



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

Check out the list of upcoming events offering
"Advanced Incident Response, Threat Hunting, and Digital Forensics (Forensics
at <http://www.giac.org/registration/gcfa>

Forensic Investigation, Analysis, Documentation, and Law

GCFA Practical Assignment
Version 1.3

Karl V. Prentner

Executive Summary

This paper contains information that illustrates the competencies necessary to investigate a computer incident in a professional, competent, complete, and accurate manner. The areas of competencies include, but are not limited to the following.

All aspects of computer forensic and forensics in general find common ground in the law. Understanding the law guides the investigator through the case. The capability to share the law accurately with colleagues, and law-people as well, is a benefit to the nature of forensics. Citing the law along with evidence has a profound affect on reports generated from the forensic results. Knowledge of how to proceed with evidence, the potential of misleading evidence and handling interrogations all play an important role in the comprehensive investigation.

Forensic analysis is not complete without describing the unknown and uncharted files of the Internet. Determining the characteristics of a suspect and potentially malicious in nature file found on a computer is paramount to determining what occurred during an incident. Lack of knowledge as to what happened at the end of the investigation leaves the door open for another incident of a similar nature.

Last, but not least, computer forensics is incomplete without the analysis of the computer system(s) identified within the incident. Completing a comprehensive analysis of physical memory, network connections and activity, active processes and services, and fixed media is the backbone of investigating what occurred within the realm of a digital information incident. Following the proper steps, analysis of the media in the correct order, and the overall thoroughness of the investigation determines how well we are able to take action against future incidents and proactively change human and system processes to safeguard against the vulnerabilities discovered.

Part 1

Analysis of an Unknown Binary

1.1 Background:

Analysis of an unknown binary is dependent on establishing an environment in which to operate. We must assume that executing, disassembling, or debugging the binary could cause damage to the system on which the binary resides. The assumption should also include the possibility of impacting other systems within the network segment on which the binary resides. To mitigate this situation the creation of a secure and sterile laboratory is essential.

To accomplish these types of investigations we created a system in the Information Security lab specifically for these types of experimental investigations. The system is a Gateway 2000 workstation with a Pentium II 300 MHz processor and 512 Megabytes of physical memory. This system has an 11 gigabyte hard drive that is dual bootable as a Windows 2000 workstation or Red Hat Linux 8.0. We use a modified "FIRE"¹ CD for verifiable, bona fide binaries. We name this workstation "Bomb_Shelter" or "Shelter" for short. We also utilize forensic analysis such as "The @stake Sleuth Kit" (TASK) and "Autopsy" a browser based GUI that utilizes TASK functionality² by Brain Carrier from @Stake, "The Forensic Tool Kit (FTK),"³ and "IDS – The Interactive Dissassembler," or "IDA Pro."⁴

FIRE is a portable bootable cdrom based distribution with the goal of providing an immediate environment to perform forensic analysis, incident response, data recovery, and virus scanning and vulnerability assessment. A secure known environment is necessary because an authorized user could have corrupted binaries (such as CDM.exe, dd.exe, nc.exe) on the server. Two different modes are available for FIRE usage. The first mode is relevant to this paper and utilized whenever the server remains on-line and nor disconnect from the network or shut down. Often, business considerations warrant a server remaining on-line. In these cases, we utilize FIRE by inserting the CD into the Cdrom drive of the victim server in order to access a reliable and secure command shell. The CD also contains most every binary that a forensic analyst would need to complete an initial investigation and digital media capture or image. We describe these techniques later in this paper. The second scenario is for use on compromised systems taken off-line. After placing the FIRE CD in the Cdrom drive, we reboot the server. During system startup, interrupt the process to enter "system setup" or BIOS. "System setup" or BIOS controls how the hardware behaves before operating system initiation. Ensure that boot to Cdrom is the first choice within boot options. Next, we save the BIOS settings and proceed with the server boot.

¹ <http://fire.dmzs.com/>

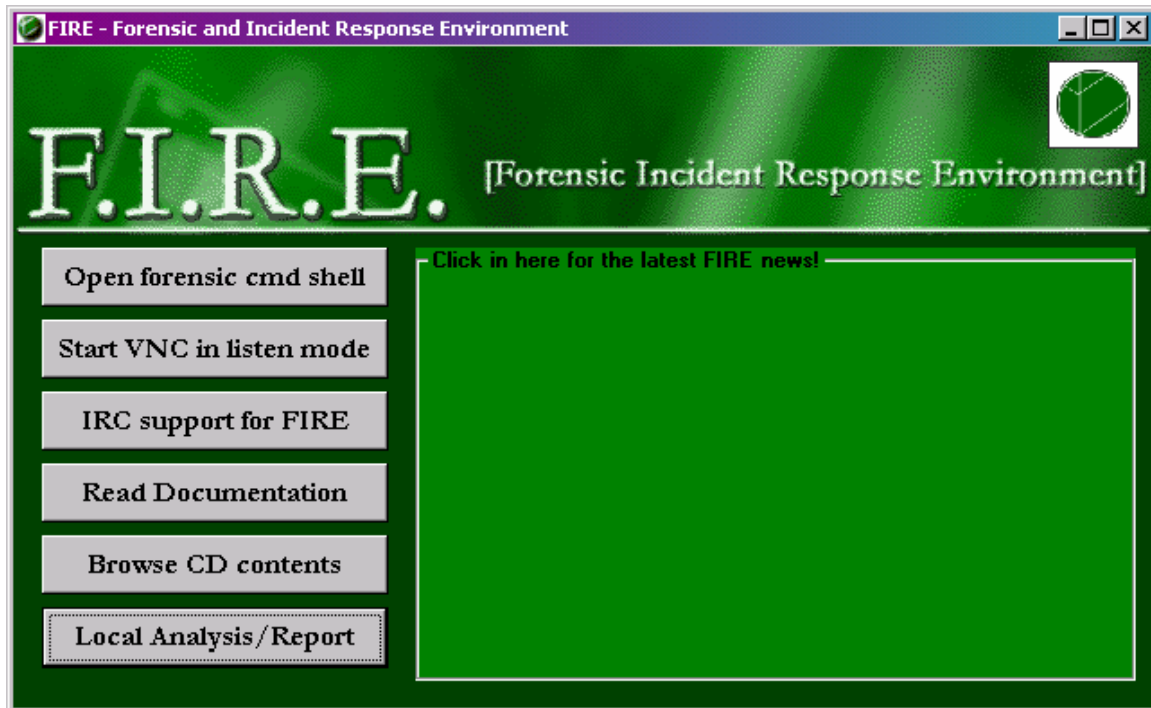
² <http://www.atstake.com/research/tools/task/>

³ http://www.accessdata.com/Product04_Overview.htm?ProductNum=04

⁴ <http://www.datarescue.com/idabase>

At this time, a LINUX operating system boots self-contained within the FIRE CD in order to create a secure and reliable environment in which to perform forensic image captures and analytical techniques. The following screen shot 1.1.1 depicts the WIN32 FIRE interface.

Screen Shot 1.1.1



TASK and Autopsy are open source shareware. They are freely available at many locations on the Internet but the most reliable source to obtain the software comes from the software creators, <http://www.atstake.com/research/tools/task/>. TASK provides great forensic capabilities utilized from a UNIX, LINUX, or BSD operating system. TASK enables the viewing and reclamation of deleted files and directories as well as unallocated disk space, timeline of file activity utilizing MACTime timestamps that include file creation/modification/access, key work string searches, and other case related jobs such as notes and reporting.

We chose FTK as our corporate WIN32 forensic utility. Another comparable tool preferred by some Forensic Examiners is Encase⁵ from Guidance Software. Both tools facilitate string searches, searches by file type, viewing of deleted files and unallocated disk space. These tools also generate file information such as cryptographic hashes. A cryptographic hash value is a number generated from a string of text, a file, or a drive/memory image and is generated utilizing a cryptographic algorithm in such a way that it is extremely unlikely that some other string, file, or image will produce the same hash value.

⁵ <http://www.encase.com/>

For the purpose of this report, we utilize the MD5sum⁶ cryptographic algorithm. This process takes input, (e.g.: a file, drive image, text string), and applies the MD5sum algorithm. The process further creates a numerical check sum that the algorithm generated while parsing the input as output. The output contains an alphanumeric string that represents the input as a “hash.” According to the IETF, “the difficulty of coming up with any message having a given message digest is on the order of 2 to the 128 power operations.” By comparing the MD5sum hashes of two instances of a file, drive image, or text string strengthens the confidence in identity. For instance, files with identical MD5sum hashes are most probably copies of the identical file.

We chose IDA Pro as our corporate tool to facilitate the disassembling and debugging of computer executables or binaries into computer language code or source code. This software allows us to step through a binary as it performs each of its program routines and processes. This becomes an invaluable tool during the analysis of what a program or binary does whenever executed.

Shelter is a stand-alone workstation that has USB support, a CDROM writer, and LS120 floppy drive. The stand-alone network configuration is preferable in these investigations. If the binary upon execution or debugging caused harm to the workstation by installing or executing malicious code, (e.g.: worm, IRC server, FTP server, etc.), other systems would not be affected.

I downloaded the unknown binary from the SANS Internet site as binary_v1.3.zip. This zip file contained a Windows executable file named target2.exe (ICMP BackDoor V0.1). An initial review of the file characteristics using windows explorer revealed the following:

1. Size 26,793 bytes
2. Size on disk 28,672 bytes
3. Created Thursday, February 20, 2003, 12:45:48 PM
4. Modified Thursday, February 20, 2003, 12:45:48 PM
5. Accessed Thursday, February 20, 2003, 12:45:48 PM

This initial information provided a cursory look at the file.

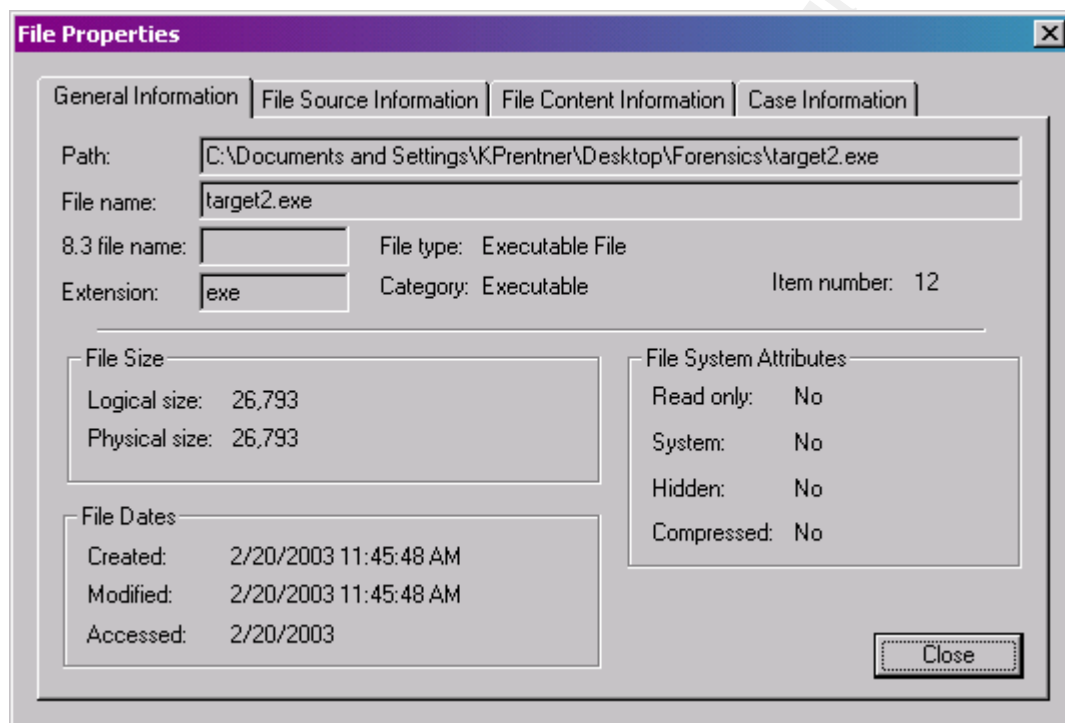
I utilized FTK to begin a more detailed analysis of the binary and verified the attributes listed above. First, I created a new case in FTK. Once created, we add articles of evidence to the case for analysis. Next, I accomplished just this by adding the target.exe as a case evidence item. FTK ensures evidence integrity by mounting the file without disturbing any evidence attributes or properties. Within FTK menu, we execute “Tools, File Properties.” This command produces a dialog box on the computer screen that reveals pertinent file information. The file information gathered from FTK in this manner includes:

⁶ <http://www.ietf.org/rfc/rfc1321.txt>

1. File Size = 26,793 bytes
2. Executable file type
3. Created Thursday, February 20, 2003, 12:45:48 PM
4. Modified Thursday, February 20, 2003, 12:45:48 PM
5. Accessed Thursday, February 20, 2003, 12:45:48 PM Read only = no, System = no, Hidden = no, Compressed = no

The following screen shot, 1.1.2, depicts the file properties from the “General Information” tab.

Screen Shot 1.1.2



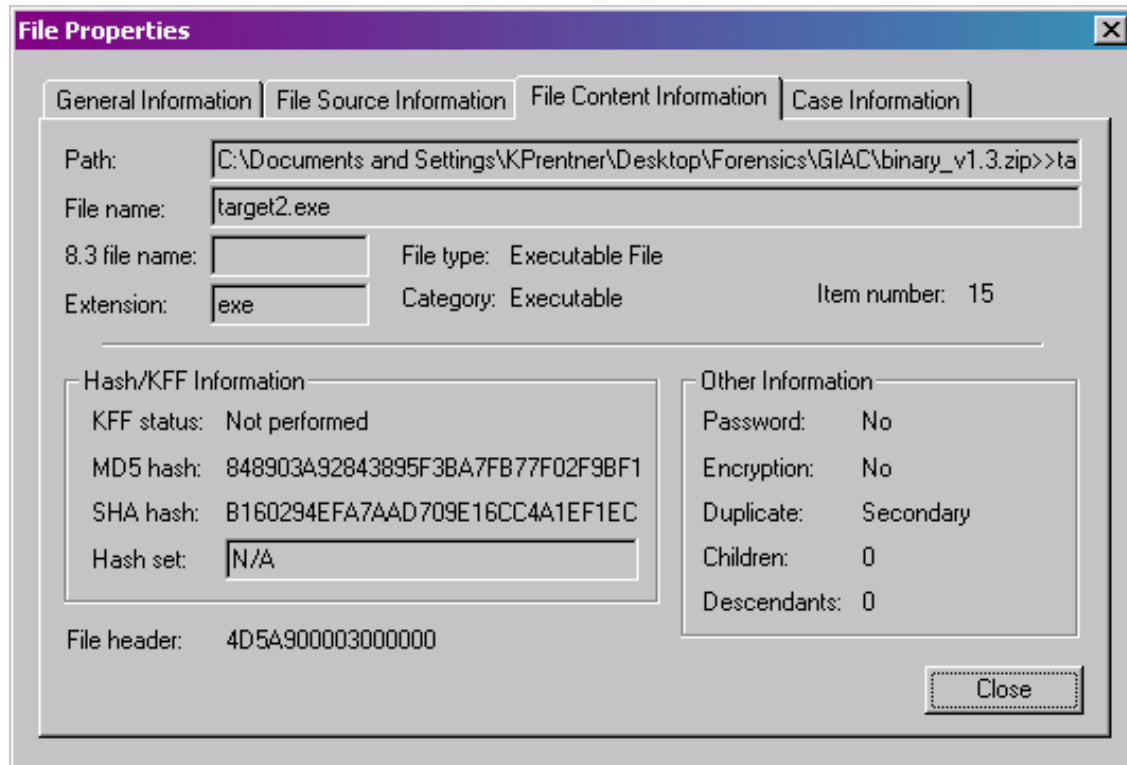
Notice that in both instances all three MAC times are identical. This poses an interesting scenario. The file has not been modified or accessed since creation. This binary has never been executed since it was compiled!

The “File Content Information Tab” reveals additional information of importance. This includes:

1. Password protection = no, encryption = no, children = no, descendants = no
2. MD5 hash sum
3. File header location

The following screen shot, 1.1.3, depicts the file properties from the "File Content Information" tab.

Screen Shot 1.1.3



Note that the MD5 hash verification when utilizing MD5sum:

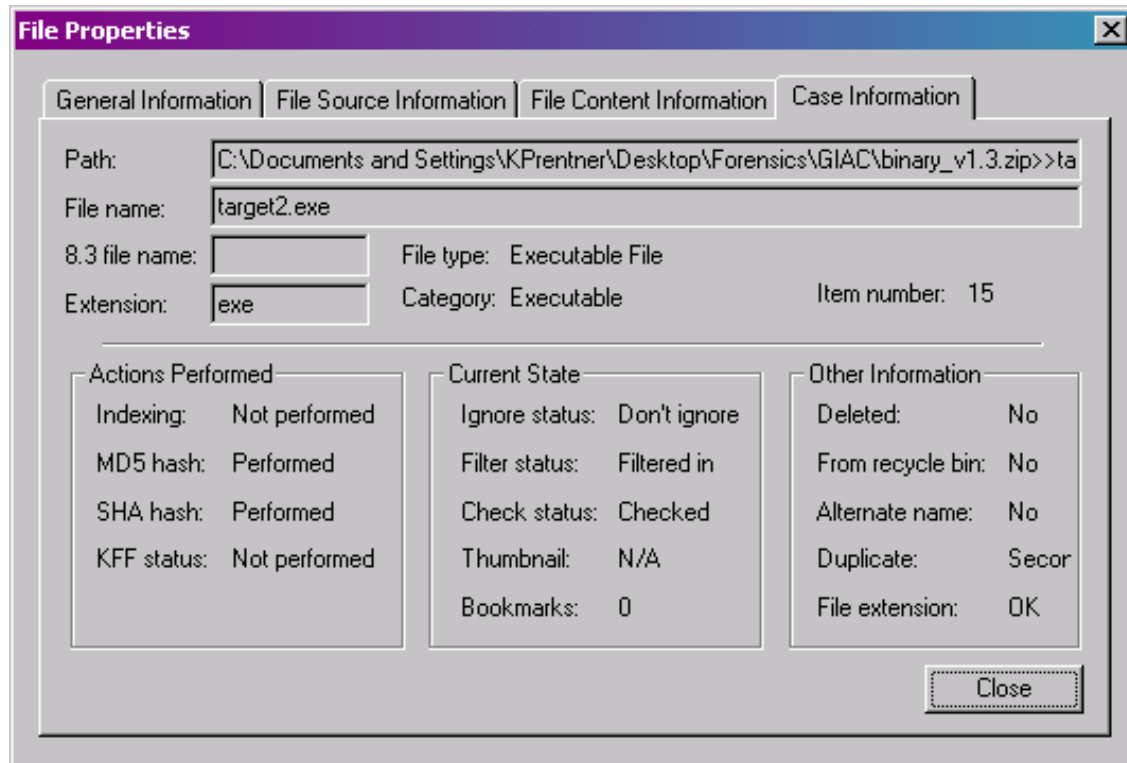
```
C:\>"C:\Program Files\Forensics\Acquire\forensic acquisition
utilities\bin\UnicodeRelease\md5sum.exe" "C:\Documents and
Settings\Shelter\Desktop\Forensics\GIAC\target2.exe"
848903a92843895f3ba7fb77f02f9bf1 *C:\\Documents and
Settings\\KPrentner\\Desktop\\Forensics\\GIAC\\target2.exe
```

Finally, more information retrieved from the file properties reveals the following:

1. File extension = OK. This verifies that the file is a WIN32 executable, matching the .exe file extension
2. Deleted = no, alternate name = no

The following screen shot, 1.1.4, depicts the file properties from the “Case Information” tab.

Screen Shot 1.1.4



We now have continued verification of the MS Explorer data. The file size confirmed at 26,793 bytes, a modification date of 2/20/2003 12:45:48, Read only = no, System = no, and Hidden = no.

FTK also contains a hex editor. A hex editor displays digital information as both ASCII text and Hexadecimal. ASCII is text, as you and I understand text, containing strings of alphanumeric characters that we interpret as language. Hexadecimal refers to the base-16 number system, which consists of the numbers 0 to 9 and the letters A to F. Hexadecimal creates an interpretation of computer binary zeros and ones that we humans find easier to understand. By using the hex editor, we can see the text strings within the target.exe binary! This facilitated the discovery of very interesting key words that included:

- “Hello form MFC!”
 - This is a default message produced by Visual C++ upon compilation of the code⁷

⁷<http://msdn.microsoft.com/visualc/>

- References to multiple Windows system files, implying that the program is dynamically linked and reliant on external libraries
- Calls to Windows systems binaries
 - CMD.exe – Windows command shell
 - Smsses.exe – Microsoft Service Control Manager
 - Reg.exe – Registry Consol tool included in Microsoft's developers toolkit
 - [\\199.107.97.191\C\\$](#) - A drive share on a remote system. This could be the remote connection attempt! Visiting that IP address yields: "Under Construction". According to ARIN WHOIS⁸, the IP address belongs to:

OrgName: CERFnet customer - Azusa Pacific University

OrgID: CCAPU-1

Address: 901 E. Alostia Ave.

City: Azusa

StateProv: CA

PostalCode: 91702

Country: US

1.2 Program Description and Forensic Details

The file is definitely an executable. FTK completed an extension analysis, which confirmed that the .exe extension correctly represents the file as a 32-bit Windows executable. These file types are executable in WIN 95, WINNT, and WIN 2000 and all other 32-bit Windows environments.

The hex editor within FTK allows us to look at the text within the source code of target.exe. Searching for interesting strings, we discovered the following:

```
impossible create raw ICMP socket%s RAW ICMP SendTo:
===== Icmp BackDoor V0.1 =====
Code by Spoof. Enjoy Yourself!
Your PassWord:lokicmd.exe
Exit OK!
Local Partners Access
```

This is very interesting! A program name "ICMP BackDoor V0.1," developer's credit "Spoof," user password, etc. More importantly, there are clues to assist in identifying the focus of the executable. ICMP and Loki, hmmm. ICMP is an interesting protocol. According to daemon9 && Alhambra in Phrack Magazine⁹ discuss how ICMP can be used to carry data within the ping packet, "ICMP_ECHO packets also have the option to include a data section. This data

⁸ <http://ws.arin.net/cgi-bin/whois.pl>

⁹ <http://www.phrack.com/show.php?p=49&a=6>

section is used when the record route option is specified, or, the more common case, (usually the default) to store timing information to determine round-trip times.” They go on to illustrate that each ping packet has a minimum of 8k of space available for extra data. Therefore, creating a “covert channel.” They continue their discussion to describe Loki. Characteristics of Loki include being able to utilize ICMP packets to carry commands onto a remote server e.g.: “cmd.exe echo hello world,” thus creating a “backdoor.” Loki can therefore be used to set up a peer-to-peer communications channel, “secretly channeling information” within the data of the ICMP packet. Because ICMP is a protocol often allowed through firewalls, firewalls do not usually protect against this type of an attack.

Other information of value revealed by FTK included a breakdown of the source code structure, identifying the types of system calls made to windows .dll files and the functions called. Further analysis required other than forensic tools. To analyze the code and step through the breaks the code takes while executing we utilize the decompiler, IDS. Through this process, we noted the following system flow:

1. Header info: Calls to Macro Support Library “IRPC.” Sets up program as a Windows 32-bit executable.
2. Main module: Establishes constants and establishes the services table. This is accomplished through an instruction set that attempts to establish a Service Control Handler that in turn could monitor Services status.
 - ADVAPI32.dll: RegisterServiceCtrlHandlerA
 - ADVAPI32.dll StartServiceCtrlDispatcherA
 - ADVAPI32.dll SetServiceStatus
 - ADVAPI32.dll QueryServiceConfigA
 - ADVAPI32.dll ChangeServiceConfigA
 - ADVAPI32.dll StartServiceA
 - ADVAPI32.dll OpenSCManagerA
 - ADVAPI32.dll CreateServiceA
 - ADVAPI32.dll OpenServiceA
 - ADVAPI32.dll QueryServiceStatus
 - ADVAPI32.dll ControlService
3. Subroutines accomplish the remaining programming directives through system calls
 - Socket establishment – calls. to establish the socket and set up host communications utilizing WS2_32.dll¹⁰: WSASStartup, SOCKET __stdcall and u_short_stdcall
 - Setup Communications attempted to 199.107.97.191 WS2_32.dll: int_stdcall gethostbyname and inet_address
 - Allocate resources WS2_32.dll: alloca_probe

¹⁰ http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/initialization_2.asp

- Heap allocations - KERNEL32.dll: HeapAlloc
- Initiating processes - KERNEL32.dll: StartProcess
- Create Pipes - KERNEL32.dll: CreatePipe
- Utilize Pipes - KERNEL32.dll: PeekNamedPipe
- Close Handle - KERNEL32.dll: CloseHandle
- End Processes - KERNEL32.dll: TerminateProcess
- Close Services - ADVAPI32.dll: CloseServiceHandle

The binary completes several functions besides running the main program. An interesting routine is the impersonation of another user. This is accomplished by utilizing the `setuser ()` function. This function establishes the user name for a session. In this case, the user is set to “matherr.” To anyone investigating the connectivity it would appear that “matherr” was the user utilizing the program. This is illustrated within the following line of code:

```
.idata:004030E4      extrn __setusermatherr:dword ; DATA XREF:
```

Other non-main function calls include initializing the environment, getting the main arguments, setting the application type, terminating the initialization, and an empty or null function. These are all standard programming calls that don not contribute directly to the main function yet compliment the program objective. The following lines of code are examples of these calls:

```
.idata:004030D8      extrn __imp__initterm:dword ; DATA XREF
```

```
.idata:004030DC      extrn __getmainargs:dword ; DATA XREF
```

```
.idata:004030E0      extrn __p__initenv:dword ; DATA XREF
```

The main function accomplishes the Loki server and client operations. The first calls set the server status and start the service center dispatcher. The lines of code associated with these calls are:

```
.idata:00403004 ; BOOL __stdcall StartServiceCtrlDispatcherA(const
SERVICE_TABLE_ENTRYA *lpServiceStartTable)
```

```
.idata:00403004      extrn StartServiceCtrlDispatcherA:dword
```

```
.idata:00403004      ; DATA XREF: _main+122□r
```

```
.idata:00403008 ; BOOL __stdcall SetServiceStatus(SERVICE_STATUS_HANDLE
hServiceStatus,LPSERVICE_STATUS lpServiceStatus)
```

```
.idata:00403008      extrn SetServiceStatus:dword ; DATA XREF:
.text:0040228E r
```

Next, there is a function to establish a connection to the service control on the specified server, to open a service, or close a service. These calls are enablers of win32 services, another clue that the program is WIN32 based. The following code accomplishes service establishment and closure:

```
.idata:00403020 ; SC_HANDLE __stdcall CreateServiceA(SC_HANDLE
hSCManager,LPCSTR IpServiceName,LPCSTR IpDisplayName,DWORD
dwDesiredAccess,DWORD dwServiceType,DWORD dwStartType,DWORD
dwErrorControl,LPCSTR IpBinaryPathName,LPCSTR IpLoadOrderGroup,LPDWORD
lpdwTagId,LPCSTR IpDependencies,LPCSTR IpServiceStartName,LPCSTR
IpPassword)

.idata:00403020          extrn CreateServiceA:dword ; DATA XREF:
sub_402320+5F  r

.idata:00403024 ; BOOL __stdcall CloseServiceHandle(SC_HANDLE hSCObject)

.idata:00403024          extrn CloseServiceHandle:dword ; DATA XREF:
sub_402320+190  r

.idata:00403024          ; sub_402320+199  r ...

.idata:00403028 ; SC_HANDLE __stdcall OpenServiceA(SC_HANDLE
hSCManager,LPCSTR IpServiceName,DWORD dwDesiredAccess)

.idata:00403028          extrn OpenServiceA:dword ; DATA XREF: sub_402320+AA  r

.idata:00403028          ; sub_4024D0+2C  r

.idata:0040302C ; BOOL __stdcall QueryServiceStatus(SC_HANDLE
hService,LPSERVICE_STATUS IpServiceStatus)

.idata:0040302C          extrn QueryServiceStatus:dword ; DATA XREF:
sub_402320+D1  r

.idata:0040302C          ; sub_402580+F5  r

.idata:00403030 ; BOOL __stdcall ControlService(SC_HANDLE hService,DWORD
dwControl,LPSERVICE_STATUS IpServiceStatus)

.idata:00403030          extrn ControlService:dword ; DATA XREF:
sub_402320+112  r
```

Once a service is established the service may either wait and listen or create a connection with the client. If network connectivity is established there is utilization of bind, getting the host name, creating a pipe and a network socket. These subroutines also facilitate the transfer and writing of a data. Code segments to accomplish these network connections are as follows:

.idata:00403044 ; BOOL __stdcall WriteFile(HANDLE hFile,LPCVOID lpBuffer,DWORD
nNumberOfBytesToWrite,LPDWORD lpNumberOfBytesWritten,LPOVERLAPPED
lpOverlapped)

.idata:00403044 extrn WriteFile:dword ; DATA XREF: sub_401EE0+4A r

.idata:00403048 ; BOOL __stdcall CreatePipe(PHANDLE hReadPipe,PHANDLE
hWritePipe,LPSECURITY_ATTRIBUTES lpPipeAttributes,DWORD nSize)

.idata:00403048 extrn CreatePipe:dword ; DATA XREF: sub_401CD0+D r

.idata:00403048 ; sub_401CD0+3A r ...

.idata:00403054 ; BOOL __stdcall PeekNamedPipe(HANDLE hNamedPipe,LPVOID
lpBuffer,DWORD nBufferSize,LPDWORD lpBytesRead,LPDWORD
lpTotalBytesAvail,LPDWORD lpBytesLeftThisMessage)

.idata:00403054 extrn PeekNamedPipe:dword ; DATA XREF:
sub_401CD0+103 r

.idata:00403054 ; sub_401CD0+12B r ...

.idata:00403060 ; HANDLE GetProcessHeap(void)

.idata:00403060 extrn GetProcessHeap:dword ; DATA XREF:
sub_4018C0+CF r

.idata:00403064 ; void __stdcall Sleep(DWORD dwMilliseconds)

.idata:00403064 extrn Sleep:dword ; DATA XREF:

There is also an exit routine for a clean closing of the network connection, services, and the program itself.

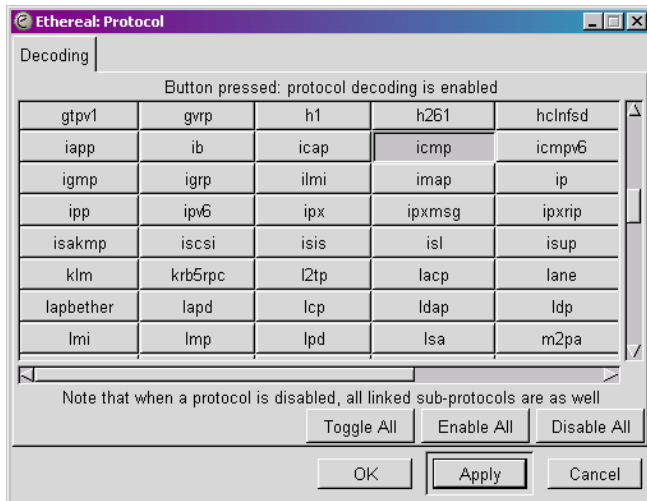
To identify if ICMP traffic is created by the binary I utilized Ethereal¹¹ and WinCap¹² to determine what, if any, network traffic the binary exhibits whenever executed. Ethereal is an open source network packet analyzer that is freely available on the Internet. Ethereal is specifically tuned for use within UNIX environments. WinCap creates architecture to capture the network packets and WinCap drivers facilitate Ethereal Win32 network analysis. WinCap allows us to capture network traffic on our Windows 2000 forensic workstation and it transparently exports the results to Ethereal. Both WinCap and Ethereal have been installed on our test system. We began by opening Ethereal. Next, we reviewed the nature of the network traffic created by the binary.

¹¹ <http://www.ethereal.com/>

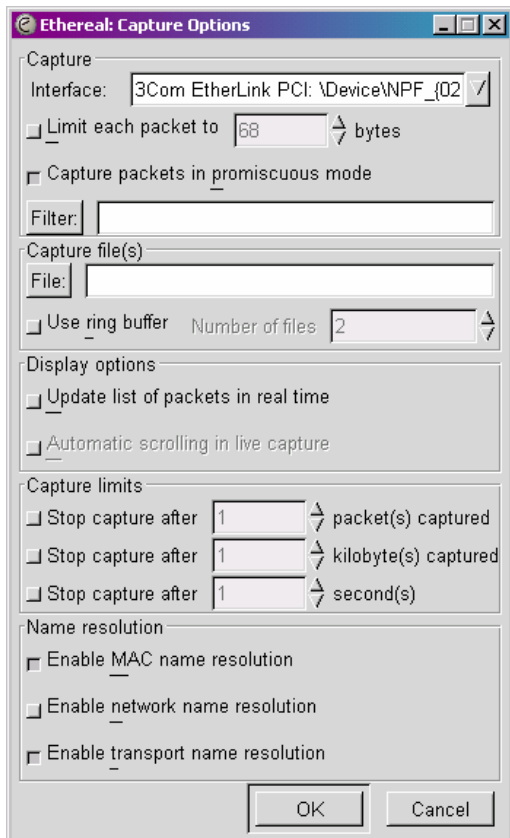
¹² <http://winpcap.polito.it/>

First, I created a baseline of the normal ICMP network traffic on the workstation by starting Ethereal and setting the product to only look for ICMP traffic and then beginning a capture. The Ethereal settings are illustrated by the following screenshots 1.2.1 and 1.2.2.

Screenshot 1.2.1

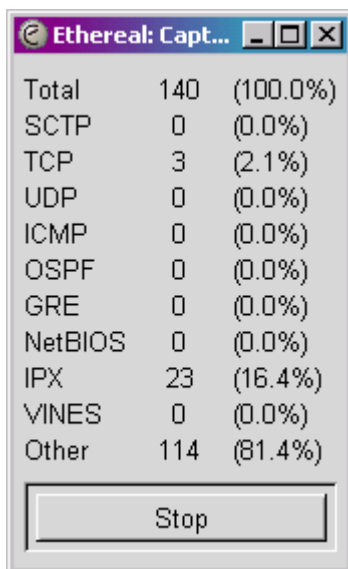


Screenshot 1.2.2



Once the baseline was established I executed the binary and began another packet capture with ethereal. I did this following the same procedures illustrated above. During the preliminary scan by Ethereal, an unknown protocol (a company proprietary one) was recorded. I illustrate this capture with the following screenshot 1.2.3.

Screenshot 1.2.3

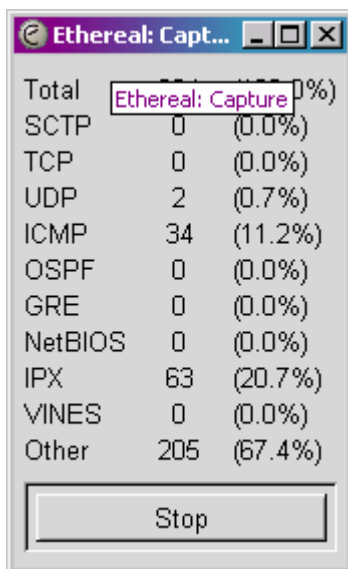


A screenshot of the 'Ethereal: Capt...' window. It displays a table of protocol statistics. The 'Total' row shows 140 packets (100.0%). The 'Other' row shows 114 packets (81.4%). A 'Stop' button is at the bottom.

Protocol	Count	Percentage
Total	140	(100.0%)
SCTP	0	(0.0%)
TCP	3	(2.1%)
UDP	0	(0.0%)
ICMP	0	(0.0%)
OSPF	0	(0.0%)
GRE	0	(0.0%)
NetBIOS	0	(0.0%)
IPX	23	(16.4%)
VINES	0	(0.0%)
Other	114	(81.4%)

Stop

The capture from the binary is quite different. We receive multiple ICMP connects and packets from our network interface card. Here we notice



A screenshot of the 'Ethereal: Capt...' window. It displays a table of protocol statistics. The 'Total' row shows 225 packets (100.0%). The 'Other' row shows 205 packets (91.1%). A 'Stop' button is at the bottom.

Protocol	Count	Percentage
Total	225	(100.0%)
SCTP	0	(0.0%)
TCP	0	(0.0%)
UDP	2	(0.7%)
ICMP	34	(11.2%)
OSPF	0	(0.0%)
GRE	0	(0.0%)
NetBIOS	0	(0.0%)
IPX	63	(20.7%)
VINES	0	(0.0%)
Other	205	(91.1%)

Stop

Finally, an attempt could be made to contact Azusa Pacific University to determine the owner of IP Address 199.107.97.191 and to determine if there is

any correlation to the names identified within the program source code. Those names are User: matherr, and the acclaimed writer of the software: Spoof

1.3 Program Identification:

As you recall the MD5 hash for the unknown binary that was provided for this example is as follows:

```
C:\>"C:\Program Files\Forensics\Acquire\forensic acquisition
utilities\bin\UnicodeRelease\md5sum.exe" "C:\Documents and
Settings\Shelter\Desktop\Forensics\GIAC\target2.exe"
\848903a92843895f3ba7fb77f02f9bf1
*C:\\Documents and Settings\\KPrentner\\Desktop\\Forensics\\GIAC\\target2.exe
```

I could not locate an exact duplicate of this binary on the Internet. I did locate many others: i.e: BackDoor – Q – ICMP, ICMP BackDoor (wrong one), icmp.c, icmpc.c, icmpbd, icmp pipe.c, etc, etc, etc. The MD5 checksums (nor file sizes) never matched. Therefore, this section requires further review.

1.4 Legal Implications

According to the United States Code, the person that installed and executed this binary on a protected system would be subject to arrest and prosecution. The first premise would be that the person took control over the computer, but did not aggravate the offence with a sex crime or piracy and that the computer did not belong to the government or one of its agencies. The punishable crime in this instance would fall under the United States Code: TITLE 18, PART I, CHAPTER 47, Sec. 1030.¹³ This part of the code covers fraud and related activity involving computers. The code states that anyone who:

“Knowingly causes the transmission of a program, information, code, or command, and as a result of such conduct, intentionally causes damage without authorization, to a protected computer; intentionally accesses a protected computer without authorization, and as a result of such conduct, recklessly causes damage; or intentionally accesses a protected computer without authorization, and as a result of such conduct, causes damage; and by conduct described caused loss to 1 or more persons during any 1-year period loss resulting from a related course of conduct affecting 1 or more other protected computers) aggregating at least \$5,000 in value”

Possession of the binary itself therefore does not constitute a crime. Proof that the code was installed and then executed within a “protected computer” system is required for successful prosecution. In addition, evidence leading to proof that

¹³ http://www4.law.cornell.edu/cgi-bin/muscat/highlighter/first_match

utilization of the binary lead to damages in excess of \$5000 is requisite to convict.

The punishment could be a fine under this title or imprisonment for not more than ten years, or both. For a second or other subsequent conviction, a fine under this title or imprisonment could be for not more than twenty years, or both.

In the case that, for some reason the person also used the computer to harbor, transmit, sell, or otherwise deal in pirated software, music, video, etc. the person would also be liable under the United State Code: TITLE 18, PART I, CHAPTER 113, Sec. 2318¹⁴. The code states:

“Whoever, in any of the circumstances described in this section, knowingly traffics in a... copy of a computer program or documentation or packaging for a computer program, or a copy of a motion picture or other audiovisual work... shall be fined under this title or imprisoned for not more than five years, or both”.

The person shall also be required to forfeit any an all pirated materials they should have control over.

Lastly, if the person had utilized materials that contained child pornography the person would be liable under the United States Code: TITLE 18, PART I, CHAPTER 110, Sec. 2252A. The code states that:

“Anyone that knowingly mails, or transports or ships in interstate or foreign commerce by any means, **including by computer**, any child pornography; knowingly receives or distributes any child pornography that has been mailed, or shipped or transported in interstate or foreign commerce by any means, including by computer; or any material that contains child pornography that has been mailed, or shipped or transported in interstate or foreign commerce by any means, including by computer; knowingly reproduces any child pornography for distribution through the mails, or in interstate or foreign commerce by any means, including by computer.”

These persons would be fined and be sentenced to no less that 15 years and no more than 30 years.

In conclusion, the penalties for simply hacking into a protected computer could be steep even at the minimum penalties.

1.5 Interview Questions

¹⁴ http://www4.law.cornell.edu/cgi-bin/muscat/highlighter/first_match

Make an appointment and keep it! Professionalism rewards itself. Pay attention, have someone else take notes for you. LISTEN.

The primary point during the interview is to keep in control and maintain the initiative. You are asking the questions not the other way around. Take advantage during the position of authority, at the very beginning, and leverage that position throughout the Interview.

A second strategy includes appropriately controlling the silent spaces between questions and answers. If the suspect is taking too much time, redirect the questioning. If you want to build the suspense, do not speak, stare at them in the silence, let them break eye contact, and then look off into the distance, occasionally revisiting a glance. Do not become too friendly or overly angry. Stay to the point and look for discrepancies in the subject's answers. Always, always remain courteous and polite.

Interview questions to consider when interrogating the subject that installed and executed the BackDoor include:

- The first questions should be used to create a dialogue. Begin with introductions and ask simple question to familiarize yourself with the subject. Be able to assess when they are uncomfortable by first determining how they act when they are relaxed. Describe to them the purpose of the interview and then ask them to describe what they think of the situation.¹⁵ (Walters) Remain friendly! The more "underground" the personality of the interviewee, the more friendly you must remain during the dialogue.
- Next, ask an open-ended question that allows the subject to narrate how they imagined the course of events to occur. Use this as an opportunity to determine the subject's credibility. A good question might be: How do you see the event to have played out? Pay attention to changes in the subject's behavior from the initial interview questions. (Walters)
- The log files on the computer show that your IP address was logged into the computer during the time of the incident. This includes initial connection, upload, and execution of the program, as well as subsequent incidents on the computer. Were you logged into the computer at x, y, or z times or could someone have spoofed your identity?
- OK, so we have determined that weren't logged in at that time. Could you help us to identify the person online at that time? Do you have any idea who could have been logged in at your IP?
- If and only if you are receiving NO cooperation through friendly means, a more aggressive approach may be necessary. I really haven't had

¹⁵ Interviewing for Credibility: Accurate Identification of Deception Behaviors by Stan B. Walters http://www.kinesic.com/interviewing_for_credibility.htm

adequate time to complete my review of the log files. I know I will get around to a complete review of the logs. We already have a court order to seize any computer that was logged in at that time to analyze the data on their hard drives. We really do not want to have to go that far, because it will probably be a waste of time. Since the log files show your IP to be logged in at the time of the incident, is there anything you can tell me about activity on the computer while you were logged in?

- If and only if the interviewee presents him or herself as cocky or overconfident and they are not cooperating at all, you could attempt some mild trickery. I don't know why we are interviewing you in the first place. You don't seem to have the technical expertise to accomplish the things that we have discovered so far. Perhaps you may know of someone that has access to your IP address and could have the technical expertise?
- We haven't gone to the authorities (management) with this so far. I really would like to handle this thing under the covers. Help me out here. What can you tell me about what happened? We can solve this thing together, go home, and forget about it!

1.6 Further Reading

- [1] Proctor, Norman E. and Neumann, Peter G. "Architectural Implications of Covert Channels." Proceedings of the Fifteenth National Computer Security Conference, pages 28-43, Baltimore, Maryland, October 13-16, 1992. URL: <http://www.csl.sri.com/neumann/ncs92.html>
- [2] RSA Laboratories. "What are covert channels." Cryptography FAQ, Version 4.1, Section 7.15. URL: <http://www.rsasecurity.com/rsalabs/faq/7-15.html>
- [3] van Hauser. "README." THC-UnixHackingTools #1, October 17, 1998. URL: <http://thc.pimmel.com/releases.htm> or <http://thc.pimmel.com/files/thc/thc-uh1.tgz>
- [4] Taran King. "Introduction to Phrack Inc." Phrack Magazine, Volume One, Issue One, File 1 of 8, November 17, 1985. URL: <http://phrack.infonexus.com/search.phtml?view&article=p1-1>
- [5] daemon9 and alhambra. "Project Loki: ICMP Tunnelling." Phrack Magazine, Volume Seven, Issue 49, File 6 of 16, August 1996. URL: <http://phrack.infonexus.com/search.phtml?view&article=p49-6>
- [6] Route (daemon9). "LOKI2 (the implementation)." Phrack Magazine, Volume 7, Issue 51, Article 6 of 17, September 1, 1997. URL: <http://phrack.infonexus.com/search.phtml?view&article=p51-6>
- [7] Rowland, Craig H. "Covert Channels in the TCP/IP Protocol Suite," November 14, 1996. URL: <http://www.psionic.com/papers/covert/covert.tcp.txt>
- [8] Simple Nomad. "Strategies for Defeating Distributed Attacks," 1999. URL: http://packetstorm.securify.com/papers/contest/Simple_Nomad.doc
- [9] edi and teso. "ICMP tunneling tool," July 10, 1999. URL: <http://teso.scene.at/releases.php3>, http://teso.scene.at/releases/itunnel-1_2.tar.gz

- [10] Tetsu Khan. "Welcome To Issue 1 Of Confidence Remains High." Confidence Remains High, Issue 1, April 16, 1997. URL: <http://www.hackersclub.com/km/magazines/crh/crh001.txt>
- [11] BiT. "ICMP backdoor client and server." Confidence Remains High, Issue 9, May 11, 1998. URL: <http://www.hackersclub.com/km/magazines/crh/crh009.txt>
- [12] sil. "Q and A with Team S0ft Project." Anti Offline. June 20, 2000. URL: <http://www.antioffline.com/s0ftpj.html>
- [13] SMaster and \sPIRIT\ "s0ftpr0ject 2000," April 21, 2000. URL: <http://www.s0ftpj.org/en/site.html>
- [14] FuSyS. "Progetto Ninja." Butchered From Inside, Number 4, Year 1, December 1998. URL: <http://www.s0ftpj.org/bfi/bfi4.tar.gz>
- [15] van Hauser. "Placing Backdoors Through Firewalls". v1.5, May 1999. URL: <http://thc.pimmel.com/files/thc/fw-backd.htm>
- [16] Brinkhoff, Lars. "GNU httptunnel." August 31, 2000. URL: <http://www.nocrew.org/software/httptunnel.html>
- [17] vecna. "B0CK." Butchered From Inside, Number 7, Year 2, December 25, 1999. URL: <http://www.s0ftpj.org/bfi/online/bfi7/bfi07-12.html>
- [18] Vidstrom, Arne. "ACK Tunneling Trojans," 1999-2000. URL: <http://www.ntsecurity.nu/papers/acktunneling>
- [19] SANS.org. URL: http://www.sans.org/infosecFAQ/covertchannels/covert_shells.htm

© SANS Institute 2003, Author retains full rights.

Part Two – Option One

Forensic Analysis of a Made Windows NT & IIS Internet Server

© SANS Institute 2003, Author retains full rights.

2.1: Synopsis

During November 2002, the Information Security Department tested a Host Based Intrusion Detection System (HIDS) in the Information Security Lab. The HIDS software tested was Okena StormWatch¹⁶; a low level agent and consol that intuitively learns a server's behavior and creates rules to allow only bonafide transactions, services, and protocols. Okena also utilizes canned rules with already established definitions for most viruses, Trojans, backdoors, denial of service attacks, etc. After completing the testing on a sanitized version of a product, "Product A" belonging to a third party company, Information Security recommended and received management approval to apply the HIDS solution with the rule base created on the sanitized server to the product's server farm within our Internet network segment.

Almost immediately after installing the Okena agents on "Product A" servers, Information Security received an alert describing unauthorized traffic from a server within the "Product A" server farm. The HIDS was successfully blocking this traffic from one of the "Product A" Internet servers. Upon close analysis, Information Security determined the suspicious behavior closely resembled IP traffic created by Internet Relay Chat (IRC). At this point, Information Security notified management and called into action the Information Security Incident Response Team (ISIRT).

Members of ISIRT interviewed the appropriate management. Management determined that the server must remain on-line. Taking the server off-line potentially affects a critical customer Service Level Agreement (SLA). Management directed Information Security to analyze the system "live" and leave the server on the Internet as long as feasibly possible.

I completed a live capture of running processes, an image of physical memory, open processes and services, network connections, login records, and an image of the physical media. We returned to the Information Security lab to begin analysis of the data. An expedient and thorough reporting¹⁷ of the forensic findings would provide the necessary information management requires to determine what should happen to the "Product A" server farm, as well as other network devices and servers on the same network segment.

The findings included information in reference to the server's vulnerabilities, proof of at least one unauthorized user taking control of the server, installed malicious software on the server, a compressed and fragmented copy of the feature film – "Men in Black 2", pornographic anime video files, and transmission of potentially illegal Internet traffic originating from the server with multiple random Internet

¹⁶ http://www.okena.com/areas/products/products_stormwatch.html

¹⁷ Appendix A: example of our reporting style

destinations. Management used this information and decided to expedite a re-imaging of the “Product A” server farm. Server Engineering rebuilt the servers and placed them back into production, complete with the HIDS solution. Information Security completed a vulnerability assessment of the new server images and determined that they were free of known exploits.

2.2: System Environments

Our company provides customers with a secure hosting environment in a company owned World class Data Center (DAT). Redundant environmental requirements include Tier I Internet connections provided by two telecommunications companies (TELECO). Independent Sonnet rings connect our network to the TELECO backbones.

2.2.1: Victim Environment:

The “Product A” server farm location within the DAT includes a secure cage that is only accessible by predetermined “Customer A” employees, our company’s DAT support staff, and our company’s Information Security Department. All access requires controlled entrance through a biometrically locked mantrap. The servers are all rack mounted, well labeled, and diagrammed. We obtained the following information about the “Product A” victim server through interviews, network topology mappings, and first hand identification and investigation:

- Primary Internet store front for “Product A” belonging to “Company A” and managed by my company
- Operating System: Widows NT 4.0, no service releases
- System software: IIS version 4.0, no service releases
- Server name: Product A Server 1
- Host name: productA.companyA.com
- Internet Host IP Address: 216.xxx.xx.xxx
- Triple 10/100 interface
 - One Internet interface for Web Page front end
 - One internal private network interface for server management
 - One Internal private network interface for Domain infrastructure

2.2.2: Forensic Environment:

Information Security has created a Forensics investigative environment that includes a sanitized workstation in the Information Security lab, two sanitized forensic mobile laptops, and multiple copies of a sanitized and

validated “Batchux Fire”¹⁸ CD. FIRE is a portable bootable cdrom based distribution with the goal of providing an immediate environment to perform forensic analysis, incident response, data recovery, and virus scanning and vulnerability assessment. A secure known environment is necessary because an authorized user could have corrupted binaries (such as CDM.exe, dd.exe, nc.exe) on the server. The CD boots to a self contained Linux operating system that does not mount the hard drive operating system. This helps to preserve the integrity of the forensic evidence on the victim system. Two different modes are available for FIRE usage. The first mode is relevant to this paper and utilized whenever the server remains on-line and nor disconnect from the network or shut down. Often, business considerations warrant a server remaining on-line. In these cases, we utilize FIRE by inserting the CD into the Cdrom drive of the victim server in order to access a reliable and secure command shell. The CD also contains most every binary that a forensic analyst would need to complete an initial investigation and digital media capture or image. We describe these techniques later in this paper. The second scenario is for use on compromised systems taken off-line. After placing the FIRE CD in the Cdrom drive, we reboot the server. During system startup, interrupt the process to enter “system setup” or BIOS. “System setup” or BIOS controls how the hardware behaves before operating system initiation. Ensure that boot to Cdrom is the first choice within boot options. Next, we save the BIOS settings and proceed with the server boot. At this time, a LINUX operating system boots self-contained within the FIRE CD in order to create a secure and reliable environment in which to perform forensic image captures and analytical techniques. Information Security verified the integrity of the sanitized forensic workstation, laptops, and Batchux CDs with checksums provided by the software vendors.

Detailed information on all forensics systems includes:

- Dual Boot Operating System: Windows 2000 Professional SP3
Red Hat Linux 8.0
- 10/100 network interface
- Two 120 Gigabyte external Iomega hard drives
- SCSI interface with cables
- CAT 5 network cables
- Four port hub

2.3: Victim System Hardware

¹⁸ FIRE is a portable bootable cdrom based distribution with the goal of providing an immediate environment to perform forensic analysis, incident response, data recovery, and virus scanning and vulnerability assessment. <http://fire.dmzs.com/>

Information Security confiscated no hardware during this forensic investigation. Management used two criteria in determining the status of the hardware. First, the customer SLA requirement and the risk of down time were significant to management. Second, management foresaw no possibility of litigation and determined not to seize the hardware.

Cataloguing and tagging of the equipment occurred through thorough documentation of the evidence scene with well-labeled photographs, documentation of DAT floor and rack locations, server ID numbers, and IP addresses associated with the victim server. I have created a chain of custody form¹⁹ to facilitate the documentation of evidence. This form contains viable information about the incident and system, e.g.: Forensic examiner name, pertinent times, evidence tag numbers, and a log for documenting when forensic procedures are performed. We labeled the victim server and the rack with forensic tags, according to ISIRT procedures, to further document the evidence. Table 2.4.1 illustrates victim system hardware:

Table 2.4.1

Victim Hardware Information			
Tag Number: December 2002 00001 Date: DEC XX, 2002 Time: 7:32 AM			
Server Name	Location	Front End Switch Port	Front End IP Address
"Product A" Server 1	3N 6TD5 SLT3	MTSC-FI2-1 e2/1	216.xxx.xx.xxx
Back End Switch Port	Back End IP Address	Remote Access Switch Port	Remote Access IP Address
MTSC-FI2-1 e2/3	10.x.xxx.xx	RCM-FI3-1 e2/10	10.x.xxx.xx
Vendor	Model	Serial #	OS
Compaq	ProLiant 1850R	D140JZG1K579	NT 4.0

The Compaq ProLiant server with tag number "December 2002 00001" has the following characteristics:

- Intel Pentium III 600 MHz (Dual Processor)
- One Gigabyte 100-MHz registered SDRAM ECC Physical memory
- Compaq 10/100 TX PCI UTP Ethernet Controller

¹⁹ Appendix B: Chain of Custody Form

- Two Intel 10/100 Ethernet Controllers
- Dual-channel Wide-Ultra SCSI 3 controller
- Three Hot Swappable 9.1-GB Wide Ultra3 SCSI 10,000 rpm Drives (mirrored)
- One 22X IDE CDROM drive
- One 1.44 Megabyte floppy drive

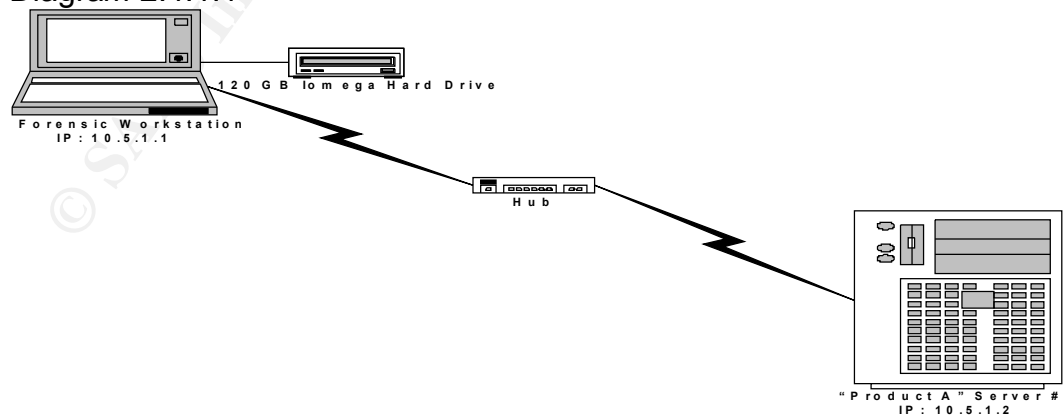
2.4: Creating the Images

Capturing the forensics information on the live system required creating a network connection between the forensic computer and the victim server. To accomplish this, an Information Security colleague and I traveled to the DAT. We generally work in pairs with one of us completing the manual tasks and the other creating documentation and ensuring that procedures are followed to establish a proper trail of evidence. I have created a procedure for creating images²⁰ to facilitate documentation and the following of procedures.

2.4.1: Network Setup

Establishing the network connection included connecting a forensics laptop (FL1) to the victim server, utilizing a DAT crash cart as a workplace for FL1 and other forensic hardware. We placed FL1, a four-port hub, and a 120 Gigabyte Iomega external hard drive on the crash cart. We located the crash cart adjacent to the “Product A” server rack. We connected FL1 to the hub using a CAT 5 patch cable, unplugged the victim server’s backend network cable, and connected this network interface with the hub using another CAT 5 patch cable. We then configured FL1 to have a victim server compatible 10.x.x.x IP address, thus completing the network connection. Diagram 2.4.1.1 illustrates the configuration used to interconnect the Victim server with the forensic workstation.

Diagram 2.4.1.1



²⁰ Appendix C: Procedures for Forensics Data Collection

2.4.2: Media Setup

We next prepared the Iomega drive for transfer of data files and images. First we created a 9.1 gigabyte partition for the hard drive image, a one gigabyte partition for the physical memory image, and a 100 megabyte partition for reports, files generated by data gathering utilities, and forensics notes. Next, we sanitized these partitions. To accomplish this we used DD, a Unix tool used for bit-wise data transfers among other things, to write zeros (00000000) on every bit of the partitions. We accomplished this by writing the Linux device *dev/zero* to each bit using the following syntax:

```
dd if=/dev/zero of=/mnt/iomega120/product_a/9.1_part
dd if=/dev/zero of=/mnt/iomega120/product_a/phy_mem_part
dd if=/dev/zero of=/mnt/iomega120/product_a/misc_part
```

2.4.3: Gathering Forensic Data

Capturing the forensics information on the live system required a prioritization of resources based on order of volatility. We first placed a FIRE CD in the victim server CDROM drive. The next step was to capture information on running processes, open files, and network connections. To ensure the integrity of the commands we utilized commands on the pre-verified FIRE CD. We executed all commands using Cmd.exe shell on the FIRE CD. We ran a batch file that I created and placed on the FIRE CD that automates the collection of live system information. The following are the contents of report.bat:

```
ads.exe C: -s > A:\ads.txt
arp.exe -a > A:\arp.txt
date.exe > A:\date.txt
env.exe > A:\environment.txt
fport.exe > A:\ports.txt
handle.exe > A:\handle.txt
hostname.exe > A:\host_name.txt
id.exe > A:\id.txt
listdlls.exe > A:\dlls.txt
netstat.exe -a -e -r -s > A:\netstat.txt
ntfsinfo.exe > A:\filesystem.txt
NTLast.exe > A:\last_on.txt
pslist.exe > A:\processes.txt
uname -a > A:\architecture.txt
whoami.exe > A:\who_am_i.txt
uptime.exe > A:\uptime.txt
psinfo.exe > A:\system_info.txt
psloggedon.exe > A:\logged_on.txt
psservice.exe > A:\services.txt
```

```
dir C:\ /S /AH /TA > A:\hidden_C.txt
dir D:\ /S /AH /TA > A:\hidden_D.txt
sniffer.exe > A:\sniffer.txt
mdmchk.exe > A:\modem.txt
sigs.exe > A:\signatures.txt
```

After placing a formatted floppy disk into the victim floppy drive, we utilized the following syntax for collection of the live system information:

Victim Server Syntax:

```
D:\statbins\win32\reports.bat
```

I created a MD5 hash²¹ during the creation and transfer of all images. MD5 hashes are an alphanumeric representation of electronic data generated by utilizing a very complex cryptographic algorithm, which is well beyond our forensics scope. This was accomplished utilizing the MD5sum arguments associated with DD. Next, I created a MD5 hash of the completed images that was transferred to the Iomega drive and compared each hash pair, to ensure the completeness and validity of each image file.

Further analysis of our process has shown that in the future the MD5 hashes should be created to a formatted floppy disk instead of to the victim file system hard drive. By doing so, we created an integrity problem. We potentially contaminated the evidence by writing to the victim fixed media. In the future, we plan to mitigate this situation by adding the following steps:

1. We have purchased a write blocker hardware device that will be installed whenever we complete a direct drive to drive image
2. All network based drive images will create MD5sums to a formatted floppy disk inserted into the victim floppy drive.

Due to the order of volatility, I next created an image of the Physical memory. I utilized the following syntax to create the image and associated MD5 hash:

FL1 Syntax:

```
/mnt/cdrom/statbins/linux2.2_x86/.nc -l -p 80 >
⇒ /mnt/iomega120/product_a/phy_mem_part/memory.img
```

Victim Server Syntax:

```
D:\statbins\win32\dd.exe if= \\.\PhysicalMemory --md5sum --verifymd5
⇒ --md5out=C:\memory.img.md5 | nc.exe 10.xxx.xx.xxx 80
```

²¹ Detailed MD5 hash information: <http://www.ietf.org/rfc/rfc1321.txt>

The following table, 2.4.3.1, illustrates the output from this command:

Table 2.4.3.1

Forensic Acquisition Utilities, 3, 16, 2, 1029 dd, 3, 16, 2, 1029 Copyright (C) 2002 George M. Garner Jr. Command Line: dd if=\\.\PhysicalMemory Based on original version developed by Paul Rubin, David MacKenzie, and Stuart Kemp Microsoft Windows: Version 5.0 (Build 2195.Server Service Pack 3) 12/12/2002 15:30:51 (UTC) 12/12/2002 10:30:51 (local time) Current User: MTMPP01\DATOperator Total physical memory reported: 1048092 KB Copying physical memory... D:\NT\bin\dd.exe: Stopped reading physical memory: The parameter is incorrect. The parameter is incorrect. 262139+0 records in 262139+0 records out

The imaging of the memory appears to be complete, but there are error messages in the results. We expect to receive the error messages, “The parameter is incorrect.” This message is returned whenever DD meets the end of the physical memory or the end of the memory file.²² (Rob Lee, SANS Course Book) In the future, we will avoid returning this error message by adding conv=noerror at the end of the command line on the victim system.

Next, we created an image of the boot sector utilizing the following syntax:

FL1 Syntax:

```
/mnt/cdrom/statbins/linux2.2_x86/.nc -l -p 80  
⇒ > /mnt/iomega120/product_a/misc_part/boot.img
```

Victim Server Syntax:

```
D:\statbins\win32\dd.exe if=\\.\PhysicalDrive0 bs=2048 count=2 --md5sum  
⇒ --verifymd5 --md5out=C:\boot.img.md5 | nc.exe 10.xxx.xx.xxx 80
```

Finally, we created an image of the hard drive utilizing the following syntax:

FL1 Syntax:

```
/mnt/cdrom/statbins/linux2.2_x86/.nc -l -p 80  
⇒ /mnt/iomega120/product_a/9.1_part/drive.img
```

²² SANS Course Book - System Forensics, Investigation, and Response

Victim Server Syntax:

```
D:\statbins\win32\dd.exe if=\\.\PhysicalDrive0 --md5sum --verifymd5  
⇒ --md5out=C:\9.1_drive.img.md5 | nc.exe 10.xxx.xx.xxx 80
```

The drive in question actually contained two logical drive partitions and was imaged using both PhysicalDrive0 and PhysicalDrive1. The following tables, 2.4.3.2 and 2.4.3.3, illustrate the output from these commands:

Table 2.4.3.2

Forensic Acquisition Utilities, 3, 16, 2, 1029 dd, 3, 16, 2, 1029 Copyright (C) 2002 George M. Garner Jr.	
Command Line: dd if=\\.\PhysicalDrive0 Based on original version developed by Paul Rubin, David MacKenzie, and Stuart Kemp Microsoft NT Windows: Version 4.0	
12/12/2002 16:23:19 (UTC) 12/12/2002 11:23:19 (local time)	
Current User: MTMPP01\DATOperator	
Disk: Compaq Disk Array (S/N) Geometry:	
Cylinders:	878
Tracks per Cylinder:	255
Sectors per Track:	32
Bytes per Sector:	512
Total Size:	3582240 KB
Media Type:	Fixed hard disk media
Partition Information:	
Partition Count:	4
Signature:	11113D57
Partition:	2
Starting Offset:	00000000023dc000
Length:	0000003630612480
Type:	IFS
Bootable?	Yes
Copying \\.\PhysicalDrive0 to CONOUT\$...	
895560+0 records in 895560+0 records out	
12/12/2002 18:45:21 (UTC) 12/12/2002 13:45:21 (local time)	

Table 2.4.3.3

```

Forensic Acquisition Utilities, 3, 16, 2, 1029
dd, 3, 16, 2, 1029
Copyright (C) 2002 George M. Garner Jr.

Command Line: dd if=\\.\PhysicalDrive1
Based on original version developed by Paul Rubin, David MacKenzie, and Stuart Kemp
Microsoft Windows: Version 5.0 (Build 2195.Server Service Pack 3)

12/12/2002 19:05:31 (UTC)
12/12/2002 14:05:31 (local time)

Current User: MTMPP01\DATOperator

Disk: Compaq Disk Array (S/N )
Geometry:
  Cylinders:      1298
  Tracks per Cylinder: 255
  Sectors per Track: 32
  Bytes per Sector: 512
  Total Size:      5295840 KB
  Media Type:      Fixed hard disk media

Partition Information:
  Partition Count: 4
  Signature:       11113D59

  Partition:       1
  Starting Offset: 0000000000004000
  Length:          0000005422923776
  Type:            LDM
  Bootable?        No

Copying \\.\PhysicalDrive1 to CONOUT$...

1487220+0 records in
1487220+0 records out

12/12/2002 23:45:22 (UTC)
12/12/2002 18:45:22 (local time)

```

The following table, 2.4.3.4, contains MD5 hash output that validates each of the images:

Table 2.4.3.4

```

\b7806f68f766631a84d04dea0d4b163c *\\.\PhysicalMemory
\b7806f68f766631a84d04dea0d4b163c *//mnt/iomega120//product_a//phy_mem_part//memory.img

\17b5cbd2c7a2f95077911da902506193 *\\.\PhysicalDrive0
\17b5cbd2c7a2f95077911da902506193 *//mnt/iomega120//product_a//9.1_part//drive0.img

\e7446e7fa0b612dea56a93c69c7aa3c8 *\\.\PhysicalDrive1
\e7446e7fa0b612dea56a93c69c7aa3c8 *//mnt/iomega120//product_a//9.1_part//drive1.img

```

2.5: Media Analysis

2.5.1: The Forensic System

The system I used for creating the data images at the DAT and analyzing the images for forensic details is a Toshiba 8100 laptop. Pertinent information about this forensic laptop, FL1 is contained within table 2.5.1.1:

Table 2.5.0.1

```
COMPUTERNAME=FL1
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 6 Model 8 Stepping 6, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=0806
USERDOMAIN=Local
Physical Memory=1.0 gigabyte
Hard Drive=Toshiba 11 gigabyte
CDROM Drive
LS 120 floppy drive
OS=WINDOWS NT
OS= Red Hat Linux 8.0
10/100 USB Ethernet adapter
120 gigabyte USB Iomega external hard drive
```

2.5.2: Report Analysis and System Analysis

The first step I took in the analysis process was to read the output contained within each of the report files generated. We accomplished the report analysis process during the creation and network transfer of the 9.1-gigabyte hard drive images. The network Netcat transfer of the image duration lasted several hours and allowed sufficient time to complete this analysis. Some of these reports contained very revealing information:

1. The command "date" revealed a ten minutes difference when comparing the server to GMT. Further findings, such as evaluating Mac times, require a ten-minute adjustment for valid reporting.
2. The command env unveiled important system information that is illustrated in table 2.5.2.1

Table 2.5.2.1

```
!::=:\
!C:=C:\
!EXITCODE=00000000
ALLUSERSPROFILE=C:\Documents and Settings\All Users
APPDATA=C:\Documents and Settings\DATOperator\Application Data
COMMONPROGRAMFILES=C:\Program Files\Common Files
COMPUTERNAME=MTMPP01
COMSPEC=C:\WINNT\system32\cmd.exe
HOMEDRIVE=C:
HOMEPATH=\
LOGONSERVER=\\MTMPP01
NUMBER_OF_PROCESSORS=2
OS=Windows_NT
OS2LIBPATH=C:\WINNT\system32\os2dll;
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
```

3. The command `dir /S /AH /T` did not reveal anything of forensic interest except the verification and location of hidden log files on the system. See table 2.5.2.2 for this information:

Table 2.5.2.2

Directory of C:\WINNT\system32\config		
12/06/2002 08:00p	1,024	default.LOG
12/11/2002 04:20p	1,024	SAM.LOG
12/12/2002 12:10a	1,024	SECURITY.LOG
12/12/2002 01:09p	1,024	software.LOG
02/04/2002 08:17a	1,024	system.LOG
08/23/2000 07:00p	0	TempKey.LOG
02/04/2002 08:17a	1,024	userdiff.LOG
7 File(s)	6,144	bytes

4. The command `pslogged on` revealed that the only accounts logged on were the DATOperator user account.
5. The commands `netstat` and `ports` revealed only appropriate connections and ports were either active or listening at the time of the report generation.
6. The commands `pslist` (lists open processes) and `psservice` (lists open services) revealed a very interesting service running, Tkbot.R00t!²³. Our server had been hacked and a root kit installed!
7. The command `uptime` revealed that the system had been operating 16 day(s), 14 hour(s), 54 minute(s), and 35 second(s).
8. The command `sigs` did not reveal any altered file extensions
9. The `tree` command produced extensive output. While perusing the tree, looking for directories that begin with ".", "//", " ", etc. I discovered a very interesting directory structure. This structure illustrated an interesting signature after expanding the directory tree. Within this structure, I also discovered a bootleg copy of "Men in Black 2." The movie contained fragmented .mpg movie files, compressed as .rar files. What fun we will have discovering the timelines and other neat items associated with those files!

While still waiting for the images to process, we began analysis of the log files. The only revelation produced was a repetitive attempt to connect to random non-company IP addresses and spawn a command shell on those

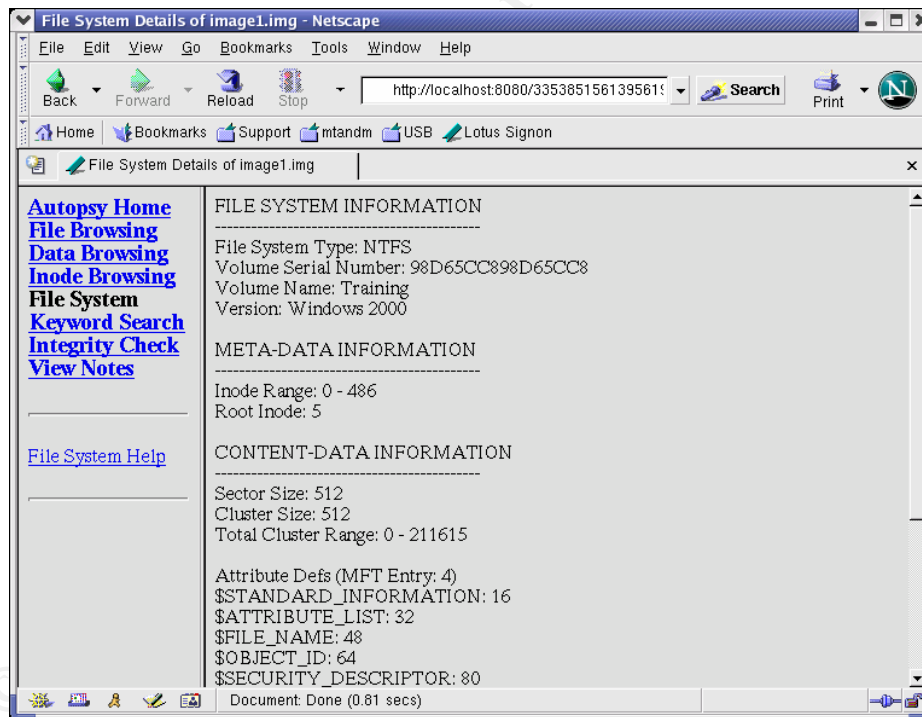
²³ A popular hacker tool, i.e. Root kit, installed as a service and therefore started automatically whenever the system starts up.

machines. We definitely found the type of behavior that would not occur naturally on this type of server. We discovered our first forensic evidence! It appears we have had an intruder operating from our system.

Finally, our imaging is complete. We have checked the hash sums and determined that the images were excellent bit-by-bit copies of the memory and hard drive partitions. The remainder of the analysis will occur in the Information Security lab.

Additional system information is available through the Autopsy Forensic Browser²⁴. All Autopsy Screen shots utilized in the following sections are examples and not actually from this case. We did not take screen shots while performing our forensic examination and management has sealed the case. Therefore, the images from the case were unavailable to replicate the screen images within autopsy. An example of system information revealed by Autopsy follows in Screen Shot 2.5.2.3.

Screen Shot 2.5.2.3



²⁴ Freeware: The Autopsy Forensic Browser is a graphical interface to the command line digital forensic analysis tools in the @stake Sleuth Kit (TASK). Together, TASK and Autopsy provide many of the same features as commercial digital forensics tools for the analysis of Windows and UNIX file systems (NTFS, FAT, FFS, EXT2FS, and EXT3FS).

This information reveals file system information such as NTFS, Operating System version number, Volume name, volume serial number, number of inodes, root inode number, and volume sign calculations.

2.5.3: Image Analysis

Upon arrival at the Information Security lab, we set up shop connecting FL1 to a docking station to ease the use with a monitor and mouse. The first step before analyzing the images is to mount each image as a loopback device. This creates a virtual view of the contents of the image as if it were an actual physical memory or hard drive device. The following switches are important whenever running the mount command to create a loopback device:

- loop: creates the loopback device
- ro: read-only, ensures that the image is not modified during the investigation
- noexec: will not allow execution of binaries within the image
- nodev: ignore all device files within the image
- noatime: do not update inodes atimes

I utilized the following syntax to mount the images:

```
Mount -o -loop,ro,nosuid,nodev,noexec,noatime  
⇒ /mnt/iomega120/product_a/9.1_part/drive.img /mnt/hack
```

Once mounted, I began examining the images. First I searched for e-mail files by grepping *.pst, and examining the Internet history file. No evidence of e-mail appeared to be on the victim server.

Next, I completed an analysis of the Internet history. First, I attempted to locate Internet history data files by utilizing the grep command from the FIRE CD. Searching for all *.dat file and specifically index.dat. within the mounted image returned no results. Further analysis of the file system through searching for Internet, IE, Netscape, and opera also yielded zero results. It appeared that there was no Internet browser installed on the server. We verified this through interviews with the personnel that installed the server. No browser had been installed with or after the operating system. Even though they had not taken the time to update the installation with service releases, they took the time to customize the server installation to include only certain packages within the operating system.

On to looking at the recycle bin, I changed to the recycled directory and perused the sid subdirectories to discover that all three were empty except

for the hidden INFO and INFO2 files. I opened each of the INFO2 subdirectories in a hex and discovered that they were also empty.

While examining the startup files, I discovered two interesting entries in the autoexec.bat file. There was a line to execute lroffer.exe²⁵ and another line to execute servu.exe²⁶. Not only was there a root kit discovered running as a service, our victim server possibly had an FTP server and an IRC server installed and set to execute each time the server restarted!

Things were definitely getting very interesting!

My next step was to examine the images for modified system executables. Accomplishing this task included identifying the inode mtime for every system executable, e.g.: cmd.exe, ftp.exe, etc. Completing this comparison, I determined that all interesting system executables have the same date and installation time as the Windows NT operating system. This is not to say that a gifted intruder could not use the touch²⁷ command to alter the inode atime, or mtime, for any files on the computer.

Running a Trojan scanner called chkroot²⁸ and F-secure anti-virus scanner²⁹ against the images produced several interesting results. We did not discover a sniffer but, there were three instances of root.exe installed, a Nimda A infection, and a Nimda E infection with an associated root.exe. What fun we will when we really look at the images utilizing Autopsy!

The last manual technique on our forensics list is a look at the registry for any unusual activity. We perused the most recent used files, searched files, typed URLs, last command executed, and last file saved. These searches did not turn up anything we found to be unusual or interesting.

²⁵ An XDCC server that allows others to access files on your computer via an IRC channel. When you start the XDCC server, you start advertising the files that you are offering in every channel selected. When you are offering your packs, somebody may get a pack from you by issuing the command /msg yournickname xdcc send #N where N is the pack number they wish to get. XDCC can transfer up to 3 MB/second.

²⁶ One of the most popular and easily managed free FTP servers that is freely available on the Internet

²⁷ <http://www.ddj.com/documents/s=880/ddj0010f/0010f.htm>

²⁸ <http://www.chkroot.org>

²⁹ <http://www.f-secure.com/products/anti-virus/workstations/>

This leads us to the final steps in our investigation utilizing Autopsy forensic browser and TASK to document the timeline using MAC times, recovering deleted files, and performing string to locate instances and associated inodes for the discovered virus files, root kits, and unauthorized servers. This is also a great means for determining when other utilities started, such as FTP and Netcat.

2.6 Timeline Analysis

During our investigation, we utilized Autopsy version 1.60 brought to us by the folks of @stake. Since that time we have upgraded to Autopsy 1.70. Autopsy is a nice GUI interface into the functionality of TASK, “The @stake Sleuth Kit” created by Brian Carrier of @stake. TASK offers the forensic examiner many specialized tools that facilitate uncovering details about a victim file system. According to @stake, utilizing TASK with the Autopsy browser allows the examiner to:

1. View Allocated and Deleted Files and Directories
2. Access to low-level file system structures
3. Keyword searches including grep regular expressions
4. Timeline of file activity
5. File category sorting and extension checking
6. Hash database lookups including the NIST NSRL
7. Documenting Investigator notes

Some functionality of TASK originates from the Coroners Tool Kit (TCT)³⁰, created by Dan Farmer and Weitse Venema. These include “file” which determines a file’s functionality (is it an executable binary, a graphic binary, ASCII text, etc.), “mactime” which collects information on when a file or directory is created modified or accessed, “ls” displays inode values, istat displays inode information, icat display disk allocation to inodes, “ifind” which inode has allocated a block, “fls” displays file and directory entries, “ffind” determines which file allocated an inode, and “fsstat” displays the details of the above in human readable ASCII. (John Green, SANS Course Book) Fortunately, I did not take screen shots of Autopsy while completing the investigation of this case. Management has closed this case, the files (including the image files created from the victim system) have been archived, and access to the files for further investigation has been denied.

The first step to starting an Autopsy browser session is setting up the fsmorgue file so that TASK, within Autopsy, can mount the image files. By mounting the images instead of the operating system, TASK allows the viewing of the file system contents without writing to any file attributes. Table 2.6.1 illustrates the

³⁰ Freeware: TCT is a collection of programs by Dan Farmer and Wietse Venema for a post-mortem analysis of a UNIX system after break-in.
<http://www.porcupine.org/forensics/tct.html>

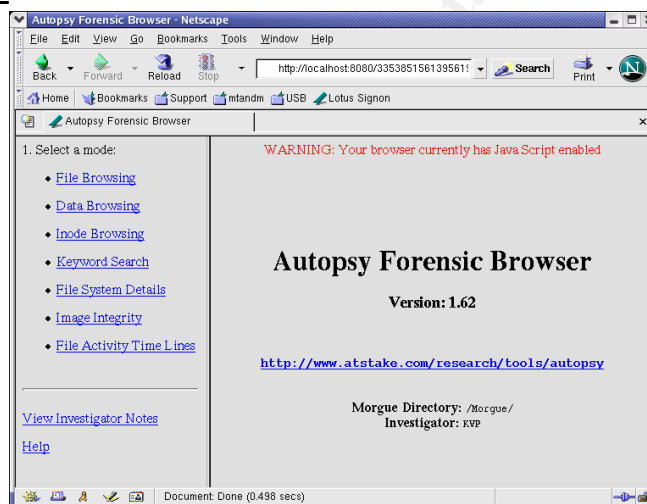
fsmorgue configuration used for this investigation. This file is located in our /images directory to relieve the necessity of declaring a path.

Table 2.6.1

```
# fsmorgue file for Autopsy Forensic Browser
#
memory.img ntfs mem EST5EST
drive0.img ntfs C:\ EST5EST
drive1.img ntfs D:\ EST5EST
boot.img ntfs C:\ EST5EST
```

Next, I started the Autopsy browser using the following syntax and then pasted the returned URL into the Mozilla Internet browser, /src/autopsy-1.60/autopsy 2222 localhost. This opens the Autopsy Forensic Browser as displayed in Screen Shot 2.6.2

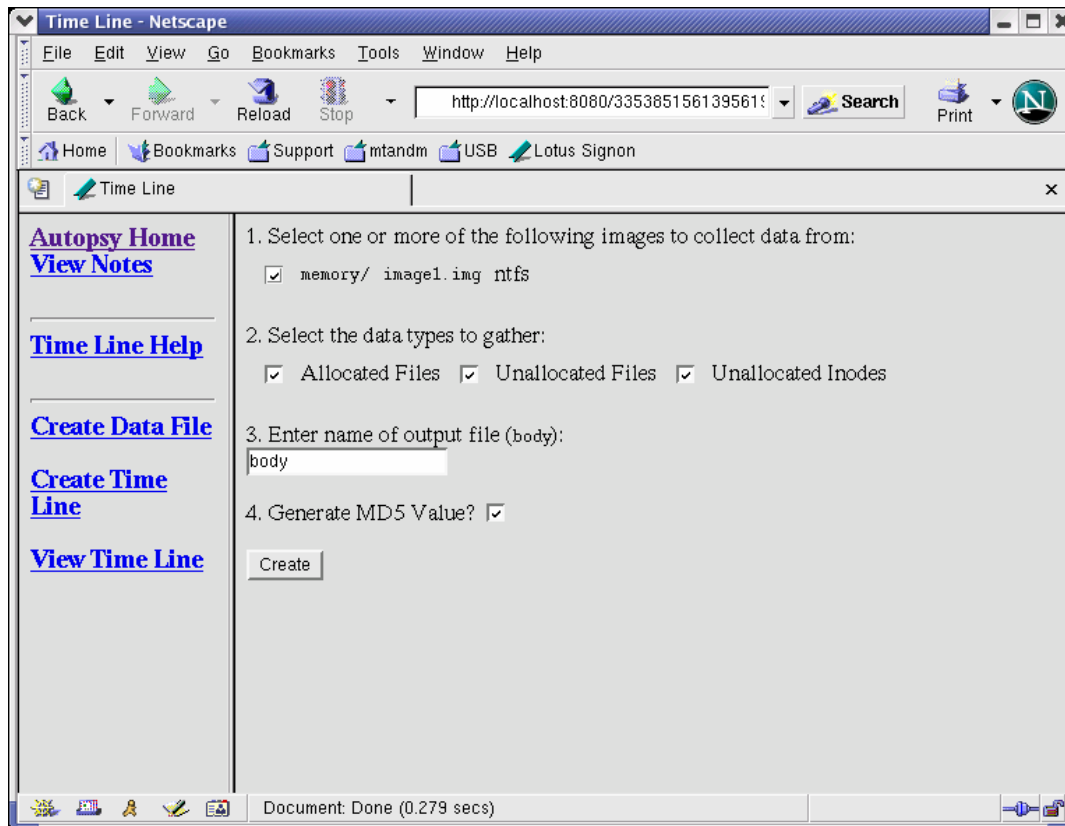
Screen shot 2.6.2



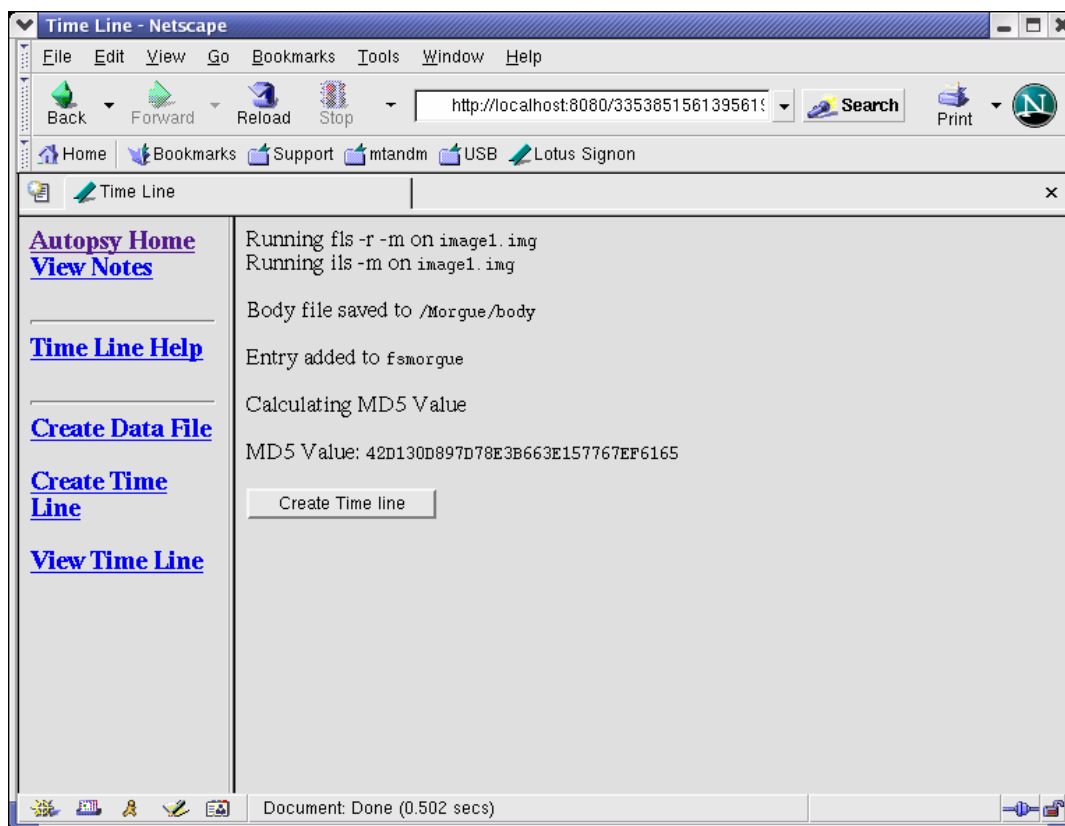
Autopsy does not analyze time signatures on the files. As stated earlier, it is a user interface to TASK. TASK utilizes the command Mactime to actually perform the analysis of file MAC times. MAC stands for modified, accessed, and created. These are the three times of importance to each file and directory within a file system. “These times are recorded within the inode for each file and directory.” (John Green, SANS Course Book) Inodes are important information holds about files and directories. Inodes are numerically ordered; contain MAC time information, file group and ownership information, as well as the file type. TASK, behind Autopsy, accomplishes this by first creating a body file utilizing “Grave-robber.” Grave robber is a tool that gathers system information, such as that contained in the inodes, and compiles them in a static format that does not influence volatility. Once the body file is created, Mactime analyzes the inode’s modified, accessed, and created attributes and creates a listing of each file and

directory with their associated MAC times. (John Green) Creation of the body file is illustrated in Screen shot 2.6.3.

Screen Shot 2.6.3



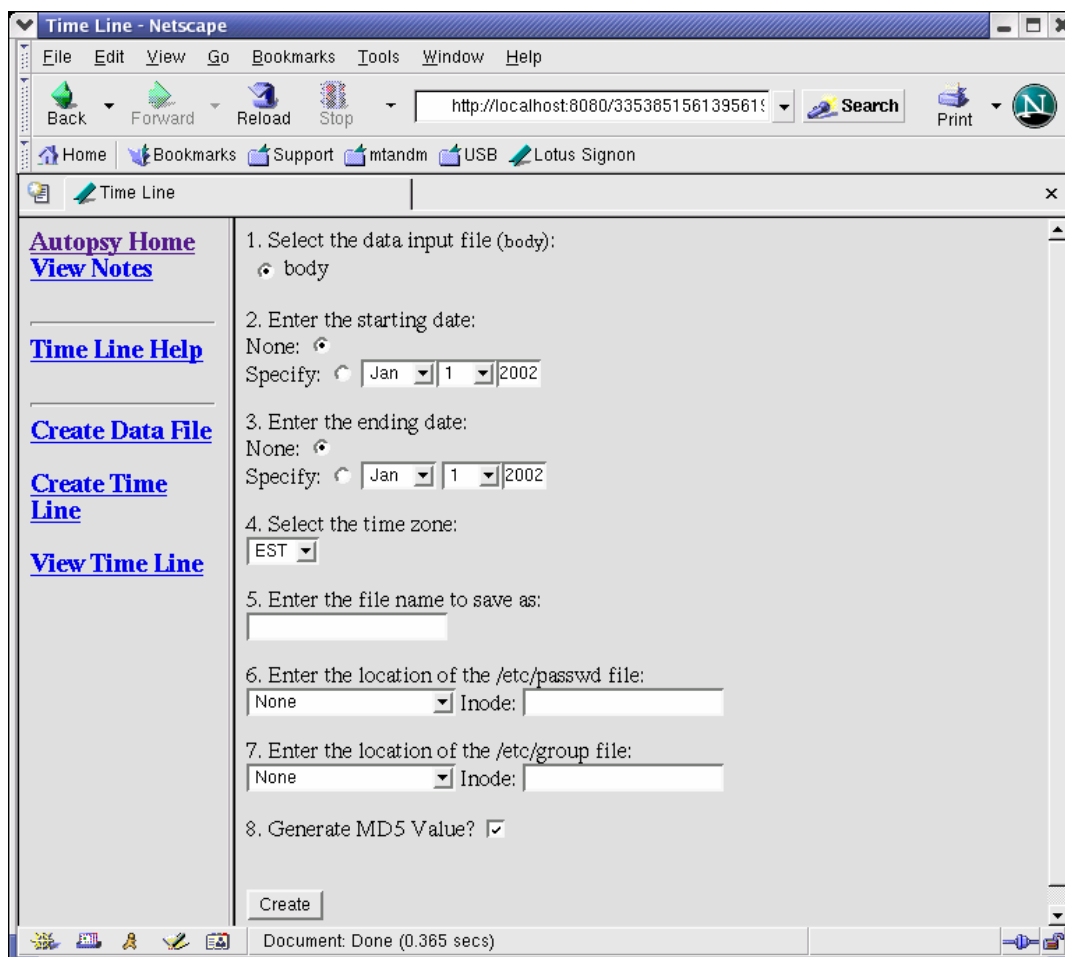
© SANS Institute



Once the body file was created, I created an output file for each image using the “Create Data File” functionality of timeline line creation within Autopsy. This utilizes the above-mentioned Grave-robber program within TASK to create the body file output. For each output file, it is important to include all allocated files, unallocated files, and unallocated inodes. This ensures that all data on within the image is included in each image.

Next, I created timelines using the “Create Time Line” functionality and utilizing each previously created output file. This utilizes the above-mentioned Mactime program within TASK to analyze the file and directory inodes MAC times. I accomplished this by selecting the input file, entering start and finish dates. For the start date I selected “none” as this will begin at the first time recorded on the media and I selected “none” as the end date as this will capture to the last change to the media. Other selections included the appropriate time zone – EST5EST - and a file name for the time line. Autopsy allows for entering a user id and password as well. In our case this was unnecessary and these fields were left as “none”. The final step in timeline creation was to click on the “create” button. This is illustrated in Screen Shot 2.6.4.

Screen Shot 2.6.4



Now that we have created the timelines, it was time to view the time lines and begin documenting significant historical events for the victim server. Utilizing the timeline, we located the root kit executables, IRC executables. Instances of FTP and Netcat use, bootlegged files, as well as Nimda instances. The following section is a breakdown of the time line documented for this server:

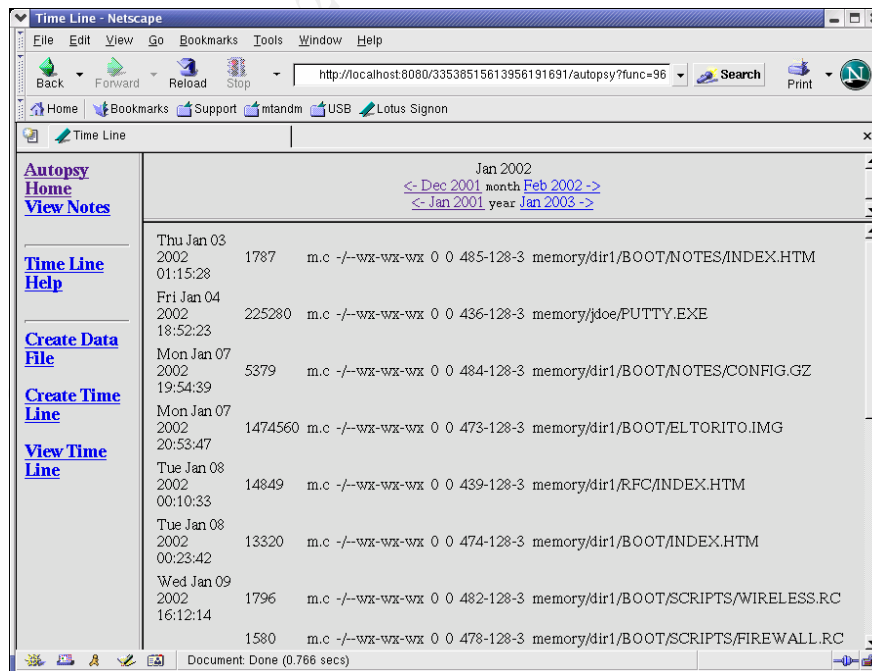
01-04-01	Windows NT Installed
12-02-01	Initial NimdaE infection
12-12-01	02:48 NimdaA infection Root.exe
01-18-01	IIS installed
06-10-02	Server put on-line installed
06-10-02	12:56 root.exe exploited
06-13-02	07:19 serv-u32 FTP uploaded and installed Backdoor/Trojan FTP server installer Possibly placed there by HelfirE and/or Neo of kewl.org User: edmn0r

Pass: ei.nnaJ1Qi7Y

06-28-2002	23:12	Serv-U started
06-29-2002	01:19	Men in Black II uploaded begins
06-29-02	7:05	Men in Black II upload complete
08-06-02		Another NimdaA infection Traces of company script mixed with traces of Nimda in !ygwn1.dll
09-05-02	18:45	iroffer started Connected to irc.fr.kewl.org #Megami TEAM
09-05-02	07:52	FTP data upload begins Intermittent through 09-11-02 Total of 921 MB
09-05-02	19:08	XDCC data transfers begins
09-06-02		nc.exe uploaded to C:\
09-10-02		nc2.exe uploaded to C:\
09-18-02		nc.exe uploaded to C:\WINNT
09-19-02	11:23	iroffer stopped Files transferred by IRC users during the two weeks that iroffer was on-line.
11-01-02		TK.R007 installed again

An example of an Autopsy time line is illustrated in Screen shot 2.6.5.

Screen shot 2.6.5



Let us examine the timeline in detail. At one point in time, this server lacked appropriate virus protection and in December of 2001 acquired a “Nimda A” infection. This infection, among other things, created a root.exe back door into the server. An unnamed employee subsequently installed “Product A” onto the server and implemented this it onto the Internet during June 2002, bypassing established change control mechanisms that would normally prevented an out of date operating system with no security releases from reaching production.

An unauthorized user, possibly HelfirE and/or Neo of kewl.org, took advantage of the Nimda root.exe backdoor and mounted the server later that very day. Kewl.org is a French IRC network. The speed at which the vulnerable server was located and hacked shows that the intruders were actively scanning the Internet looking for servers for their file transfers. The FTP server scripts had been customized for this specific server and contained the hacker ID information; and login information; User: edmn0r and Pass: ei.nnaJ1QiT7Y. Three days later, on 13 June, the intruders installed a popular hacker toolkit that includes Tkb0t, another root backdoor, and an FTP server Serv-u. For some unknown reason, there was a delay in use of the installed tools. The hackers were adept enough to utilize and customize known freeware tools as well as customize them for a particular system.

On June 28 the intruder started the FTP server and file uploads onto the server began. File uploads commenced with the “Men in Black 2” compressed .rar files. The transfers began at approximately 1:19 the next morning and took nearly six hour to complete. The total file size transferred exceeded One gigabyte of data! Evidence within the directory structure revealed something about the hackers skill set.

At this time, ISIRT held a round table discussion to determine whether to involve law enforcement. The management decision was to not engage the authorities; it had been six months since the pirating activity had ceased. The pirates had come and gone.

Later, in 2002, another file upload began. A French Internet user group named Megami TEAM, also originating from kewl.org, uploaded 921 megabytes of soft pornography anime files. This occurred on September 5 through September 11. Later on the fifth, the XDCC IRC file transfer server started. The unauthorized users configured the server to advertise the availability of the bootlegged software in several IRC channels. Later that day file transfers began from our exploited server to Internet users who had joined the IRC channels. The file transfers continued through September 19, 2002.

During this period in September, the intruders transferred three other files onto the server. These file included different versions of Netcat³¹, named nc.exe in one location, nc.exe in another location, and nc2.exe in a third location. Netcat is a very powerful networking tool utilized by many network professionals, security professionals, as well as most hackers.

2.7: Deleted File Recovery

Recovering deleted files from a system is often helpful during a forensic examination. I found this to be true in this case. Once again, I used Autopsy to assist in the analysis at hand for each of the images. TASK accomplishes this technique utilizing the “dls” command, which is similar to the “unrm” UNIX command utilized by TCT. (John Green) This command is the undelete command for UNIX. To accomplish this I went into the main Autopsy menu and selected to browse the files. One option within the “File Browsing” menu is “All Deleted Files.” Selecting this item allowed me to peruse a listing of the deleted files and to select individual files by clicking on the link the file path and name create.

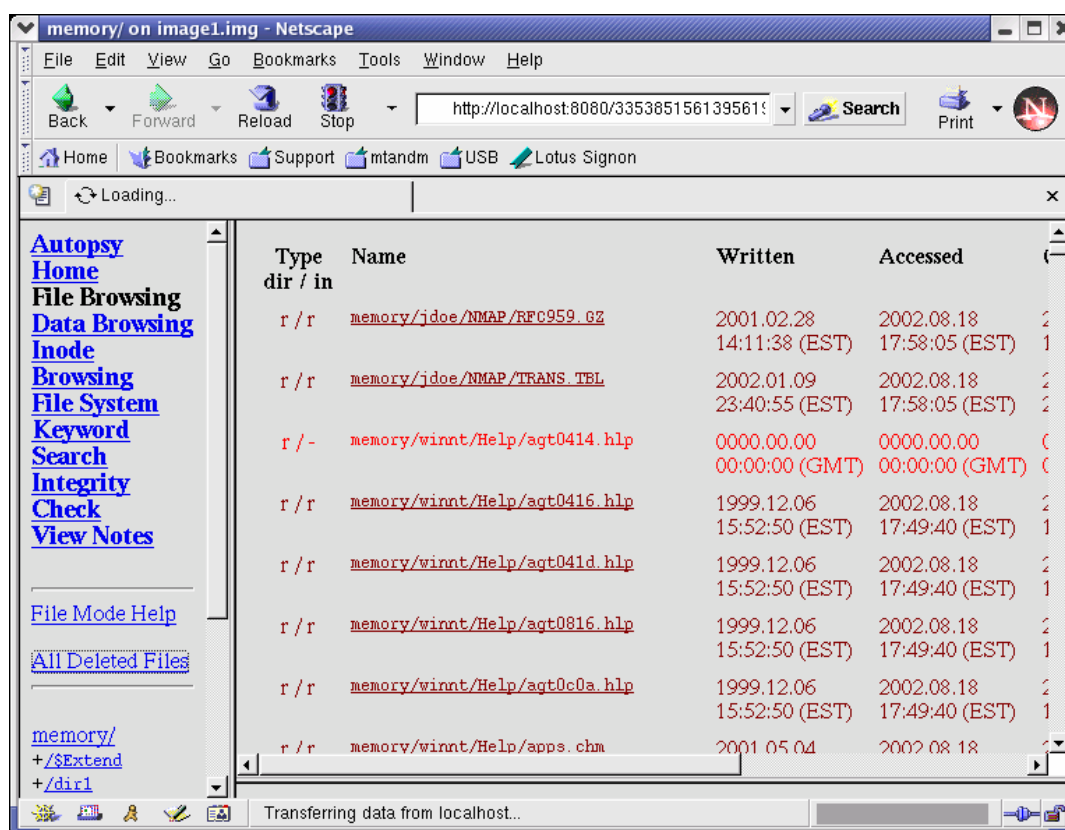
Most of the deleted files were temporary files from Windows and IIS installation. During further analysis, I discovered what appeared to be eleven .avi files. The names of the files were very generic. This was interesting, indeed! I decided to export one of the files. I transferred the exported file to a workstation in our Information Security lab. We utilize this workstation for examining potentially hazardous files. Executing the file launched an Anime video clip. This file was of high, broadcast, quality. Executing the other recovered .avi files, utilizing the same methodology, resulted in further installments of the same anime show. After searching on-line, we discovered that the program was indeed broadcast quality. The files were .avi files of a Japanese prime time anime serial!

No other files of interest were discovered during the review of recovered, previously deleted, files and directories.

An illustration of Autopsy used for displaying deleted follows on the next page within Screen Shot 2.7.1.

³¹ Information on Netcat is available from SANS:
<http://www.sans.org/rr/audit/netcat.php>

Screen shot 2.7.1



2.8: String Searches

During our string searches of the images we focused on words and acronyms that would help us identify inodes for time line focus on already identified exploits, as well as potentially identify additional forensic findings. We completed string searches on both of the victim server's drive images and the image of the physical memory. The string searches were performed utilizing Autopsy. The commands called by Autopsy to complete these techniques included the UNIX command "grep" and "strings."

From the main Autopsy interface, I selected the "Keyword Search" option. The step from there was to create a file that contains ASCII text strings and to create an unallocated data file. The strings file speeds the search for keywords by eliminating non-text related data. The unallocated file ensures that words of interest that have been deleted or are located within unallocated disk space are also included within the search findings. Once these files were created, I entered the keywords into the "Enter String:" field within the Autopsy window. Autopsy then parsed through the strings and unallocated files to locate and return all instances of the chosen search terms.

We first focused on identifying when and how some of the already identified exploits occurred. To accomplish this we searched for iroffer, servu, tk, root, men in black, avi, and FTP and NC. The first search terms in our list allowed us to identify the inodes associated with each of the known exploits that were on the victim server. The FTP searches defined when and where the FTP utilities executed to upload root kits, IRC software, and bootlegged video files.

Additional searches for root, several expletives, r007, and other hackerese terms resulted in discovering the same exploits previously discovered in other manners. We discovered no additional exploits using these terms.

2.9: Conclusion

We may make several suppositions based on our forensic evidence. At least two unauthorized users gained access into the server. The December of 2001, "Nimda A" infection and resulting root.exe back door into the server opened the server to free access due to the June bypass of established change control mechanisms. The intruders first installed a popular hacker toolkit that includes Tkb0t, another root backdoor, and an FTP server Serv-u. Later that month the intruder started the FTP server and file uploads commenced with the one Gigabit of "Men in Black 2" compressed .rar files.

A French Internet user group named Megami TEAM, uploaded 921 megabytes of soft pornography anime files. The XDCC IRC file transfer server started and later that day file transfers began from Internet users who had joined the IRC channels in which the XDCC advertised.

Other hacker tools, including Netcat, were uploaded onto the server more than one time. This server was wide open. It appeared that the hackers were using the server for more than just file transfers. The server seemed to be utilized as a sandbox to practice hacking techniques.

Overall, we identified several deficiencies within the controls utilized to prevent implementation of a vulnerable Internet server within our network environment. We developed remedial activities with management to correct the weaknesses and have received commitments from individual managers as well as timelines for completion of the agreed to action plans.

Part Three

Incident Handling Protocols for an Unknown ISP **Contacted by the Authorities**

Important changes to law affecting ISPs occur regularly post September 11, 2001. The United States Congress reactively seeks to prevent future negative events against the United States and at the same time increase the strength and power of government law enforcement agencies. Congress attempts to keep in mind the sanctity of individual rights while granting the wishes of law enforcement and at the same time guaranteeing the authority of the federal government to obtain “remedies and sanctions for nonconstitutional violations of chapter 121 of Title 18 of the US criminal code- Stored Wire And Electronic Communications And Transactional Records Access.”³² Disclosure requirements of an ISP is fully described within

3.1: Initial Contact

Initial contact with law enforcement may often originate through a telephone call. The first priority of the ISP should be to determine the authenticity of the caller. This is difficult if not impossible during the online conversation. Therefore, the ISP should only provide publicly available information. This information could include the ISP’s name, address, and URL. The ISP should use the initial conversation to gather information. Firstly, the ISP should ascertain the authenticity of the government agent. This could be accomplished by getting the agents contact information, verifying the information, and finally through a call back.

Next, the ISP should determine whether or not to voluntarily submit the requested information to the law enforcement agent. The ISP could gather information pertinent to the case that will allow them to determine whether or not the activity occurred within the ISP’s systems and if the activity was potentially conducted by a subscriber of the ISP’s services. This information could include the time of the incident, the origin IP address of the incident, and the Internet protocols or other details of the Internet traffic generated during the activity. The ISP could then investigate their log files to determine what happened and what user conducted the activity. It is extremely important to verify who conducted the activity. In particular, the ISP is strictly prohibited from volunteering any information pertaining to electronic communications or data stored electronically within the ISP’s systems if that information pertains to a valid customer of that ISP. The 18 U.S.C. § 2702³³ illustrates this by stating that:

“A person or entity providing an electronic communication service to the public shall not knowingly divulge to any person or entity the contents of a communication while in electronic storage by that service...or contents of any communication which is carried or maintained on that service...a

³² [18 U.S.C § 2708. – Exclusivity of remedies](#)

³³ [18 U.S.C. §2702. - Voluntary disclosure of customer communications or records](#)

provider of remote computing service or electronic communication service to the public shall not knowingly divulge a record or other information pertaining to a subscriber to or customer of such service... to any governmental entity.”

As always, there are exceptions to this rule. Foremost, if the ISP should determine whether the user committing the activity is included within the listing of authorized users and that the activity did occur within the ISP’s system in an unauthorized manner. If the user conducting the named activity is determined to be a valid user of the ISP’s services, the exceptions include divulging information to the addressee of information, upon the subscriber or customer’s authorization, or to a downstream forwarder of the information. The ISP could also volunteer information if the activity was in the commission of a crime or the ISP believes that there is an immediate risk of danger or a threat to someone’s life. The 18 U.S.C. § 2702 states that the service provider may release information to law enforcement:

“If the contents were inadvertently obtained by the service provider; and appear to pertain to the commission of a crime; or if the provider reasonably believes that an emergency involving immediate danger of death or serious physical injury to any person requires disclosure of the information without delay.”

Since the government agent cannot provide these exception requirements to the ISP during an initial phone conversation, the ISP should not release any information at this time. Further investigation is necessary. If the ISP does determine that exception requirements are fulfilled the ISP should divulge information at that time only.

3.2: Information Preservation

Often law enforcement will ask the ISP maintain or preserve copies of evidence, such as log files, prior to providing the ISP with a warrant or court order. The law does not explicitly state how the request is made, it only requires “upon the request of a governmental entity.” Therefore, one may assume that either a verbal or a written request to preserve evidence could suffice. Whenever an ISP receives such a request from a bona-fide agent of the government the ISP is required to “take all necessary steps to preserve records and other evidence in its possession pending the issuance of such a court order or other process (warrant).”³⁴ According to the statute the requirements to preserve evidence includes a retention time of 90 days from the date of the request and may be extended no more than an additional 90 days upon the request of the agency.

³⁴ [18 U.S.C. § 2703 - Required disclosure of customer communications or records](#)

3.3: Legal Authority

The legal authority for compulsory submission of evidence, such as log files, to an agent of the government requires at the minimum a warrant or a court order. There are strict rules that pertain to these court orders or processes contained within 18 U.S.C. §2703, and they are:

“A court order for disclosure under subsection (b) or (c) may be issued by any court that is a court of competent jurisdiction and shall issue only if the governmental entity offers specific and articulable facts showing that there are reasonable grounds to believe that the contents of a wire or electronic communication, or the records or other information sought, are relevant and material to an ongoing criminal investigation. In the case of a State governmental authority, such a court order shall not issue if prohibited by the law of such State. A court issuing an order pursuant to this section, on a motion made promptly by the service provider, may quash or modify such order, if the information or records requested are unusually voluminous in nature or compliance with such order otherwise would cause an undue burden on such provider.”

In the case of the state of Georgia, the rules are much stricter. According to Cassandra Schansman, Assistant Attorney General for the State of Georgia, “In many states the vehicle is as simple as obtaining a subpoena. Unfortunately, Georgia does not possess an investigative subpoena power. Thus for Georgia criminal investigations, unless the suspected crime involves a statutorily-defined subpoena, either a court order or a search warrant is usually required to obtain even the most basic computer information.”

Much of the pains that state governments have been eased with the passing of the USA Patriot Act. As an example, 3rd Party/ISPs can honor court orders signed by a state court judge in another state for subscriber information and other related subscriber data related to digital evidence. This is commonly referred to as a ‘2703 “d” order’. (18 U.S.C. §2703)

3.4: Permitted Investigative Activity

If the agent of the government has not made a mandatory demand for the evidence with a warrant or court order, the ISP is free to complete their own forensic investigation. This investigation may be used to determine whether the suspicious activity originated from the ISP servers, what user completed the activity, if the activity originated upstream, and what down stream servers may have been impacted by the activity. The ISP’s investigation is forbidden to do anything that could tamper with or alter the potential evidence. This is affirmed within 18 U.S.C. §2703. “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.”

If the government agency does provide a warrant or court order, the only activity legally required of the ISP would be to back up the information. The ISP could not perform any other activity at this time. The government agency may require the ISP to perform such a backup. If so, the ISP is required to perform the back up and to “release it to the government agency no later than 14 after the request has been made.”³⁵ The ISP may not quash a request from law enforcement for a back up. The back up must be completed and could be destroyed only after the time that the court order itself is quashed.

A difficulty with warrants and court orders occurs due to the protection of the fourth amendment to the constitution. The prevention of illegal search and seizure has been upheld repeatedly in state and federal court. For instance, “Under US v. Bach, Crim. No. 01-221 (D. Minn. Dec. 14, 2001), the federal district court in Minnesota found that it was a violation of the 4th Amendment and other applicable federal law for the officer to fax the warrant to the ISP for execution without being present. Thus under Bach, the third Party/ISP cannot obtain the information in advance and have it ready when the officer arrives. If an officer is not present, the information is tainted and all other evidence collected as a result of the leads developed from the tainted data is inadmissible.”³⁶ (Edwards and Schansman)

3.4: Government System Hacked

A different scenario could arise that would change the responsibility of the ISP during the reaction to the agents initial request for information. If the ISP determines, through any investigative means, that a government server was indeed hacked either directly through their server(s) or from a downstream source using their server(s) as a proxy, the ISP is required to provide any and all information requested by the government agency. (18 U.S.C. § 2702) In this alternate case, the ISP did determine through log file analysis that an unauthorized user gained access to the system, created an account on the system, and then hacked a government system.

³⁵ [18 U.S.C. 2704. - Backup preservation](#)

³⁶ However, it should be noted that this case is currently under appeal and is being addressed by Congress. (Senate Bill 11010 and HR 2215 - October 2002). It is anticipated that this requirement for an officer to be present will be done away with because of both the federal law and the impending U.S. 8th Circuit Court opinion.

According to 18 U.S.C. § 2702 The ISP would be compelled to voluntarily provide requested information for two reasons. Firstly, the information was “inadvertently obtained” by the ISP due to the unauthorized account creating the activity. The events did not occur according to normal and customary ISP customer activity. Finally, the activity “pertained to the commission of a crime” due to the hacking of a government system. This is protected at a minimum under 18 U.S.C § 1362 which states that:

“Whoever willfully or maliciously injures or destroys any of the works, property, or material of any radio, telegraph, telephone or cable, line, station, or system, or other means of communication, operated or controlled by the United States, or used or intended to be used for military or civil defense functions of the United States, whether constructed or in process of construction, or willfully or maliciously interferes in any way with the working or use of any such line, or system, or willfully or maliciously obstructs, hinders, or delays the transmission of any communication over any such line, or system, or attempts or conspires to do such an act, shall be fined under this title or imprisoned not more than ten years, or both.”

In conclusion, an ISP’s responsibility may vary from state to state as far as whether or not to expect/respect a warrant, court order, or polite request from law enforcement. But, an ISP must always keep in mind ultimate federal authority occurring from changes due to the US Patriot act under Title 18 that guarantee the federal government to obtain “remedies and sanctions for nonconstitutional violations of chapter 121 of Title 18 of the US criminal code- Stored Wire And Electronic Communications And Transactional Records Access.” (18 U.S.C. § 2708)

© SANS Institute

References

- [1] "Basic Steps in Forensic Analysis of Unix Systems". URL: <http://staff.Washington.edu/dittrich/misc/forensics> (March 2003)
- [2] Farmer, Dan. "FAQ Webpage". URL: <http://www.fish.com/tct/FAQ.html> (March 2003)
- [3] Farmer, Dan. "Help! How do I recover that important file?". URL: <http://www.fish.com/tct/help-recovering-file> (March 2003)
- [4] Farmer and Venema. "Computer Forensic Class Handouts". URL: <http://www.porcupine.org/forensics/handouts.html> (March 2003)
- [5] Farmer, Dan. "What are MAC Times?". Dr. Dobbs Journal. October 2002. URL: <http://www.ddj.com/documents/s=880/ddj0010f/0010f.htm> (March 2003)
- [6] Green, John. "Basic Forensic Principles Illustrated with UNIX". SANS Course Book. System Forensics, Investigation and Response. (November 2002)
- [7] Lee, Rob. "Windows Forensics and Incident Response". SANS Course Book. System Forensics, Investigation and Response. (November 2002)
- [8] Venema, Wietse. "File Recovery Techniques". Dr Dobbs Journal. December 2000. URL: <http://www.ddj.com/articles/2000/0012/0012h/0012h.htm> (March 2003)
- [9] @stake, Inc. "@stake Research Tools". URL: <http://www.atstake.com/research/tools> (March 2003)
- [10] @stake, Inc. "Digital Forensic Analysis". Course Book, January 2003
- [11] "The Coroners Toolkit". URL: <http://www.porcupine.org/forensics/tct.html> (March 2003)
- [12] Legal Law Institute. "United States Code". Cornell University. URL: <http://www4.law.cornell.edu/uscode> (March 2003)
- [13] Library of Congress. "THOMAS – U.S. Congress on the Internet". URL: <http://thomas.loc.gov> (March 2003)
- [14] Edwards, Steve and Schansman C. "Obtaining Electronic Evidence from Third Parties". State of Georgia Archives

- [15] Okena Security Software, "Okena StormWatch". URL: http://www.okena.com/areas/products/products_stormwatch.html (March 2003)
- [16] Prentner, Karl. "Forensic Reporting Style".
- [17] Prentner, Karl. "Procedures for Computer Forensic Data Collection".
- [18] Prentner, Karl. "Chain of Custody".
- [19] Biatchux. "Fire". URL: <http://fire.dmzs.com/> (March 2003)
- [20] IETF.org. "RFC 1321: MD5 Hash Sums", URL: <http://www.ietf.org/rfc/rfc1321.txt> (March 2003)
- [21] Surge Force. "Project – Templer Knight Bot". URL: <http://sourceforge.net/projects/tkbot/> (March 2003)
- [22] Iroffer.org. "Iroffer – IRC Fileserver". URL: <http://iroffer.org> (March 2003)
- [23] RhinoSoft. "Serv-u, the Industries Most Popular FTP Server". URL: <http://www.serv-u.org> (March 2003)
- [24] F-secure. "F-secure Virus Protection". URL: <http://www.f-secure.com/products/anti-virus/workstations/> (March 2003)
- [25] Chkroot.org. "Chkroot Root Kit and Trojan Detector". URL: <http://www.chkroot.org> (March 2003)
- [26] Armstrong, Tom. "The TCP/IP Swiss Army Knife". URL: <http://www.sans.org/rr/audit/netcat.php> (March 2003)
- [27] daemon9 and alhambra. "Project Loki: ICMP Tunnelling". Phrack Magazine, Volume Seven, Issue 49, File 6 of 16, August 1996. URL: <http://phrack.infonexus.com/search.phtml?view&article=p49-6> (March 2003)
- [28] King, Taran. "Introduction to Phrack Inc". Phrack Magazine, Volume One, Issue One, File 1 of 8, November 17, 1985. URL: <http://phrack.infonexus.com/search.phtml?view&article=p1-1> (March 2003)
- [29] Walters, Stan B. "Interviewing for Credibility: Accurate Identification of Deception Behaviors". URL: http://www.kinesic.com/interviewing_for_credibility.htm (March 2003)

- [30] Ethereal.com. "The Ethereal Network Analyzer." URL:
<http://www.ethereal.com/>
- [31] Wincap.polito.it. "Windows Packet Capture Library." URL:
<http://wincap.polito.it/>

© SANS Institute 2003, Author retains full rights.