.exe
```

The next step is to run the strings command, which shows 212 lines of output, most of which is garbage. The interesting output is shown below:

```
impossibile creare raw ICMP socket
RAW ICMP SendTo:
======================== Icmp BackDoor V0.1 =
========= Code by Spoof. Enjoy Yourself!
 Your PassWord:
loki
cmd.exe
 Exit OK!
Local Partners Access
Error UnInstalling Service
Service UnInstalled Sucessfully
Error Installing Service
Service Installed Sucessfully
Create Service %s ok!
CreateService failed:%d
Service Stopped
Force Service Stopped Failed%d
The service is running or starting!
Query service status failed!
Open service failed!
Service %s Already exists
Local Printer Manager Service
smsses.exe
Open Service Control Manage failed:%d
Start service successfully!
Starting the service failed!
starting the service <%s>...
Successfully!
Failed!
Try to change the service's start type...
The service is disabled!
Query service config failed!
```

Program Description & Forensic Details

The program appears to be ICMP backdoor v0.1 if we believe the alleged author's tag line. An ICMP back door communicates via the ICMP protocol, normally used to transmit network errors, announce network congestion, and assist troubleshooting (LIUtilities). One common use of the ICMP protocol is found in the command "ping". Ping is primarily used to test basic network connectivity between two hosts. A ping or more formally, an ICMP echo request, can be issued to a remote host. If the remote host is reachable, it normally will reply with a "pong" known as an ICMP echo reply. For security reasons, many companies block incoming ICMP echo requests. However, many companies do not block incoming ICMP echo replies.
An ICMP backdoor can communicate via ICMP echo replies, thus passing through some firewalls. The traffic might slip by a network administrator unnoticed and if noticed, may be inappropriately classified as harmless. For a program like this to work, you need a "client" that

can convert your commands into ICMP network packets, and a "server" that can receive the ICMP packets and convert them back into commands.

Examining the output of the strings command, we see the phrase "RAW ICMP SendTo:". This looks like a prompt for a destination address. Two lines later we see an apparent prompt for a password, "Your Password:", with the word "loki" appearing below. Loki was one of the first ICMP backdoor programs. We also see cmd.exe in the output. Cmd.exe is the 32 bit Windows command prompt used in Windows NT/2000/XP. It looks like target2.exe is accessing cmd.exe. which will allow an attacker to execute programs on a compromised system.

```
RAW ICMP SendTo:
======================== Icmp BackDoor V0.1 =
========= Code by Spoof. Enjoy Yourself!
 Your PassWord:
loki
cmd.exe
 Exit OK!
```

It also appears to be installing a service, based on all of the starting and stopping of services we see mentioned. The Printer Manger Service is listed by name, perhaps it is modifying or claiming to be the printing service.

We also see smsses.exe listed. This may be the name of the service under which the program runs. This would be a good name to choose because of its similarity to a legitimate service called smss.exe, that controls a number of critical functions, including starting the windows logon process and loading the kernel for the Win32 subsystem (Liutilities).

The location in which the program looks for smsses.exe gives further clues about the operating system(s) on which it will execute. Viewing target2.exe in the freely available Hex Editor 2.0 http://www.hhdsoftware.com, we can see the following:

```
...\.w.i.n.n.t.\
.s.y.s.t.e.m.3.2
.\.s.m.s.s.e.s..
.e.x.e.....▌ÿSM
```

The winnt folder contains the operating system on Windows 2000, NT4.0 and NT3.51 systems. Windows XP keeps it's operating system in a folder called "windows".

The screen capture below shows that the program calls winnt\system32\reg.exe.

```
.....
.1..\.w.i.n.n.t
\.s.y.s.t.e.m.3
2.\.r.e.g...e.x
e.....▌ÿSMB¢..
```

The Registry Console Tool (reg.exe) is a utility that enables command line changes, additions and deletions to the Windows Registry. The Windows registry is really just a database that contains the computers configuration. Reg.exe is included with the Windows NT4.0 resource kit, and the Windows 2000 support tools. We can conclude that target2.exe is targeted towards Windows NT 4.0 or Windows 2000. At this point, we still don't know if target2.exe is a client or server.

The next step was to observe the binary in action. I copied the binary to a Dell Cpia laptop running Windows 2000 service pack 3 and assigned it an IP address of 192.168.1.103. The Windows computer was isolated behind a router and was monitored by another computer running Linux Redhat 7.3 attached to a network tap. The monitoring computer's network interface was brought up without an IP address, and issued the command tcpdump 'host 192.168.1.103'.

The filemon utility from http://www.sysinternals.com was used to monitor the file access of the suspected trojan. Filemon has the ability to monitor file system activity in real-time and save the results. The figure below shows the output of the program. This is the output of the second run of target2.exe. During the first run, I noticed that target2.exe failed to find 3 files that were missing, namely msvcp60.dll, mfc42loc.dll, and target.exe.Local. I downloaded the missing dlls from http://www.dll-files.com/ and installed them into the system32 directory. Next I copied the target.exe to the system32 directory and renamed it to target.exe.Local, and executed target2.exe again.

403	9:10:10 PM	cmd.exe:884	OPEN	C:\target2.exe	SUCCESS Options: Open Access: Execute	
404	9:10:10 PM	cmd.exe:884	CLOSE	C:\target2.exe	SUCCESS	
405	9:10:10 PM	cmd.exe:884	QUERY INFORMATION	C:\	SUCCESS Attributes: DHSA	
406	9:10:10 PM	target2.exe:668	OPEN	C:\	SUCCESS Options: Open Directory Access: Traverse	
407	9:10:10 PM	target2.exe:668	QUERY INFORMATION	C:\WS2_32.dll	FILE NOT FOUND	Attributes: Error
408	9:10:10 PM	target2.exe:668	QUERY INFORMATION	C:\WS2_32.dll	FILE NOT FOUND	Attributes: Error
409	9:10:10 PM	target2.exe:668	QUERY INFORMATION	C:\WINNT\System32\WS2_32.dll	SUCCESS	Attributes: A
410	9:10:10 PM	target2.exe:668	OPEN	C:\WINNT\System32\WS2_32.dll	SUCCESS Options: Open Access: Execute	
411	9:10:10 PM	target2.exe:668	CLOSE	C:\WINNT\System32\WS2_32.dll	SUCCESS	
412	9:10:10 PM	target2.exe:668	QUERY INFORMATION	C:\WS2HELP.DLL	FILE NOT FOUND	Attributes: Error
413	9:10:10 PM	target2.exe:668	QUERY INFORMATION	C:\WS2HELP.DLL	FILE NOT FOUND	Attributes: Error
414	9:10:10 PM	target2.exe:668	QUERY INFORMATION	C:\WINNT\System32\WS2HELP.DLL	SUCCESS	Attributes: A
415	9:10:10 PM	target2.exe:668	OPEN	C:\WINNT\System32\WS2HELP.DLL	SUCCESS Options: Open Access: Execute	
416	9:10:10 PM	target2.exe:668	CLOSE	C:\WINNT\System32\WS2HELP.DLL	SUCCESS	
417	9:10:10 PM	target2.exe:668	QUERY INFORMATION	C:\MFC42.DLL	FILE NOT FOUND	Attributes: Error
418	9:10:10 PM	target2.exe:668	QUERY INFORMATION	C:\MFC42.DLL	FILE NOT FOUND	Attributes: Error
419	9:10:10 PM	target2.exe:668	QUERY INFORMATION	C:\WINNT\System32\MFC42.DLL	SUCCESS	Attributes: A
420	9:10:10 PM	target2.exe:668	OPEN	C:\WINNT\System32\MFC42.DLL	SUCCESS Options: Open Access: Execute	
421	9:10:10 PM	target2.exe:668	CLOSE	C:\WINNT\System32\MFC42.DLL	SUCCESS	
422	9:10:10 PM	target2.exe:668	QUERY INFORMATION	C:\MSVCP60.dll	FILE NOT FOUND	Attributes:

Error
423 9:10:10 PM target2.exe:668 QUERY INFORMATION C:\MSVCP60.dll FILE NOT FOUND Attributes: Error
424 9:10:10 PM target2.exe:668 QUERY INFORMATION C:\WINNT\System32\MSVCP60.dll FILE NOT FOUND Attributes: Error
425 9:10:10 PM target2.exe:668 QUERY INFORMATION C:\WINNT\system\MSVCP60.dll SUCCESS Attributes: A
426 9:10:10 PM target2.exe:668 OPEN C:\WINNT\system\MSVCP60.dll SUCCESS Options: Open Access: Execute
427 9:10:10 PM target2.exe:668 CLOSE C:\WINNT\system\MSVCP60.dll SUCCESS
428 9:10:10 PM target2.exe:668 QUERY INFORMATION C:\target2.exe.Local FILE NOT FOUND Attributes: Error
429 9:10:11 PM target2.exe:668 QUERY INFORMATION C:\WINNT\System32\MFC42LOC.DLL FILE NOT FOUND Attributes: Error
430 9:10:11 PM target2.exe:668 QUERY INFORMATION C:\WINNT\System32\MFC42LOC.DLL FILE NOT FOUND Attributes: Error
431 9:10:26 PM target2.exe:668 CLOSE C:\ SUCCESS

Test Results:

The first thing we see on line 403 is target2.exe being launched from the command line. Throughout the output, we see target.exe first searching in the directory in which it resides. In this case, it is simply at the root level of the C drive. The optimal place to locate this trojan would be in %windir%\system32, because most of the files it is looking for reside in the system32 directory. This would also have the benefit of making the trojan harder to detect, because the system32 folder is generally not viewed by many end users. Starting on Line 407 we see the trojan trying to open WS2_32.dll.

Looking up this dll at http://www.liutilities.com/products/wintaskspro/dlllibrary, we find that it is used for network connections. The functions that WS2_32.dll uses for establishing a network socket are contained in the next dll that is opened, WS2HELP.dll (line 414).

The next file opened is MFC42.DLL on line 419. MFC or Microsoft Foundation Class, is an object oriented application framework that provides a relatively simple programming interface. Many newer programs use the MFC (Dll Zone).

Continuing on line 422, the trojan attempted to open was msvcp60.dll, which is the Microsoft C Runtime Library. This library contains the standard C functions, such as printf. Thus far, it appears that target2.exe is attempting to initialize a network socket.
The last file opened was MFC42LOC.DLL. MFC42LOC is a localization file for the Microsoft Foundation Class. This file should only be found on non-English systems (Schwartz). I didn't need to copy this mfc42loc after all.

Running the Filemon Utility showed what files were being accessed however it is also useful to see what the process was doing in memory. The Process Explorer, gives us real-time information about what the process is doing in memory. Process Explorer is freely available from SysInternals http://www.sysinternals.com/ntw2k/freeware/procexp.shtml.

Process: target2.exe Pid: 896

Base	Size	MM	Description	Version	Time	Path
0x240000	0x16000	*			12/7/1999 7:00 AM	C:\WINNT\system32\unicode.nls
0x260000	0x2F000	*			12/7/1999 7:00 AM	C:\WINNT\system32\locale.nls
0x290000	0x41000	*			12/7/1999 7:00 AM	C:\WINNT\system32\sortkey.nls
0x2E0000	0x4000	*			12/7/1999 7:00 AM	C:\WINNT\system32\sorttbls.nls
0x300000	0x2000	*			12/7/1999 7:00 AM	C:\WINNT\system32\ctype.nls
0x400000	0x6000				2/20/2003 12:45 PM	C:\target2.exe
0x6C370000	0xF2000		MFCDLL Shared Library - Retail Version	6.00.8665.0000	12/7/1999 7:00 AM	C:\WINNT\system32\mfc42.dll
0x75020000	0x8000		Windows Socket 2.0 Helper for Windows NT	5.00.2134.0001	12/7/1999 7:00 AM	C:\WINNT\system32\ws2help.dll
0x75030000	0x14000		Windows Socket 2.0 32-Bit DLL	5.00.2134.0001	12/7/1999 7:00 AM	C:\WINNT\system32\ws2_32.dll
0x77D40000	0x6F000		Remote Procedure Call Runtime	5.00.2193.0001	12/7/1999 7:00 AM	C:\WINNT\system32\rpcrt4.dll
0x77DB0000	0x5A000		Advanced Windows 32 Base API	5.00.2191.0001	12/7/1999 7:00 AM	C:\WINNT\system32\advapi32.dll
0x77E10000	0x65000		Windows 2000 USER API Client DLL	5.00.2180.0001	12/7/1999 7:00 AM	C:\WINNT\system32\user32.dll
0x77E80000	0xB6000		Windows NT BASE API Client DLL	5.00.2191.0001	12/7/1999 7:00 AM	C:\WINNT\system32\kernel32.dll
0x77F40000	0x3C000		GDI Client DLL	5.00.2180.0001	12/7/1999 7:00 AM	C:\WINNT\system32\gdi32.dll
0x77F80000	0x79000		NT Layer DLL	5.00.2163.0001	12/7/1999 7:00 AM	C:\WINNT\system32\ntdll.dll
0x78000000	0x46000		Microsoft (R) C Runtime Library	6.01.8637.0000	12/7/1999 7:00 AM	C:\WINNT\system32\msvcrt.dll
0x780C0000	0x61000		Microsoft (R) C++ Runtime Library	6.00.8168.0000	3/27/2001 4:11 PM	C:\WINNT\system\MSVCP60.DLL

The output from Process Explorer shows that target2.exe first accesses the memory locations for five different National Language Support (.nls) files. NLS files provide the language to ASCII mapping necessary for a program to run. The rest of the output mirrors what was shown in the Filemon output, that is, we see an indication that a network socket will be opened. Process Explorer shows the accessing of the GDI Client DLL, gdi32.dll. This dll handles a number of graphics functions. This is consistent with the program attempting to write something to the screen.

Program Identification:

The program has a very unique characteristic, namely the Italian text "Impossibile creare raw ICMP socket". This line was searched using the Google search engine yielding a number of pages that all pointed to the same header file; icmp_tunnel.h, written by Dark Schneider. This header file contained the phrase "Impossibile creare raw ICMP socket", and was probably where the target2.exe exploit code originated. It can be found in the "Butchered from Inside" (Bfi), Italian online "security" magazine. http://www.s0ftpj.org/bfi/online/bfi7/bfi07-13.html

A header file is a collection of information that the functions of a program need. In a C program, one includes this information using a "#include" statement at the beginning of the program. Unfortunately, I could not find the actual code used in target2.exe, and it appeared as though target2.exe did not properly execute. Running the command netstat -an (list all network connections and listening ports in numerical form), did not show anything suspicious. Had an ICMP backdoor been present on the system, we would have expected to see something like

"raw 0.0.0.0: 0.0.0.0:0 LISTENING"

Conclusion: We see that the target.exe program ran for 16 seconds when executed. The programs calling of the WS2_32 networking libraries, and the text strings we found in the binary claiming to be an ICMP back door are consistent with the notion that this is indeed a backdoor. This appears to be the client component of a client-server based trojan. One would need a special server running to communicate with this trojan because of the use of ICMP echo replies. Typical client/server applications like Telnet do not use ICMP to communicate, rather they use the TCP protocol. We do not see target2.exe attempting to open C:\winnt\System32\smsse.exe or C:\winnt\system32\reg.exe in the Filemon output, despite the fact that we find both of these programs listed in the strings output obtained from target2.exe. It is possible that these two programs (smess.exe & reg.exe) are a necessary component of the server portion of the backdoor. That is, the client connects to a server and executes one or both of these programs.

Legal Implications.

We cannot prove that this program was executed on the host system based on the available evidence. If we could get a complete image of the compromised system, we could build a timeline. The obtained timeline could be compared to the filemon output of our test system and we might be able to show target2.exe's accessing the same files, in the same order, as our test system. This however would be unlikely, because the system in question probably continued to be in use for sometime after the alleged running of target2.exe.

We probably can't even prove that our companies acceptable use policy was violated, unless we can prove that he put the program on a system he wasn't supposed to access. Here is an excerpt from the acceptable use policy:

> Some examples of inappropriate uses of Computer Resources include:
> Attempting to break into or monitor any Computer Resource without proper authorization.
> Accessing confidential or sensitive information on computer resources without authorization

So, I would have to prove that he put the server component of this backdoor on a computer he was not supposed to have access to, then accessed the computer and viewed confidential or sensitive documents.

Interview Questions:

Set the tone: Narrative – *We both work for the same company. I'm working on my day off, in a hurry to leave and don't really want to do an in depth investigation of anything. I'm not out for blood, I just want to get things wrapped up.*

Talking to our suspect, Mike:

Mike, I need your help, I was supposed to be off today, but before I can go home I have to get this thing resolved. The Intrusion detection team found some strange traffic on the network. Joe, the Sysadmin found this program on a computer, I'm not sure what it does, but it looks like it might be an alternative shell-like program, similar to Telnet or SSH. Who knows it may even be more secure than telnet.

1. Mike, the program is called target2.exe. Like I said, seems like a telnet program of some sort. Have you ever used Telnet before? *(Assume I know he has. I want him to admit something innocuous.)*. Telnet's not very secure, especially on Windows, this (target2.exe) looks better.

2. Well, we are still trying to figure out where this originated. The Intrusion Detection folks need more time to go through the logs. Early indications are that it was coming from a computer you might have been working on. It's always better if these things can be solved before they get escalated. It starts to turn into a big deal when you have to get more people working on the problem. I'm hoping you can help me nip this in the bud. Can you think of any reason why someone would run this program?

Just to see if he will come up with a reason, maybe even the reason given in question 1.

3. Have you ever seen this program before?

4. Look, I just have to know, so we can stop this before it gets out of hand. I just want this solved. Did you install this program?

5. *If no*: Do you have any idea who might want to install something like this?
   If Yes: Can you tell me what systems you connected to?

References:

Connected: An Internet Encyclopedia. Ed. Baccala, Brent. Apr. 1997. Freesoft. 15 Aug. 2003. <http://www.freesoft.org/CIE/Topics/81.htm>

The DLL Zone. Fortune City. 18 Aug. 2003. <http://www.fortunecity.com/skyscraper/fortune/570/mfc42.html>

LIUtilities Online. LIUtilities Inc. 15 Aug. 2003 <http://www.liutilities.com/products/wintaskspro/processlibrary/smss/>

Schwartz, David. Redistributing Microsoft Visual C++ 6.0 Applications. MSDN. Aug. 2000. Microsoft. 18 Aug. 2003. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvc60/html/redistribvc6.asp>

Legal Issues

A. What points if any can you provide to the law enforcement officer over the phone during the initial contact?

Because my company offers services to the public, the Electronic Communications Privacy Act (ECPA) applies. The ECPA 18 U.S.C. § 2702 covers the rules for voluntarily disclosing customer records and communications. Generally, it restricts companies that provide services to the public from disclosing customer data, particularly to government entity's.

There are several exceptions. I am allowed to voluntarily disclose data to Law Enforcement; if I inadvertently stumble on communications that appear to be related to the commission of a crime, or if I believe that there is a danger of death of serious physical injury to another person.

Additionally, I may disclose customer information if I have permission from the customer, including consent given by clicking on a banner or agreeing to a user agreement, or if I need to protect my own property (i.e. If I need to stop disruption or hacking of my computers). Assuming none of these exclusion conditions are met, I can not disclose much information to law enforcement at this time.

I can tell the Law Enforcement official that I have a record of a valid account being "dialed in" during the time of the incident.
I can also tell Law Enforcement that after reviewing my log files, I have found no evidence of hacking during this time period, and I was therefore unable to determine if the attack came from my system or from upstream.

B. What must the law enforcement officer do to ensure you preserve this evidence if there is a delay in obtaining any required legal authority?

The law enforcement official can issue a retention order, requiring that I preserve all relevant log files for a period of 90 days. (Morris p5). The retention order is made possible by Title 18 of the US Code, Section 2703(f) , that states;

"A provider of wire or electronic communication services or a remote computing service, upon the request of a governmental entity, shall take all necessary steps to preserve records and other evidence in its possession pending the issuance of a court order or other process"

The request to preserve the evidence can be done over the phone, but the Department of Justice (DOJ) suggests that it be done by fax, or email in order to avoid misinterpretations.

The DOJ publication Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations, phrases it like this:

> "There is no legally prescribed format for § 2703(f) requests. While a simple phone call should therefore be adequate, a fax or an e-mail is better practice because it both provides a paper record and guards against miscommunication."

A sample of the suggested DOJ preservation of evidence letter can be found here: http://www.cybercrime.gov/s&sappendix2002.htm#_C_

C: What legal authority, if any, does the law enforcement officer need to provide to you in order for you to send him your logs?

Law enforcement would be required to obtain a court order to get access to my logs. A court will issue an order if the agent can show "articuable facts" that make it reasonable to believe that the information requested is "relevant and material to an ongoing criminal investigation" (Morris p4). The court order would allow them to see everything in the log files related to the alleged perpetrator, such as source and destination IP addresses, account names, dates and time spent online. If the investigator only needs the identity, name, address and phone billing records, he/she could use a subpoena. This could also allow the investigator to read already opened emails, however unopened emails require a search warrant.

D: What other "investigative" activity are you permitted to conduct at this time?

Assuming I have no banner's or user agreement in place, there is not much that I can do. I can search my own log files to ensure the integrity of my system. I could also find out what type of attack was perpetrated against the government, and then watch outbound traffic for that type of activity. For example, if the hacker used an rpc exploit, I could use my provider exemption to monitor rpc traffic on the system as a whole, and look for suspicious activity. I could not single out one user and monitor their transmissions because that would basically be a wiretap and violation of US Code, Title 18 Section 2511. This is especially true if I am monitoring the userid as the result of information provided by Law Enforcement.

E: How would your actions change if your logs disclosed a hacker gained unauthorized access to your system at some point, created an account for him/her to use, and used THAT account to hack into the government system?

If the hacker gains unauthorized access to my system and creates a new account to hack from, I am still somewhat limited in what I can do because of the contractual agreement that exists between the user and my company.

However, the USA Patriot Act of 2001 added an amendment to subsection 2703(c)3 that states "service providers do have the statutory authority to disclose non-content records to protect their rights and property" (Morris) This means that I can turn over the log files to Law Enforcement in this particular case.

References:

Interception and disclosure of wire, oral, or electronic communications prohibited. 18 US Code. Sec. 2511. http://www4.law.cornell.edu/uscode/18/2511.html

Morris, Daniel. Tracking a Computer Hacker. U.S. Department of Justice. 10 July 2001. <http://www.usdoj.gov/criminal/cybercrime/usamay2001_2.htm>

The Electronic Communications Privacy Act. 18 US Code. Sec. 2701-2712. 1986. http://www4.law.cornell.edu/uscode/18/2701.html

Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations. U.S. Department of Justice. 02 July 2002. <http://www.cybercrime.gov/s&smanual2002.htm#_IIIC1_>

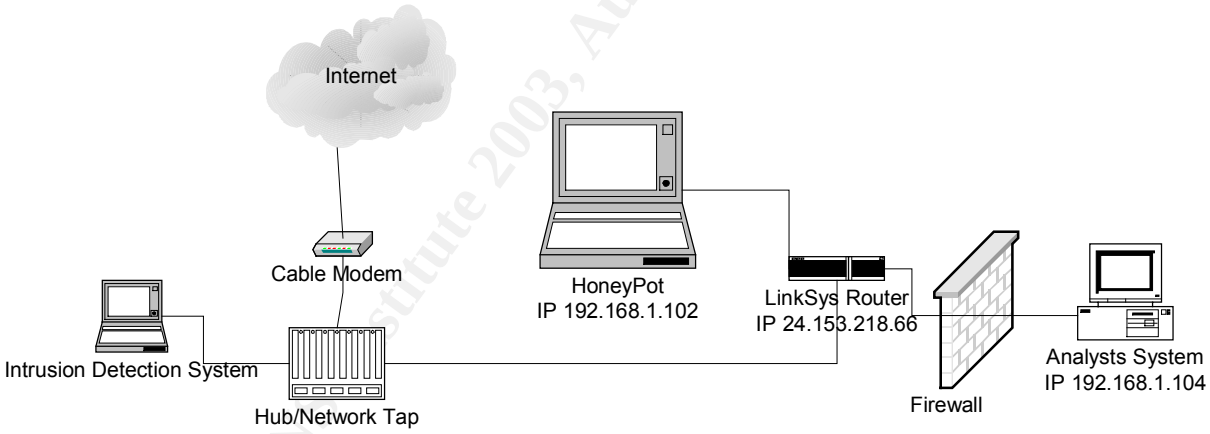
Forensics Analysis of Compromised System

Synopsis of Case Facts & Description of the System Analyzed:

An IBM laptop was connected to a cable modem router and deliberately put outside the firewall, to act as a honeypot. The computer was first prepared by 'wiping' the hard disk using the Wipe V2.0 utility by Tom Vier. This procedure ensured that no data would remain on the disk, giving the investigator a "clean slate" to work from. After wiping the disk, the system was loaded with the Windows 2000 Advanced server operating system and formatted with the NTFS file system.

Only 2048 kilobytes of the 5 gigabyte hard disk was partitioned and formatted. This was done because the investigator's computer only had 3 gigabytes of free hard disk space to hold the hard drive image of the honeypot. The default installation options were selected, as was the option to install Internet Information Server version 5.0. No application or operating system patches were installed.

The honeypot was monitored via another computer running the Linux Redhat Operating System version 7.3. The monitoring computer was connected to a network tap and the network interface was brought up without an IP number. This made the monitoring system invisible to the Internet.



The honeypot was placed outside the firewall and was scanned using a web based port scanner called "Shields Up" found at http://grc.com/intro.htm. This was done to verify that port 80 was exposed to the internet.

All network traffic was collected using tcpdump (http://www.tcpdump.org). The specific command was; tcpdump -i eth0 ' ! arp ' -w /home/tmp/traffic. The "! arp" directive was used to avoid recording the barrage of Address Resolution Protocol (arp) traffic generated by the service provider.

Unfortunately, a mistake was made when connecting the honeypot to the network. Tcpdump was not activated until after the honeypot had been connected to the network.
As a result, the tcpdump log does not contain a record of the actual system compromise.

The honeypot was disconnected from the network as soon as the system compromise was detected, by simply unplugging the network cable. The system was left running in order to collect an image of the volatile dynamic memory before system shutdown.

Hardware:

The computer system is a IBM Thinkpad 600 laptop, with a 6 gigabyte hard drive, 128 megabytes of memory, built in USB port, sound card and an Internal CDrom drive and a 3com Megahertz PCMCIA network card.

Tag #s	**Description**
Tag 001	IBM Model 600 Thinkpad Laptop Serial # 78-HBI70
Tag 002	IBM Travelstar hard drive. Model DADA-25120 Size: 5120 Megabytes. Serial #: 11s03l560z1mf98w

Image Media.

The memory was the first thing imaged. At this point I did not want to re-connect the computer to the network, so I attached a 128mb USB 'pen' drive to the compromised system, inserted the SANS Forensics Course v1.6 CD, and issued the command dd if=\\.\PhysicalMemory conv=noerror | gzip -c| dd of=C:\mem.img.gz. Dd.exe, version 4.1 was run from the SANS Forensics Course v1.6 CD, because this version of dd has the ability to image a system's physical memory. Gzip was included in the middle of the command because the system's memory was 128 mb and the USB pen drive only had 124 megabytes of available space.

After obtaining the memory image, a hard shutdown was performed by removing the battery and power adapter.

The next step was to image the hard drive. The compromised system was booted from the CDrom drive using the Forensics and Incident Response Environment (F.I.R.E) version 4.0 bootable CDrom. F.I.R.E is a collection of forensics utilities as well as a bootable Linux environment that contains a number of useful statically compiled binaries.

Because of disk space limitations, only the partitioned portion of the hard disk was imaged. The fdisk v2.11 utility was run from the F.I.R.E CDrom to verify that there was still only one partition. The command fdisk –l was issued to list the partition table of the disk. The output of the fdisk utility below shows the single 2 gig partition.

```
Disk /dev/hda: 240 heads, 63 sectors, 839 cylinders
Units = cylinders of 15120 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1       277   2094088+   7  HPFS/NTFS
```

The program dd, version 3.16, was used to obtain the hard drive image and the output was sent to the investigators computer via the network transfer program Netcat version 1.10. The specific command was dd if=/dev/hda1 | nc 192.168.1.104 31337. This sent the entire contents of the hard drive partition to the Analysts' system. The Analysts' system was prepared to accept the image by issuing the command nc -l -p 31337 > /home/tmp/ibm.img. This command tells Netcat to listen (-l) on port (-p) 31337 and redirect the output to the file ibm.img.

To verify that the image was identical to the original partition, a program called md5sum version 2.0.14 was used to obtain the MD5 checksum. The Message Digest Number 5 (MD5), algorithm was invented by Ron Rivest, and is used to generate a 128 bit number to verify the integrity of a file. Each file will have a different MD5 checksum, even if a file differs from another file by only one character (MD5 Checksum Utility). In short, if an image has the same MD5 checksum as the original filesystem, one can be sure that they have an exact copy.

Below is the check sum of the transferred image:

```
[root@localhost image]# md5sum ibm.img
773934f7fd8a0ad37302144dc2c48944  ibm.img
```

Here is the checksum collected from the original filesystem:

```
[root@FIRE] sda1> md5 /dev/hda1
773934f7fd8a0ad37302144dc2c48944        /dev/hda1
```

We can see that the numbers are identical. The image is an exact copy of the compromised filesystem.

Media Analysis of System:

There are several pieces of evidence available for analysis that are summarized in the table below.

Source	Description	Location/Name
Investigator's Notes	Notes taken by the investigator while configuring, deploying and analyzing the honeypot.	On the analysis system. Name: notes.txt
Tcpdump Logfile	File containing all network traffic entering and exiting the honeypot,	On the analysis system. Name: sunday.tcp

Source	Description	Location/Name
	excluding arp broadcasts.	
Windows Log Files	Windows System & event logs.	Windows Log files: C:\Winnt\system32\config\AppEvent.evt C:\Winnt\system32\config\SecEvent.evt C:\Winnt\system32\config\SysEvent.evt Filenames: Unchanged
IIS Log Files	Log files generated by Internet Information Server (IIS).	C:\Winnt\system32\LogFiles\W3SVC2\ex030727.log C:\Winnt\system32\LogFiles\W3SVC2\ex030727.log FileNames: Unchanged
Record of running processes, and open files/dlls.	Listing of running processes, open files and dlls obtained by using the pstools from http://www.sysinternals.com	All collected from running honeypot and saved to a USB pen drive. Filenames: autoruns.txt – listing of Registry "start up" entries. Handle.txt – listing of open files on the honeypot. Procinterrogate.txt – listing of processes & associated dlls running on the honeypot.
Media Image	Exact copy of the only partition on the honeypot hard drive.	Taken directly from the honeypot and transferred over the network to the analysts system. Name: ibm.img
Honeypot Ram Image	Dump of the Random Access Memory (ram) obtained from the honeypot after the incident.	Taken directly from the honeypot's ram. Name: mem.img

Analyst's Notes:

The first available evidence is from the analyst's (my) rather sparse notes. The notes tell the time

that the honeypot was connected to the network, when the Shields Up scan occurred, the time of network disconnection and the imaging of the drive. The full notes.txt file is shown below:

```
$ cat notes.txt
10:32 put machine on network and scanned from shields up site.
10:35 started tcpdump on monitoring computer
10:45 pulled out network cable
~10:50 accessed taskmanger
11:00 inserted usbdrive
12:00 imaged drive
```

Tcpdump/Network Log File

Analysis is made easier by the fact that we have the tcpdump logfile of all network traffic entering and exiting the honeypot. The tcpdump logfile is named sunday.tcp, and is analyzed on a Windows machine using WinDump.exe from http://windump.polito.it. Windump is a network analyzer that allows one to examine network traffic in real time or stored traffic that is saved to a compatible file. Windump is fully compatible with the UNIX/Linux program tcpdump that was used to collect the network traffic.
The output below shows the command Windump.exe being invoked with the "-n" option which prevents hostname resolution, and the "-r" option that tells WinDump.exe that it will be reading from the file sunday.tcp. The output shows that the honeypot, (IP 24.153.218.66) is sending a "Syn", or a request for a network connection to port 80, usually used for web servers, to over 100 different IP addresses. The source IP shown is that of the router (24.153.218.66). This is expected as the network tap with the connected computer running tcpdump and recording the traffic, is between the router and the cable modem. Only three "Syn's" are shown below for brevity.

```
$ ./WinDump.exe -n -r sunday.tcp |more
22:36:22.760164 IP 24.153.218.66.3743 > 192.168.248.3.80: S 108741014:108741014(0) win 16384 <m
ss 1460,nop,nop,sackOK> (DF)
22:36:22.761041 IP 24.153.218.66.3752 > 192.80.250.105.80: S 109180076:109180076(0) win 16384 <
mss 1460,nop,nop,sackOK> (DF)
22:36:22.761926 IP 24.153.218.66.3753 > 192.168.198.121.80: S 109238723:109238723(0) win 16384
<mss 1460,nop,nop,sackOK> (DF)
```

The large amount of network activity directed to so many different address in such a short time, is a strong indication that the honeypot has already been compromised, and is actively scanning for web servers. As stated earlier, tcpdump was not activated until after the honeypot had been connected to the network. By this time, it was too late. Had tcpdump been logging network traffic before placing the honeypot on the network, there would be a record of the exact exploit used to compromise the system.

Thus far, there is evidence that the machine has been compromised. To investigate further, the Intrusion Detection System (IDS) Snort was used to analyze the tcpdump logfile. Snort is a widely used and freely available IDS that can among other things, read network traffic from a file and compare it to "signatures" of exploits. If a match is found Snort triggers an alert, that in this case is sent to an alert file. Snort also writes the packet that triggered the alert to a tcpdump compatible file. Below are the first two entries in the Snort alert file.

The first line shows an "ICMP Destination Unreachable (Communication Administratively Prohibited)" message. This means that the honeypot tried to communicate with the system 192.65.201.221, but was rejected by a router along the way.

```
[**] [1:485:2] ICMP Destination Unreachable (Communication Administratively Prohibited)
[Classification: Misc activity] [Priority: 3]
07/27-22:36:23.244829 172.31.9.27 -> 24.153.218.66
ICMP TTL:52 TOS:0x0 ID:26390 IpLen:20 DgmLen:56
Type:3  Code:13  DESTINATION UNREACHABLE: ADMINISTRATIVELY PROHIBITED,
PACKET FILTERED
** ORIGINAL DATAGRAM DUMP:
24.153.218.66:3816 -> 192.65.201.221:80
TCP TTL:117 TOS:0x0 ID:20369 IpLen:20 DgmLen:48 DF
Seq: 0x6B312DC  Ack: 0x0
** END OF DUMP
```

The next entry in the Snort alert file warns that the honeypot may be attempting to exploit another web server. Snort's message is: " WEB-IIS ISAPI .ida attempt".

```
[**] [1:1243:8] WEB-IIS ISAPI .ida attempt [**]
[Classification: Web Application Attack] [Priority: 1]
07/27-22:36:23.845407 24.153.218.66:3882 -> 192.121.89.163:80
TCP TTL:128 TOS:0x0 ID:20476 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x6E2BC85  Ack: 0x4D4BB910  Win: 0x4470  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-007
tyfocus.com/bid/1065][Xref => http://www.whitehats.com/info/IDS552]
```

To verify that this was not a false positive, Windump was used to examine the snort log file. Windump was invoked with the -X option to print ASCII characters along with hex. We also specify the destination IP number that was found in the Snort warning. The Windump output is below:

```
$ c:/WinDump.exe -X -r snort.log.1060865909 'host 192.121.89.163'

22:36:23.845407 IP 24.153.218.66.3882 > 192.121.89.163.80: . 115522693:115524153(1460)
808208 win 17520 (DF)
0x0000   4500 05dc 4ffc 4000 8006 9827 1899 da42       E...O.@....'...B
0x0010   c079 59a3 0f2a 0050 06e2 bc85 4d4b b910       .yY..*.P....MK..
0x0020   5010 4470 91b3 0000 4745 5420 2f64 6566       P.Dp....GET./def
0x0030   6175 6c74 2e69 6461 3f58 5858 5858 5858       ault.ida?XXXXXXX
0x0040   5858 5858 5858 5858 5858 5858 5858 5858       XXXXXXXXXXXXXXXX
0x0050   5858 5858 5858 5858 5858 5858 5858 5858       XXXXXXXXXXXXXXXX
>>>>>>>>>>>>>>Cut Several Lines of X's

0x0250   4630 9a02 0000 e80a 0000 0043 6f64 6552       F0.........CodeR
0x0260   6564 4949 008b 1c24 ff55 d866 0bc0 0f95       edII...$.U.f....
```

We can see that the honeypot did send a malicious packet to 192.121.89.163. Line 0x0020 shows the start of the statement GET/default.ida? followed by a number of "X's". This is characteristic of the Code Red II Worm. The "X's" are padding to overflow the ida buffer on a

vulnerable machine. The text shown on line 0x0250 leaves little doubt that this is indeed the Code Red II worm (CodeRedII Worm Analysis).

We have solid evidence that the honeypot has been compromised by the Code Red II worm and that it is attempting to infect other machines. However, there is no record of the compromise of the honeypot in the network log file, only evidence that the honeypot is attacking other machines. From this point finding further evidence of Code Red II is expected, but there could also be other things that were installed during the time the honeypot was exposed.

Windows Log Files:

The next source of evidence for analysis comes from the Windows event logs. According to Microsoft Knowledge Base Article 315147, Windows 2000 has three primary event logs:

System log - records operating system events like a system service failure.
Security log – records logon and logoff successes and failures along with other security information.
Application log - records events that have to do with applications that are running on the server.

By default all three of these logs are kept in the %windir%\system32\config folder. The %windir% is a variable that represents the location of the Windows operating system. For Windows 2000 computers, the operating system installation is usually C:\Winnt.
In order to view the files the hard drive image of the honeypot was mounted on the analyst's Linux system using the mount command with the "noatime" and "noexec" options. This ensures that the image will not be modified during the examination, and that no programs can be executed from the mounted filesystem. The command used is shown below:

```
[root@localhost root]# mount -t ntfs -o loop,noatime,noexec ./ibm.img /mnt/vault
```

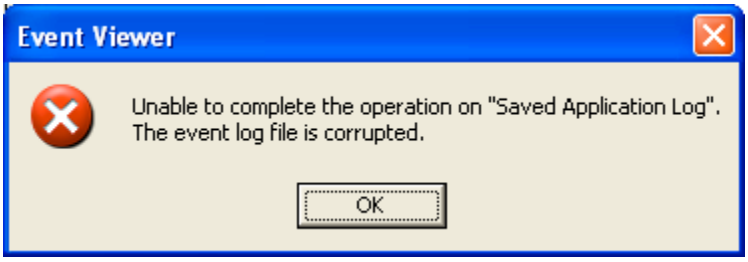
After mounting the filesystem, I copied the three event files found in C:\Winnt\system32\config to the Pen Drive for analysis on a Windows computer. This was done because the event files are not in a plain text format, and require special in order to be read. Two tools are readily available to read the event files; Windows Event Viewer, and psloglist from http://www.sysinternals.com. To ensure the integrity of the copied files, a MD5 checksum was computed for each file before transferring them to the Windows computer.

```
[root@localhost screenshots]# md5sum /mnt/vault/WINNT/system32/config/*.Evt
4cd6182439b4c2c6c5f425f53371647a  /mnt/vault/WINNT/system32/config/AppEvent.Evt
4407052e2885c8319711f5112fafaf5c  /mnt/vault/WINNT/system32/config/SecEvent.Evt
b64ec44a1384385729d4d354a0afe853  /mnt/vault/WINNT/system32/config/SysEvent.Evt
```

Another MD5 checksum was performed on the event files after they had been copied to the hard drive of the Windows machine. The files are identical.

```
$ md5sum.exe *.Evt
4cd6182439b4c2c6c5f425f53371647a *AppEvent.Evt
4407052e2885c8319711f5112fafaf5c *SecEvent.Evt
b64ec44a1384385729d4d354a0afe853 *SysEvent.Evt
```

Unfortunately, the event log files proved to be of little value because they are unreadable. Below is a representative error message found when attempting to open any of the three files.



The event log files appear to be corrupt. Perhaps psloglist can read the files. Psloglist was invoked with the "-l" option that specifies a stored event log file for input. Psloglist was run against all three event log files. A representative error message is shown below.

```
$ psloglist.exe -l AppEvent.Evt

PsLogList v2.3 - local and remote event log viewer
Copyright (C) 2000-2003 Mark Russinovich
Sysinternals - www.sysinternals.com

System log on \\MOTO-8CH4ZNEYRR:
Could not open AppEvent.Evt event log on MOTO-8CH4ZNEYRR:
The event log file is corrupted.
```

Unfortunately, it appears that the event log files are indeed corrupt and unreadable.

IIs Log Files:

The IIs log files are examined next. Once again MD5 checksums are done on the files before and after copying.

Md5checksum on the Honeypot:

```
[root@localhost screenshots]# md5sum /mnt/vault/WINNT/system32/LogFiles/W3SVC1/ex030727.log
494e41b5fb7827a80696a96b3a2f9387  /mnt/vault/WINNT/system32/LogFiles/W3SVC1/ex030727.log
[root@localhost screenshots]# md5sum /mnt/vault/WINNT/system32/LogFiles/W3SVC2/ex030727.log
3f7913b3c21221023a13d800cf9d61e7  /mnt/vault/WINNT/system32/LogFiles/W3SVC2/ex030727.log
```

Md5checksum after copying to the analyst's system.

```
$ md5sum.exe ./W3SVC1/ex030727.log ./W3SVC2/ex030727.log
494e41b5fb7827a80696a96b3a2f9387 *./W3SVC1/ex030727.log
3f7913b3c21221023a13d800cf9d61e7 *./W3SVC2/ex030727.log
```

The files are identical. By default, IIs logs all accesses to the websites it is serving to %windir%/system32/Logfiles/W3SVCx. The x stands for the number of the website. In the case of the honeypot there are two; the default, and the administration website. The logfiles are given a date-based name (Jones p1).

As expected there are two directories found in %windir%/system32/Logfiles, W3SVC1 and W3SVC2. Examining the files found in these directories yielded little information. Because of the similarity of the two files, and the fact that there are only two line entries in each log file, only the %windir%/system32/Logfiles/W3CVC1/ex030727.log is shown below.

```
$ cat ex030727.log
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2003-07-28 00:48:11
#SubComponent: Process Accounting
#Fields: date time s-event s-process-type s-user-time s-kernel-time s-page-faults s-total
 s-active-procs s-stopped-procs
2003-07-28 00:48:11 Reset-Interval-Start All 00.000% 00.000% 0 0 0 0
2003-07-28 00:48:11 Logging-Interval-Start All 00.000% 00.000% 0 0 0 0
```

The file only shows the starting of the IIS web server component. No other events are logged in the files. Normally one would expect to find a record of showing what was requested from the web server, and where it was requested from (IP number). After searching the Internet, it was discovered that several people have noted that IIS will shutdown without logging any events when presented with an IDA buffer overflow (Lovy p1).

Running Processes/Startup Files

In contrast to the log files, a wealth of information was found by examining the post compromise, pre-shutdown, honeypot. The running processes and a record of the start up files on the honeypot were captured by using three tools from Sysinternals.

The first file is called Handle.txt and was generated by running a program included on the SANS Forensics Course v1.6 CD named Handle v2.0, from Sysinternals. The first thing that stands out while examining the file, is the number of threads devoted to inetinfo process. According to Chapter one of the Microsoft Technet Overview of Internet Information Services 5.0 guide, inetinfo is the process that hosts the web and ftp Internet services.

A few definitions are in order. A process is simply a program to be executed along with the resources it requires, such as space in memory. A thread belongs to a process, and a process can have many threads when running on a modern operating system. Threads can share the resources, such as memory, of the process under which they run. This increases efficiency as the resources do not have to be duplicated. Threads can be added as needed. Inetinfo.exe is a process that can have many threads. (Maheswaran).
Looking at the output of the Handle.txt file, one can see that there are a large number of threads associated with the inetinfo process on the honeypot.
To find out just how many threads were associated with inetinfo, the following command was

executed.

```
$ grep inetinfo.exe\(852\) handle.txt | cut -d ":" -f2 |uniq -c
      1  Process         inetinfo.exe(852)
    330  Thread          inetinfo.exe(852)
```

The first part of the command grep, simply looks for inetinfo.exe(852), the process id of inetinfo. Next, the output is "cut" and the number of unique threads that belong to inetinfo.exe process id 852 are counted. As shown in the output above, there are 330 threads running within inetinfo. By default, IIS runs with approximately 10 inetinfo threads (Fenton p1).

IIS was very busy doing something. From the captured network traffic, and the Snort analysis, there was evidence of the honeypot sending hostile traffic that was consistent with a Code Red II infection. The large number of inetinfo threads found running on the honeypot is expected with a Code Red II infection, because the worm spawns 300 threads for a non-Chinese system. 600 threads on a Chinese system (CodeRedII Worm Analysis).

The second file to examine is named procinterrogate.txt. This file was generated using a program called ProcInterrogate version 0.0.1, written by Kirby Kuehl and was executed from the F.I.R.E CD with output going to a USB Pen Drive. ProcInterrogate lists the running processes, along with any Dynamic Link Libraries (dll's) the process is using. A dll is a library of data or functions that a Windows program can use if needed (DLL, Webopedia).
Dll's can also give clues as to the true function of a program. One tactic employed by Hackers is to give a malicious program a name that is similar to the name of a common Windows program. By examining the dll's associated with a program, one can sometimes detect when a program has the potential to do more than expected. For example, suppose you find a program called notepad.exe, a very common Windows program, running on a computer. At first glance the program appears innocuous. However, if ProcInterrogate were to show the WSOCK32.dll associated with the notepad process, the program would become suspect. This is because the WSOCK32.dll is used for a variety of network connections, and is not normally associated with notepad.exe. Examining dll's quickly becomes tedious with larger programs like inetinfo.exe, which requires well over 50 dll's to run.
The file procinterrogate.txt shows that 24 processes were running on the honeypot, none of which were surprising. All the processes were expected to be found running on web server, with the exception of tp4mon.exe, shown in the ProcInterrogate output below.

```
C:\WINNT\System32\tp4mon.exe (Process ID: 1124)

        Entry Point  Base       Size       Module
        0x00408650   0x00400000 0001B000   C:\WINNT\System32\tp4mon.exe
        0x00000000   0x77F80000 00079000   C:\WINNT\System32\ntdll.dll
        0x775A75F0   0x775A0000 00240000   C:\WINNT\system32\SHELL32.dll
        0x00000000   0x77F40000 0003C000   C:\WINNT\system32\GDI32.DLL
        0x77E8C3D8   0x77E80000 000B6000   C:\WINNT\system32\KERNEL32.DLL
        0x77E33BB4   0x77E10000 00065000   C:\WINNT\system32\USER32.DLL
        0x77DB87C7   0x77DB0000 0005A000   C:\WINNT\system32\ADVAPI32.DLL
        0x77D43958   0x77D40000 0006F000   C:\WINNT\system32\RPCRT4.DLL
        0x77C76944   0x77C70000 0004A000   C:\WINNT\system32\SHLWAPI.DLL
        0x77B5BEE4   0x77B50000 0008A000   C:\WINNT\system32\COMCTL32.DLL
        0x00000000   0x66CB0000 00009000   C:\WINNT\System32\tp4res.dll
```

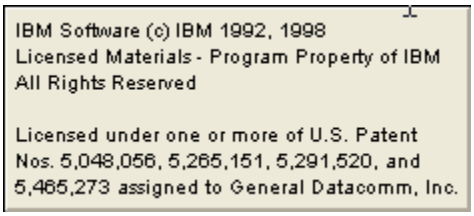
After examining the tp4mon.exe process and performing a Google search, it was found to be a program to monitor the Track Point found in IBM laptops (Task List Programs).

The final file to examine is named autoruns.txt. This file is the result of running the program autoruns.exe version 1.2 from Sysinternals. Autoruns.exe was launched from the F.I.R.E CD and output was saved to a USB Pen drive for later analysis. Autoruns.txt is a listing of the programs that are configured to start when the computer boots. Sometimes virii or trojan programs will be configured so that they launch every time the computer boots. It is therefore prudent to examine the output of autoruns.exe. The contents of the autoruns.txt file is shown below.

```
$ cat autoruns.txt
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
+ C:\WINNT\system32\userinit.exe
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce\
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\
HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
+ C:\WINNT\MWW32\manager\mwremind.exe autorun
+ tp4mon.exe
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\
C:\Documents and Settings\All Users\Start Menu\Programs\Startup
+ ThinkPad Modem Copyright.lnk -> C:\WINNT\MWW32\manager\mwcpyrt.exe
C:\Documents and Settings\Administrator\Start Menu\Programs\Startup
HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows\Load
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices\
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce\
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices\
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce\
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce\
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnceEx\
C:\WINNT\win.ini
```

The first thing listed is the userinit.exe program. This program is expected, and is a vital part of Windows. Userinit runs logon scripts & starts the shell, which is called Explorer.exe (Liutilities).
The next items of interest are on lines 7-8, where mwremind.exe and tp4mon.exe are both set to start at boot time. Both of these programs are expected on a IBM Thinkpad laptop computer. As stated before, tp4mon monitors the trackpad. Mwremind.exe is part of the IBM Thinkpad modem software package.
Line 11 shows a program called mycpyrt.exe configured to launch when a user logs on to the

computer. Mycpyrt.exe is a copyright banner associated with the IBM Thinkpad modem. Launching mycpyrt.exe from the command line shows the banner below:



The final entry in the list is C:\Winnt\win.ini. The win.ini file was used on older versions of Windows to launch programs, however it does not start programs in the Windows 2000 operating system. There are no unusual entries in the file.

Media Image

The next step is to mount the drive image for examination. This was accomplished using the following command.

```
[root@localhost root]# mount -t ntfs -o loop,noatime,noexec ./ibm.img /mnt/vault
```

Autopsy v1.73, written by the security group @stake (http://www.atstake.com) was used to examine the mounted image. Autopsy has a number of useful features such as the ability to create a timeline, do searches for a particular string of characters, and recover deleted files.

One of the first things found using Autopsy, was the file root.exe. This file was found in c:\Inetpub\scripts\root.exe. This file is really a copy of cmd.exe the Windows command usually found in c:\winnt\system32\cmd.exe. The Code Red II virus copies cmd.exe to C:\Inetpub\scripts\root.exe to make the command available through a web browser. This makes it possible for anyone to connect to the web server and execute a command like this: "http://ipAddress/c/winnt/system32/cmd.exe?/cmd.exe/c+dir". This command would execute the dir command on drive C. Running other programs on the compromised computer is just a matter of substituting the command in place of the "dir" given in the example above (Russell & Mackie).

To ensure that root.exe is really a renamed cmd.exe, a MD5 checksum was computed for both files.

```
[root@localhost vault]# md5sum Inetpub/scripts/root.exe WINNT/system32/cmd.exe
53fcda64f7122bcb4b601287039a8075  Inetpub/scripts/root.exe
53fcda64f7122bcb4b601287039a8075  WINNT/system32/cmd.exe
```

The files are indeed identical.

Another characteristic of the Code Red virus, is that it creates a file called explorer.exe to the

root level of the C drive. Explorer.exe is found in the expected location, C:\explorer.exe. The file was examined using the strings v2.11.93.0.2 utility with the -n option. The "-n 5" shown in the command below, eliminates any extraneous binary noise by only printing strings longer than 5 characters.

```
[root@localhost vault]# strings -n 5 explorer.exe
\EXPLORER.EXE
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
SFCDisable
SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots
/Scripts
/MSADC
c:\,,217
d:\,,217
KERNEL32.dll
ADVAPI32.dll
Sleep
GetWindowsDirectoryA
WinExec
RegQueryValueExA
RegSetValueExA
RegOpenKeyExA
RegCloseKey
```

The strings output shows the troubling information contained in explorer.exe. The code has the potential to create three virtual roots; one is the entire c drive (c:\,,217) another is the entire d drive (d:\,,217) and the last is the c:\Program Files\Common Files\System\msadc directory. Creating these virtual roots effectively exposes the entire C & D (if there is one) logical drives to the internet.

The c:\explorer.exe file is executed after a post-infection reboot. It was not executed on the honeypot, because the computer was never rebooted. The stat command verifies this, by showing that the Access, Modify, and Change times are all the same. If it were executed, we would expect the access time to be later than the Change time.

```
[root@localhost vault]# stat explorer.exe
  File: "explorer.exe"
  Size: 8192            Blocks: 18         IO Block: -4611692409338723328 Regular File
Device: 700h/1792d      Inode: 9283        Links: 1
Access: (0400/-r--------)  Uid: (    0/    root)   Gid: (    0/    root)
Access: Mon Jul 28 03:32:20 2003
Modify: Mon Jul 28 03:32:20 2003
Change: Mon Jul 28 03:32:20 2003
```

TimeLine Analysis.

A timeline was generated using the timeline feature of Autopsy v1.73. The key events are summarized in the table below.

Date	Time	Event	Source
07/27/03	13:31:42	Windows 2000 Advanced Server and IIS 5.0 installation begins.	Timeline output of Autopsy. Creation of Metafile-data begins at 13:31
07/27/03	~ 22:22:00	The honeypot is put outside firewall. To verify that the honeypot was reachable via an external network, the honeypot was port scanned by Shields Up, a web based port scanner found at http://www.grc.com.	Investigator's notes.
07/27/03	22:24:58	Honeypot is compromised. The file root.exe, which is characteristic of the Code Red II worm, is created in the c:\Inetpub\scripts directory.	Autopsy timeline output.
07/27/03	22:32:20	The file c:\exporer.exe is created. This file is also part of the Code Red II worm.	Autopsy timeline output.
07:27:03	22:36:22	The program tcpdump is invoked on the investigator's computer.	Analysis of tcpdump packet capture.
07:27:03	Sometime between 22:24:58 and 22:36:22	The Code Red II worm begins scanning for port 80 on other hosts.	Tcpdump logfile.
07:27:03	~ 22:45:00	The Honeypot is physically disconnected from the network.	Investigators notes.

Examining the TimeLine in more detail, we can see the first activity occurs on July 27, 2003. The files shown below are hidden files, all created at the time Windows 2000 was installed.

```
Sun Jul 27 2003 13:31:42   268232 mac -/-r-xr-xr-x 0        0        9-128-8  C:\/$Secure:$SDS
                             4096 mac -/-r-xr-xr-x 0        0        1-128-1  C:\/$MFTMirr
                               56 mac -/-r-xr-xr-x 0        0        9-144-14 C:\/$Secure:$SDH
                             8192 mac -/-r-xr-xr-x 48       0        7-128-1  C:\/$Boot
                             2560 mac -/-r-xr-xr-x 48       0        4-128-4  C:\/$AttrDef
                           131072 mac -/-r-xr-xr-x 0        0        10-128-1 C:\/$UpCase
                                0 mac -/-r-xr-xr-x 0        0        8-128-2  C:\/$BadClus
                       2144346112 mac -/-r-xr-xr-x 0          0        8-128-1  C:\/$BadClus:$Bad
                         12828672 mac -/-r-xr-xr-x 0        0        2-128-1  C:\/$LogFile
                          9520128 mac -/-r-xr-xr-x 0        0        0-128-1  C:\/$MFT
                           130888 mac -/-r-xr-xr-x 0        0        6-128-1  C:\/$Bitmap
                                0 mac -/-r-xr-xr-x 48       0        3-128-3  C:\/$Volume
                              344 mac d/dr-xr-xr-x 0        0        11-144-4 C:\/$Extend
                               56 mac -/-r-xr-xr-x 0        0        9-144-11 C:\/$Secure:$SII
```

Next we see the file root.exe is created at 22:24:58. Recall that root.exe is really a renamed copy of cmd.exe. We can also see that ipconfig.exe was accessed at 22:25:00. One of the first thing that the Code Red II worm does, is determine the IP number of machine it has infected. This is done so that the worm can figure out what IP numbers to attack, and so that the compromised machine will not re-infect itself (Code RedII Analysis). Ipconfig.exe will among other things, list any IP's assigned to the computer on which it is executed.

```
Sun Jul 27 2003 22:24:58^     8368 .a. -/-r-xr-xr-x 0        0        669-128-4 C:\/WINNT/Fonts/ega40woa.fon
                              4304 .a. -/-r-xr-xr-x 0        0        337-128-4 C:\/WINNT/Fonts/cga80woa.fon
                            236304 ..c -/-rwxrwxrwx 0        0        9281-128-3 C:\/Inetpub/scripts/root.exe
                             36656 .a. -/-r-xr-xr-x 0        0        599-128-4 C:\/WINNT/Fonts/dosapp.fon
Sun Jul 27 2003 22:25:00     35600 .a. -/-rwxrwxrwx 0        0        944-128-4 C:\/WINNT/system32/ipconfig.exe
```

Sometime shortly after 22:25:00, the Code Red Worm attempts to propagate itself over the network.

Approximately, eight minutes later, (22:32:20) the file explorer.exe is created.

```
Sun Jul 27 2003 22:32:20     8192 mac -r-------- 0        0        9283     /mnt/vault/explorer.exe
```

Finally, at 22:36:22 tcpdump is invoked on the investigators computer. The Investigator's computer is connected to a network tap so that all traffic entering and exiting the honeypot is recorded. Unfortunately, tcpdump is not started until after the honeypot has been compromised. This is because the Investigator did not want to start recording network traffic until after the honeypot had been scanned by Shields Up. This was obviously a mistake, as the honeypot compromise took place less than 3 minutes after the honeypot was exposed to the Internet. Below are the first entries in the tcpdump logfile, showing that the Code Red II Worm has already entered the propagation stage.

```
22:36:22.760164 IP 24.153.218.66.3743 > 192.168.248.3.80: S 108741014:108741014(0) win 16384 <mss 1460,nop,nop,sack
OK> (DF)
22:36:22.761041 IP 24.153.218.66.3752 > 192.80.250.105.80: S 109180076:109180076(0) win 16384 <mss 1460,nop,nop,sac
kOK> (DF)
```

The network cable is unplugged around 22:45:00, and the USB Pen drive is attached to the honeypot 15 minutes later at 23:00:17. We can tell the USB drive was plugged in at that time by looking at the access time of USBSTOR.SYS, which is the Microsoft driver for USB storage devices.

```
Sun Jul 27 2003 23:00:17    19760 .a. -r-------- 0        0        9285     /mnt/vault/WINNT/system32/drivers/USBST
OR.SYS
```

The last time the system was used can be determined by looking at the last entries in the timeline. The last entry shows that the ntuser.dat.LOG file was written to on July 27, 2003 at 23:40:39. The computer must have been turned off around this time or there would be later entries in the timeline. This also matches the turn off time given in the investigators notes.

```
Sun Jul 27 2003 23:40:39   217088 mac -/-r-xr-xr-x 0        0        9108-128-4 C:\/Documents and Settings/Administrator
/NTUSER.DAT
                             1024 mac -/-r-xr-xr-x 0        0        9187-128-4 C:\/Documents and Settings/Administrator
/ntuser.dat.LOG
```

Analysis of Memory Dump:

At approximately 22:45, the contents of the honeypot's memory was dumped to a file called ibm.img, and transferred via USB Pen drive to the analyst's system. The strings command was used to show the readable text found in the honeypot's memory.

The first search was for the string "ida?", which is contained in Code Red II network packets. This string is expected, and found in the honeypot's memory.

```
$ strings -a5 mem.img |grep ida?
/default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXX%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u
%u531b%u53ff%u0078%u0000%u00=a
```

By this point there is little doubt that the honeypot has been compromised by the Code Red II worm. What is not known, is whether something besides the Cod e Red II worm was loaded onto the honeypot. Other malicious programs such as network sniffers, to monitor network traffic, or key loggers to capture keystrokes, could loaded and running on the honeypot. To look for evidence of these sorts of programs the following commands were executed:

strings -a5 mem.img | grep logger
strings -a5 mem.img | grep keylog
strings -a5 mem.img | grep sniff

Interestingly the last command returned something called sniffer.exe. To investigate further, the "-a 15" option was passed to grep to show the 15 lines before and after the word sniff.

```
$ strings -a5 mem.img |grep -a15 sniff
rsync.exe
Sample script.whs
SAMP~1B1.WHS
scp.exe
sdiff.exe
sed.exe
seq.exe
service.exe
SFind.exe
share.exe
Shortcut.dll
showin.exe
sid2user.exe
sigs.exe
size.exe
sniffer.exe
sort.exe
split.exe
ssh.exe
ssh-add.exe
ssh-agent.exe
SSH-~1DA.EXE
ssh-host-config
SSH-~1DD
response_kit
RESPON~8
win2k_xp
procinterrogate.txt
PROCIN~1.TXT
response_kit
RESPON~8
```

As shown above, a number of suspicious programs were found, however it seems odd that they are mostly in alphabetical order and so many of them start with the letter "s". Upon closer review, the investigator recognized the string "response_kit" as being a directory found on the SANS Track 8 Course CD v1.6. Running the command ls -l s* in the response_kit/win2k_xp directory of the SANS CD showed the following:

```
$ pwd; ls -l s*
/cygdrive/d/response_kit/win2k_xp
-r-xr-xr-x    1 ra6912   None        26112 Sep 27  2001 scp.exe
-r-xr-xr-x    1 ra6912   None        14848 Jan 17  2000 sdiff.exe
-r-xr-xr-x    1 ra6912   None        46592 May 31  2000 sed.exe
-r-xr-xr-x    1 ra6912   None        23552 Mar 30  2001 seq.exe
-r-xr-xr-x    1 ra6912   None       215040 Oct 25  2001 service.exe
-r-xr-xr-x    1 ra6912   None       374784 Oct 25  2001 share.exe
-r-xr-xr-x    1 ra6912   None        24064 Nov 12  2000 showin.exe
-r-xr-xr-x    1 ra6912   None        49152 Apr 26  1998 sid2user.exe
-r-xr-xr-x    1 ra6912   None        12800 Oct 25  2001 sigs.exe
-r-xr-xr-x    1 ra6912   None       303104 Oct  2  2001 size.exe
-r-xr-xr-x    1 ra6912   None       215040 Oct 25  2001 sniffer.exe
-r-xr-xr-x    1 ra6912   None        40960 Jun 16  2001 sort.exe
-r-xr-xr-x    1 ra6912   None        24064 Jun 16  2001 split.exe
-r-xr-xr-x    1 ra6912   None       554496 Sep 27  2001 ssh-add.exe
-r-xr-xr-x    1 ra6912   None       494080 Sep 27  2001 ssh-agent.exe
-r-xr-xr-x    1 ra6912   None        12324 Sep 27  2001 ssh-host-config
-r-xr-xr-x    1 ra6912   None       560640 Sep 27  2001 ssh-keygen.exe
-r-xr-xr-x    1 ra6912   None       401408 Sep 27  2001 ssh-keyscan.exe
-r-xr-xr-x    1 ra6912   None         4657 Sep 27  2001 ssh-user-config
-r-xr-xr-x    1 ra6912   None       726016 Sep 27  2001 ssh.exe
-r-xr-xr-x    1 ra6912   None        29696 Sep 13  2001 strace.exe
-r-xr-xr-x    1 ra6912   None       566448 Mar  8  2000 strace.sys
-r-xr-xr-x    1 ra6912   None       303104 Oct  2  2001 strings.exe
-r-xr-xr-x    1 ra6912   None       491008 Oct  2  2001 strip.exe
```

The sniffer.exe file found in memory was not a malicious sniffer, rather it was a program called Sniffer Detector, written by H. Carvey and found on the SANS Course CD v1.6.

Recovering Deleted Files:

Unfortunately, the Code Red Worm only resides in memory, therefore deleted files are not expected and were not found on the victim computer. However, I did recover a file to illustrate the technique involved.

Recovering files in Autopsy is a simple as clicking on the "All Deleted files" button. This makes all deleted files visible in a web browser window. From there, it is just a matter of clicking on the file, then selecting export from the bottom pane. Once this is done, a dialog box prompts for the location to save the file. Autopsy makes it easy, however a more in depth explanation is in order. At this point, it is necessary to understand a little bit about hard disks and the NTFS file system.

At its most basic level, a hard disk is divided into small chunks called blocks. The hard disk of the honeypot is divided into 512 byte blocks, which is a very common size. The NTFS file system organizes these smaller blocks into clusters, which are the smallest units that can hold a file. For disks that have a partition (logical subdivision of the disk) size larger than 1 gigabyte and smaller than 2 gigabytes, the default cluster size is 2048 bytes (Default Cluster). This means that any file written will consume at least 2048 or 2k, of disk space. So for example, if you have a file that is 1.5k it will take up 2k of disk space as far as the file system is concerned. Similarly, if you have file that is 2.5k it will need 2 clusters and therefore consume 4k of disk space. The 2.5k file will not actually fill both clusters, there will be 1.5k of unused space called "slack space" at the end of the second cluster that is unable to be used by the filesystem. This slack space might contain remnants of an old file, or it may be used by a hacker to hide a file.

The NTFS filesystem keeps track of the files contained in the clusters on a hard disk through the use of a special file called the Master File Table, or MFT. Each file known to the operating system has record in the MFT, that includes the location of the file. For example, notice the output below. This is from the deleted file C:\WINNT\inf\dispdet.inf found through Autopsy:

```
$ cat recover.rpt.txt
            Autopsy MFT Entry Report (ver 1.73)

----------------------------------------------------------------
MFT Entry: 555-128-4
Pointed to by file:
  C:\WINNT/inf/dispdet.inf
MD5 of istat output: 6171307854f592d98997d4822771c6c5
Image: /home/mikew/tools/locker//newhackcst/newhackcst/images/ibm2.img
Image Type: ntfs
Date Generated: Sun Sep  7 19:42:05 2003
Investigator: mikew
----------------------------------------------------------------
MFT Entry: 555
Sequence: 1
Allocated
UID: 0
DOS Mode: File
Size: 13416
Links: 1
Name: dispdet.inf

$STANDARD_INFORMATION Times:
Created:        Tue Dec  7 06:00:00 1999
File Modified:  Tue Dec  7 06:00:00 1999
MFT Modified:   Sun Jul 27 13:42:06 2003
Accessed:       Sun Jul 27 13:32:53 2003

$FILE_NAME Times:
Created:        Sun Jul 27 13:32:23 2003
File Modified:  Sun Jul 27 13:32:53 2003
MFT Modified:   Sun Jul 27 13:32:53 2003
Accessed:       Sun Jul 27 13:32:53 2003

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-5)   Name: N/A   Resident   size: 88
Type: $SECURITY_DESCRIPTOR (80-3)   Name: N/A   Resident   size: 148
$
656245 656246 656247 656248 656249 656250 656251

File Type: ASCII English text, with CRLF line terminators
```

At the top of the file, we can see the number that it used to keep track of the file called the MFT Entry. For this file the number is 555-128-4. Five lines later we see that the size of the file is 13416 bytes. We can also see the times that the MFT was modified for this record and the attributes of the file. The 2nd from the last line in the Autopsy output, shows that this file occupies the 7 clusters 656245 – 656251. This is expected because the file is 13416 bytes \ 2048 cluster size = 6.5 or 7 clusters. One way to recover a deleted file, is simply to extract the location information from the MFT. This is the method employed by the utility icat from http://www.sleuthkit.org. Below is the command used to recover our example file C:\WINNT\inf\dispdet.inf.

```
[root@localhost bin]# ./icat -f ntfs /home/mikew/tools/locker/newhackcst/newhackcst/images/ibm2.img 555-128-4 >
d/sda1/recover/combine
```

In the example above, icat is given the filesystem type (-f ntfs) then the dd image of the honeypot, followed by the MFT Entry number 555-128-4. Finally, the output is redirected to a file called combine. Examining the recovered file below, we see the expected file size of 13416

bytes.

```
$ ls -l combine
-rwx------+   1 ra6912    None        13416 Sep  7 20:23 combine
```

It is also possible to recover a file without using the MFT Entry. In the example below, dcat from http://www.sleuthkit.org is used to extract each cluster separately, and to redirect the output to separate files, named a – g. Luckily, all of the clusters are contiguous. Many larger files are not, and may have 100's of clusters scattered around the hard disk making recovery much more tedious.

```
[root@localhost bin]# ./dcat -f ntfs /home/mikew/tools/locker/newhackcst/newhackcst/images/ibm2.img 656245 > /mnt
da1/recover/a
[root@localhost bin]# ./dcat -f ntfs /home/mikew/tools/locker/newhackcst/newhackcst/images/ibm2.img 656245 > /mnt
6 > /mnt/sda1/recover/b
[root@localhost bin]# ./dcat -f ntfs /home/mikew/tools/locker/newhackcst/newhackcst/images/ibm2.img 656246 > /mnt
7 > /mnt/sda1/recover/c
[root@localhost bin]# ./dcat -f ntfs /home/mikew/tools/locker/newhackcst/newhackcst/images/ibm2.img 656247 > /mnt
8 > /mnt/sda1/recover/d
[root@localhost bin]# ./dcat -f ntfs /home/mikew/tools/locker/newhackcst/newhackcst/images/ibm2.img 656248 > /mnt
9 > /mnt/sda1/recover/e
[root@localhost bin]# ./dcat -f ntfs /home/mikew/tools/locker/newhackcst/newhackcst/images/ibm2.img 656249 > /mnt
50 > /mnt/sda1/recover/f
[root@localhost bin]# ./dcat -f ntfs /home/mikew/tools/locker/newhackcst/newhackcst/images/ibm2.img 656250 > /mnt
1 > /mnt/sda1/recover/g
```

Examining the files recovered with dcat, we can see below that all are 2048 bytes in size, or in other words, the entire cluster has been recovered.

```
[root@localhost clusters]# ls -la
total 18
drwxr-xr-x    2 root     root         2048 Sep  7 20:44 .
drwxr-xr-x    3 root     root         2048 Sep  7 20:44 ..
-rwxr-xr-x    1 root     root         2048 Sep  7 20:20 a
-rwxr-xr-x    1 root     root         2048 Sep  7 20:20 b
-rwxr-xr-x    1 root     root         2048 Sep  7 20:20 c
-rwxr-xr-x    1 root     root         2048 Sep  7 20:20 d
-rwxr-xr-x    1 root     root         2048 Sep  7 20:20 e
-rwxr-xr-x    1 root     root         2048 Sep  7 20:21 f
-rwxr-xr-x    1 root     root         2048 Sep  7 20:21 g
```

Below, the files are combined to recreate the deleted file:

```
[root@localhost clusters]# cat a b c d e f g > ../combined.dcat

$ ls -la
total 28
drwxr-xr-x+   2 ra6912    None            0 Sep  7 23:02 .
drwx------+   4 ra6912    None            0 Sep  7 23:02 ..
-rwx------    1 ra6912    None        13416 Sep  7 23:02 combine
-rwx------    1 ra6912    None        14336 Sep  7 23:02 combined.dcat
```

We see that the combined.dcat file is actually larger than the deleted file. This can be explained by the fact that we recovered more than just the file, we recovered all of the clusters. Seven clusters x 2048 = 14336. The diff command was run to compare the two files, they didn't differ by much.

```
$ diff combine combined.dcat
526a527
>
\ No newline at end of file
```

Had there been file remnants or part of hidden file in the slack space of the last cluster, we could have extracted it using the dcat recovery method.
The C:\WINNT\inf\dispdet.inf that was recovered, is an information file that contains a list of the displays that are supported by Windows 2000. After the initial installation, the file is no longer needed and is deleted. Rather than list the entire file, only the first 10 lines of the file are shown below. The head command version 2.0.21, was used to extract the first 10 lines from the recovered file.

```
$ head combine

; Detect.inf (for SUR)
;
; List of supported displays, manufacturers
;

[Version]
Signature="$CHICAGO$"
Provider=%MS%
LayoutFile=layout.inf
```

String Search.

For the string search, I once again used Autopsy version 1.73. Autopsy makes the search easy by providing a "Keyword Search" button at the top of a browser window. The search covers the entire mounted media image. The output below is from the Autopsy keyword search page. The number of occurrences of each search term is found in the parenthesis following the searched word.

The first search was for the word "sniff" . This was done to check for the presence of a network sniffer. A network sniffer is a program that listens to all traffic addressed to or passing a network interface. Output from the sniffers are typically written to a log for later retrieval, or emailed to a throw away account. The goal is usually to obtain valid computer login names and passwords. Interestingly, 69 occurrences of the word sniff were found, none of which proved to be suspect.

The next words searched were “code red”, “hack” & “default.ida”. This was done to look for evidence of the Code Red II worm and it’s tell-tale phrase, “Hacked by Chinese”, as well as the string default.ida which was found in the captured network traffic. Unfortunately, it appears that may programmers also like the work hacked, because 313 occurrences were found. All of these occurrences were unrelated to the Code Red II worm, and after further examination, none were found to be suspect. No occurrences of the phrase “code red” or “default.ida” was found.

Finally, I ran strings against the memory image. Because the Code Red II worm resides in memory, we should be able to find evidence of the worms signature. I ran 'strings mem.img | grep default.ida' resulting in the following output.

```
$ strings -a5 mem.img |grep ida?
/default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXX%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u
%u531b%u53ff%u0078%u0000%u00=a
```

The output shown above is also listed in the Memory Analysis section of the paper, along with additional string searches performed against the image of the physical memory.

Conclusion:

The system that was examined was the victim of a successful Code Red II worm attack. The trademark files were found in the expected locations. That is a copy of the C:\winnt\system32\cmd.exe was found to be copied to C:\inetpub and renamed to root.exe and the file explorer.exe was found at the root level of the C drive.

Had the system been rebooted, C:\explorer.exe would have executed at first login and created a virtual root, making the entire C drive available via the Microsoft IIS web server.
Further evidence that this was a Code Red II worm infection, comes from the network logs, which show that the machine began sending out network packets that contained the characteristic signature of Code Red II.

The memory analysis also showed that the Code Red II worm was present in the systems memory, and had spawned 330 threads, a number consistent with the default behavior of the worm running on a non-Chinese computer.

The evidence of the Code Red II worm is very clear, however we also wanted to see if any additional exploits, or hacking tools were installed on the honeypot. Was it possible that the Code Red II worm was used to gain access to the computer, so that the machine could be loaded with hacking tools etc?

There is no evidence that this occurred, based on the registry information present in the autoruns.txt file, or from the string searches of the hard drive and memory of the honeypot. In hindsight, it would have been more interesting to terminate the propagation phase of the Code Red II worm by rebooting the system and allow the virtual root to be created. It is very likely that if this would have been done, the system would have been exploited further.

References:

CodeRedII Worm Analysis. 04 Aug 2001. Eye Digital Security. 20 Aug. 2003 <http://www.eeye.com/html/Research/Advisories/AL20010804.html>

Default Cluster Size for FAT and NTFS. Microsoft Knowledge Base Article – 140365.

Microsoft. 25 Aug. 2003. http://support.microsoft.com/?kbid=140365

Dll. Webopedia. 08 Mar. 2003. Internet.com. 25 Aug. 2003. http://www.webopedia.com/TERM/D/DLL.html

Fenton, Jon. "Re: IIS developers, please help me, very thank's !" Online Posting. 16 Oct. 2001. Microsoft.Public.Inetserver.IIS. Forum. 20 Aug. 2003. <http://groups.google.com/groups?hl=en&lr=&ie=UTF-8&oe=UTF-8&selm=u%233jAQmVBHA.1596%40tkmsftngp07>

HOW TO: Clear the Event Logs in Windows 2000. Microsoft Knowledge Base Article – 315147. Microsoft. 20 Aug. 2003. http://support.microsoft.com/?kbid=315147

Lovy, Scott. "Microsoft." Online Posting. 02 Aug. 2001. Microsoft.Public.Inetserver.IIS Forum. 20 Aug. 2003. <http://groups.google.com/groups?selm=ujTSDX2GBHA.1892%40tkmsftngp02&oe=UTF-\8&output=gplain>

Jones, Allen. It's All in the Logging. Windows Web Solutions. Nov. 2000. Windows & .Net Magazine Network. 20 Aug. 2003. http://www.windowswebsolutions.com/Articles/Index.cfm?ArticleID=15835&pg=1

LIUtilities Online. LIUtilities Inc. 15 Aug. 2003 < http://www.liutilities.com/products/wintaskspro/processlibrary/userinit/>

Maheswaran, Muthucumaru. "Processes Lecture Notes". Mcgill University. 20 Aug. 2003. http://www.cs.mcgill.ca/~cs310/2003B/LECTURES/PPTs/02-process.ppt

MD5 Checksum Utility. 3L Corporation. 20 Aug. 2003. < http://www.shen.myby.co.uk/threel/tech/tools/md5.htm>

Overview of Internet Information Services 5.0. Microsoft Press. 20 Aug. 2003. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/iis/iis5/reskit/iis50rg/iischp1.asp

Russell, R. & Mackie, A. Code Red II Worm. 05 Aug. 2001. Security Focus. 21 Aug. 2003. http://aris.securityfocus.com/alerts/codered2/010805-Analysis-CodeRedII.pdf

Schwartz, David. Redistributing Microsoft Visual C++ 6.0 Applications. MSDN. Aug. 2000. Microsoft. 18 Aug. 2003. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvc60/html/redistribvc6.asp>

Task List Programs. Answers That Work.com. 21 Aug. 2003. http://www.answersthatwork.com/Tasklist_pages/tasklist_t.htm