Global Information Assurance Certification Paper

GIAC Certified Forensic Analyst Practical Assignment
Version 1.0
Roland E. Miller, III
September 16, 2002

Part 1 – Option 1: Forensic Analysis of a System

Case Facts Synopsis

This case involves the investigation of an "unknown" computer system to determine what information can be obtained about the user of the system. This particular system was also chosen to help investigate the application of Linux/UNIX and Microsoft Windows forensic techniques and technologies using the Mac OS X operating system both as the target system and as the forensic analysis system. The target system was recently retired and therefore presented an excellent opportunity to employ the forensic analysis techniques required for this assignment.

The target system is a functionally obsolete Apple Macintosh acquired from the organization's senior UNIX systems administrator (SA). The system was originally transferred to the SA in 2000 so that the capabilities of Apple's new UNIX based operating system Mac OS X (Apple 1, p.1) could be tested out. The SA installed the Public Beta version of Mac OS X, which had a built-in expiration period. The SA eventually setup the system to be a fail-over system for the organization's DNS infrastructure utilizing the Darwin (Apple 3, p.1) core functionality of Mac OS X after the Public Beta GUI portion expired. Early in 2002, the organization's primary DNS system had a hardware failure and the target system was setup to service the DNS requests until the original DNS hardware could be replaced. The target system was taken off-line after the repaired DNS was up. It has been sitting in the SA's office unpowered since then.

Seizure Inventory

The following hardware represents the target system that was seized on July 23, 2002 at 7:25 PM MDT from the organization's UNIX support office, room 321. The information listed includes the case Tag Number, a description of the item, the item's serial number (if present) and the organization's inventory number (if present.) The system was not live when it was seized and therefore no system state information was obtained prior to drive imaging.

Tag Number	Description	Serial Number	Org. Inv. #
20020723A1	Power Macintosh 9500/132. Manufactured: 7/13/95 B-2 11:22PM.	XB5280785UZ	226963
20020723A2	Apple Design Keyboard.	NN451E4A33G	N/A

	Model # M2980.		
20020723A3	Apple Desktop Bus Mouse II. Family # M2706.	LC525GQWT18	N/A
20020723A4	Apple MultipleScan 1705 Display. Model # M4436. Manufactured: June 1996.	CY6254BY5X5	N/A

The case on Tag #20020723A1 had a rust colored water ring on the top. The case was opened by removing six fasteners. Besides the typical amount of dust in the case there appeared to be small green needles in the case, possibly from a small fern. The system had the following peripherals installed; it was not possible to see how much and what type of RAM was installed at this point.

Number	Description
1	SCSI CD-ROM
1	3-1/4 Floppy Drive
2	SCSI Hard Disks
1	PCI Video Card
1	PCI 10/100 NIC

The hard disks were disconnected and removed. The disks were tagged, information was removed from the case labels and the jumper settings were examined. Information from each disk is shown below.

Tag #	Description	Part Number	Serial Number	Size	Jumper
20020723A1a	Seagate Barracuda, ST32171N. Upper HD Bay.	9C6002-010 Lot A-01-9722-4	JE206628	2.16 GB	ID 1, TP
20020723A1b	Seagate Hawk, ST32430N. Lower HD Bay. Apple OEM.	9B1001-048 Lot S-01-9549-2	TX241770	2.14 GB	ID 0, TE, TP1, TP2, PE

Media Imaging

The hard disk specifications were pulled up from Seagate's web site to determine if there were any jumper settings available that would put the disks into read-only mode. Based on the specifications, both disks had this option that was called write protect. The associated jumper (WP) on each disk was set in order to prevent alteration of the original data. In addition, Tag #20020723A1b had the TE jumper set (termination enabled) as it would be the only device on the SCSI chain during imaging.

The imaging system was an Apple Power Macintosh G4, 500 MHz dual-processor, 768 MB RAM and a 20 GB hard disk split into three equal HFS+ partitions running Mac OS X 10.1.5 (Build 5S66) with an Apple OEM SCSI PCI card installed. All imaging was conducted in single-user mode to avoid any interaction by the GUI. To get into single-

user mode, the system was booted with the 'cmd-s' key combination. Once booted, the file system was checked and then changed to read-write mode by utilizing the following commands:

```
localhost:# fsck -y /
localhost:# mount –uw /
```

These commands were entered each time the system was subsequently booted into single-user mode.

Tag # 20020723A1b was connected to the SCSI card and the internal power of the imaging system. The system was booted into single-user mode. Once in single-user mode, the VPC partition of the internal OEM IDE hard disk was mounted and sterilized using the following commands:

```
localhost# mount_hfs /dev/disk0s11 /Volumes/VPC
localhost# dd if=/dev/zero of=/Volumes/VPC
```

Upon startup, Tag # 20020723A1a was assigned to device disk1s7 by the OS. The MD5 hash was calculated as shown below:

```
localhost# openssl md5 /dev/disk1s7

MD5(/dev/disk1s7)= 221467af347d6d50c762fff6b7056954
```

A *dd* copy of the disk was then made as shown below:

```
localhost# dd if=/dev/disk1s7 of=/Volumes/VPC/disk2.img

4175293+0 records in
4175293+0 records out
2137750016 bytes transferred in 821 secs (2603836 bytes/sec)
```

The MD5 hash of the image was calculated in order to verify that an exact copy of the original data was made as shown below:

```
localhost# openssl md5 /Volumes/VPC/disk2.img

MD5(/Volumes/VPC/disk2.img)= 221467af347d6d50c762fff6b7056954
```

This verifies that the image is an exact copy of the original disk. At this point, the system was halted and Tag # 20020723A1b was removed and placed in a locked cabinet.

Tag # 20020723A1a was then connected to the SCSI card and the internal power of the imaging system. The system was booted into single-user mode and the VPC partition was mounted. The disk was assigned to device disk1s5 by the OS after booting. The MD5 hash was calculated as shown below:

```
localhost# openssl md5 /dev/disk1s5
```

```
MD5(/dev/disk1s5)= 891c838fb907b38de28a51b9f6c9c34d
```

A *dd* copy of the disk was then made as shown below:

```
localhost# dd if=/dev/disk1s5 of=/Volumes/VPC/disk1.img

4221926+0 records in
4221926+0 records out
2161626112 bytes transferred in 324 secs (6671685 bytes/sec)
```

The MD5 hash of the image was calculated in order to verify that an exact copy of the original data was made as shown below:

```
localhost# openssl md5 /Volumes/VPC/disk1.img

MD5(/Volumes/VPC/disk1.img)= 891c838fb907b38de28a51b9f6c9c34d
```

This verifies that the image is an exact copy of the original disk. At this point, the system was halted and Tag # 20020723A1a was removed and placed in a locked cabinet.

To summarize, the following items were successfully imaged with the corresponding MD5 hashes verified as being identical.

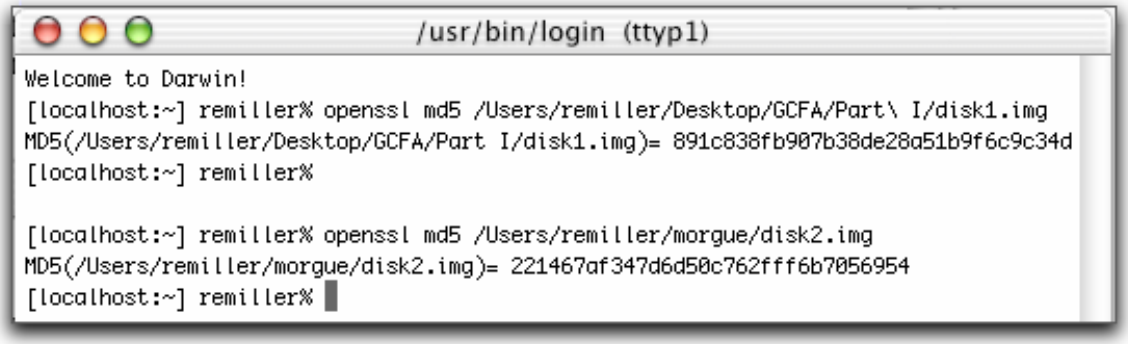
Tag #	Image Name	MD5
20020723A1a	disk1.img	891c838fb907b38de28a51b9f6c9c34d
20020723A1b	disk2.img	221467af347d6d50c762fff6b7056954

Media Analysis

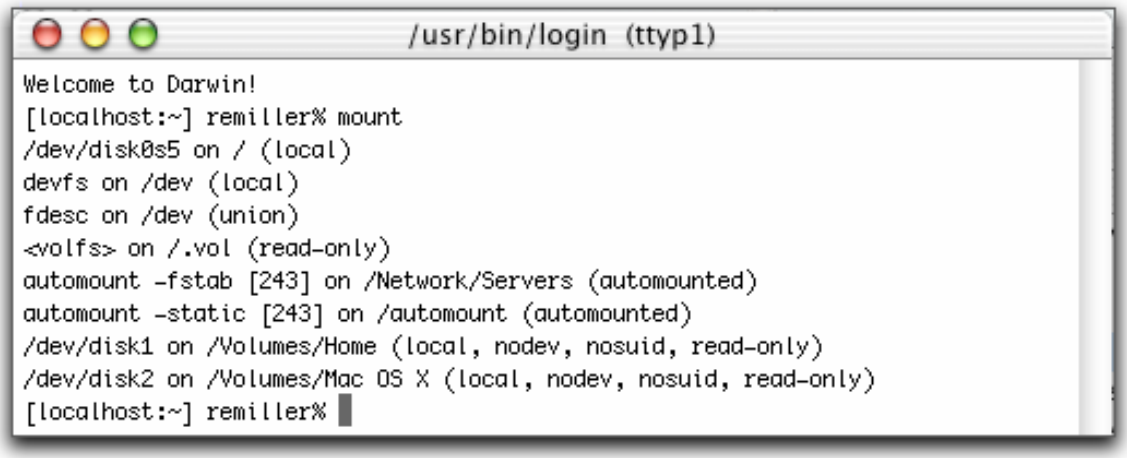
The forensic analysis system was an Apple PowerBook G4, 667 MHz, 512 MB RAM, 30 GB HD running Mac OS X 10.1.5 (Build 5S66). Besides the programs and tools present in the base install of the OS, the December 2001 release of the Mac OS X Developer Tools was installed. For the analysis, the following software was utilized. Specifics regarding the usage of each package are stated in the respective analysis.

Software	Version
BSD Subsystem of Mac OS X (Apple 2, p.1)	10.1.5
Mac-robber (@stake, p.1)	1.00
TASK (@stake, p.1)	1.00
Autopsy (@stake, p.1)	1.50
Norton Utilities for Macintosh UnErase (Symantec, p.1)	7.0
ProSoft Data Rescue X (demo) (Prosoft, p.1)	10.0
McAfee Virex (Network Associates, p.1)	7.1

The disk images were transferred via 100 Mbps Ethernet to the analysis system. The MD5 hash was calculated for each image as shown below to verify that the image was not changed in the transfer.



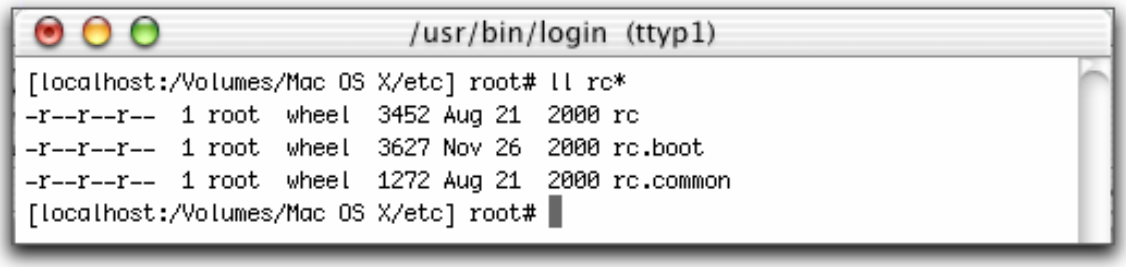
While in the GUI, the permissions were changed to read-only for both disk images and mounted as such using Disk Utility. This built-in application is simply a wrapper for the various command line utilities that perform the actual mounting. It should be noted that after consulting the man pages for these commands, Mac OS X does not appear to have the capability to prevent access times from being changed (i.e., there is no *noatime* equivalent). The images and their associated mount points and properties are shown below.



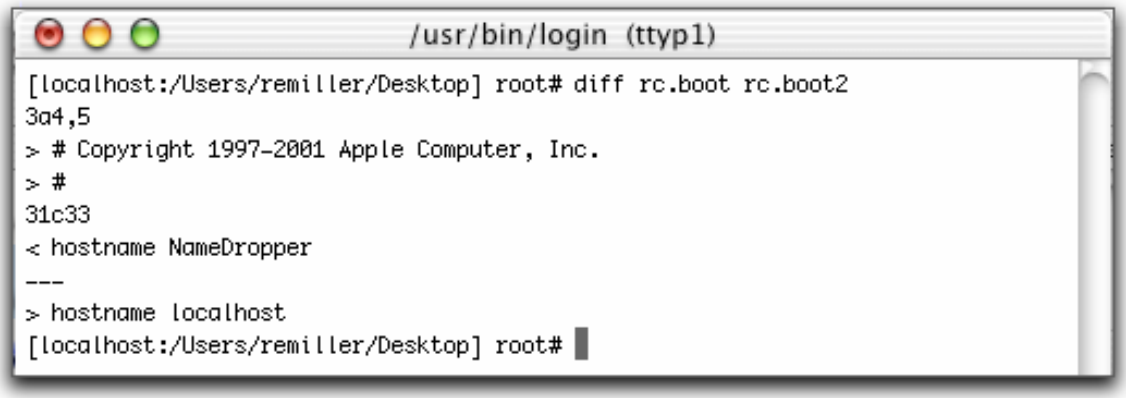
Basic information about the target system was determined by examining the associated system files as shown below.

Information	Value	Origin
OS Version	Mac OS X Public Beta (1H39)	/System/Library/CoreServices/software_version
OS Install Date	Nov 17 2000	/Library/Receipts/Essentials.pkg & BSD.pkg
TimeZone	Mountain	/etc/localtime
Last Boot Date	Apr 29 2002	/var/log/system.log

System startup is defined by the /etc/rc* files, the /etc/hostconfig file and also by what is present in the /System/Library/StartupItems folder. A directory listing of the /etc/rc* files is shown below.



According to the directory listing above, the date for the rc.boot file indicates that it has been modified after the apparent system installation date. Running *diff* on rc.boot with an unaltered version of the file obtained gives the output shown below. Clearly, the only substantive change in the rc.boot file is the addition of the specific hostname of the system.



The contents of the hostconfig file are shown below. The actual router entry has been replaced with xxx's.

```
##
# /etc/hostconfig
##
# This file in maintained by the system control panels
##

# Network configuration
HOSTNAME=NameDropper
ROUTER=xxx.xxx.xxx.x

# Services
AFPSERVER=-NO-
APPLETALK=-NO-
AUTHSERVER=-NO-
AUTOCONFIG=-YES-
```

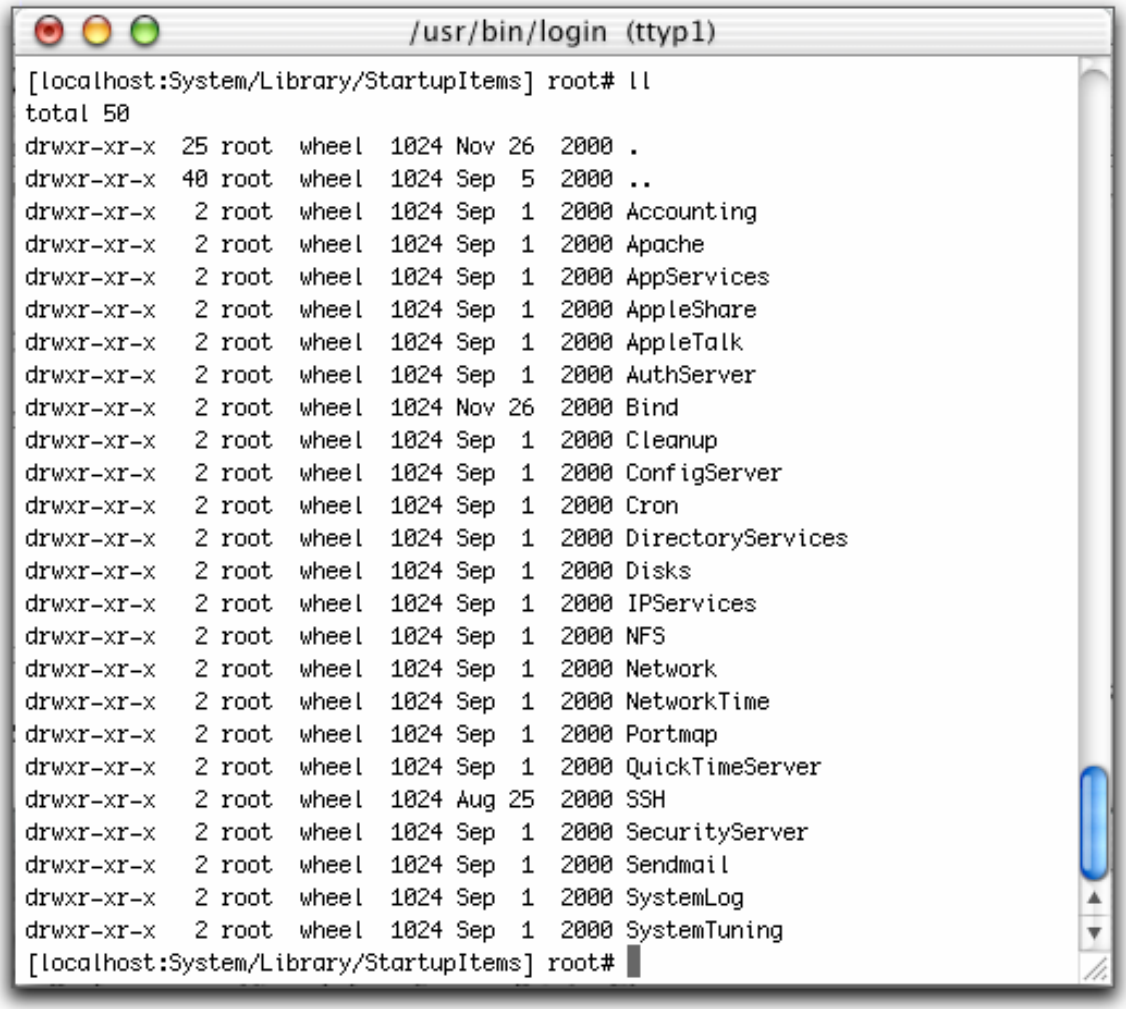
```
AUTODISKMOUNT=-REMOVABLE-
AUTOMOUNT=-YES-
BIND=-YES-
CONFIGSERVER=-NO-
IPFORWARDING=-NO-
MAILSERVER=-NO-
MANAGEMENTSERVER=-NO-
NETBOOTSERVER=-NO-
NISDOMAIN=-NO-
SSHSERVER=-YES-
TIMESYNC=-YES-
QTSSERVER=-NO-
WEBSERVER=-NO-
```

A typical default installation of the hostconfig file is shown below.

```
##
# /etc/hostconfig
##
# This file is maintained by the system control panels
##

# Network configuration
HOSTNAME=-AUTOMATIC-
ROUTER=-AUTOMATIC-

# Services
AFPSERVER=-NO-
APPLETALK=-NO-
AUTHSERVER=-NO-
AUTOCONFIG=-YES-
AUTODISKMOUNT=-REMOVABLE-
AUTOMOUNT=-YES-
CONFIGSERVER=-NO-
IPFORWARDING=-NO-
MAILSERVER=-NO-
MANAGEMENTSERVER=-NO-
NETINFOSERVER=-AUTOMATIC-
NETBOOTSERVER=-NO-
NISDOMAIN=-NO-
TIMESYNC=-NO-
QTSSERVER=-NO-
SSHSERVER=-NO-
WEBSERVER=-NO-
```

Again, we see that the hostname has been specified as well as the router on the target system. Both of these values are usually stored elsewhere in Mac OS X and are not placed here by the system control panels. Looking at the service listing, we see that an entry for Bind has been added to the normal service startup sequence of the system and the SSHSERVER has been set to initialize at startup.

A listing of the items located in the StartupItems folder is shown below.



Comparing this listing to a default installation of Mac OS X indicates that the only non-default item is the Bind folder. This result is also corroborated by the date listing above. This folder is also referenced by the hostconfig file and is set to run at startup. Overall, there appear to be no significant alterations to the startup environment of the system other than the addition of Bind, startup of SSH and the specification of the system networking characteristics.

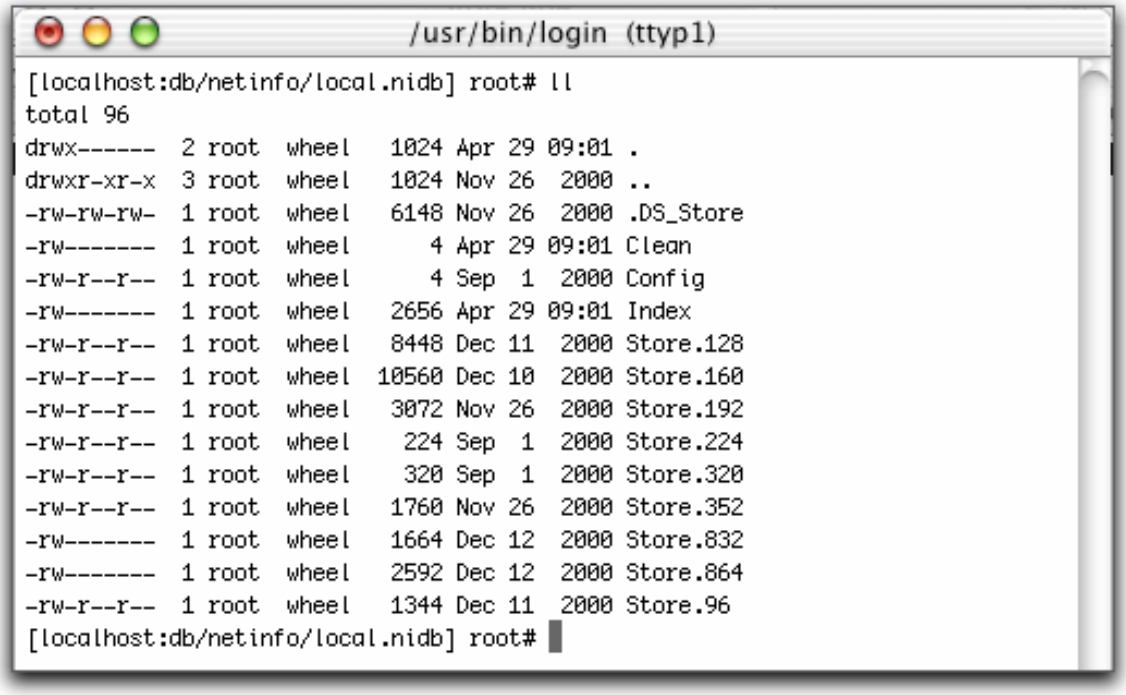
Examination of the other files in the /etc directory indicate the following changes have been made to the default system installation.

- The default router has been entered into a file called defaultrouter. Mac OS X does not normally utilize this configuration file.
- A gdb.conf file exists suggesting that the Developer Tools (Apple 6, p.1) or something similar has been installed. Gdb is not part of the default installation of Mac OS X.

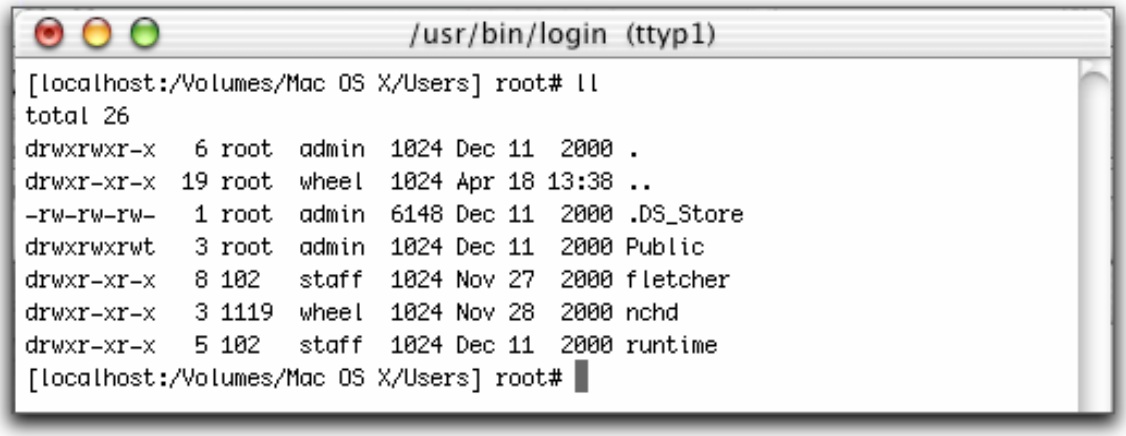
- The group file has been altered from default with several groups having been added.
- The hosts file has been altered with the addition of the fully qualified domain name of the local system.
- The iftab file has been altered with the addition of the IP of the local system.
- The master.password and the passwd file have been altered with additional user accounts.
- The motd file was altered with a warning about accessing the system.
- Several named related configuration files have been added to the directory. These files appear to support the Bind startup service. Examination of the files indicated that they were either "created by hand" by the SA on December 8, 2000 or by named-bootconf.pl.
- A file called pwd.db.tmp exists. This is the temporary insecure password database file. There is no reason for this file to still be present here.
- The services file has been altered with most services commented out. This file is only consulted when the system is booted into single-user mode.
- Multiple ssh configuration files have been created. This indicates that ssh was setup to run on the system.

Most of these changes appear to be typical system setup changes made for a traditional UNIX system and are actually unnecessary to the normal operation of Mac OS X.

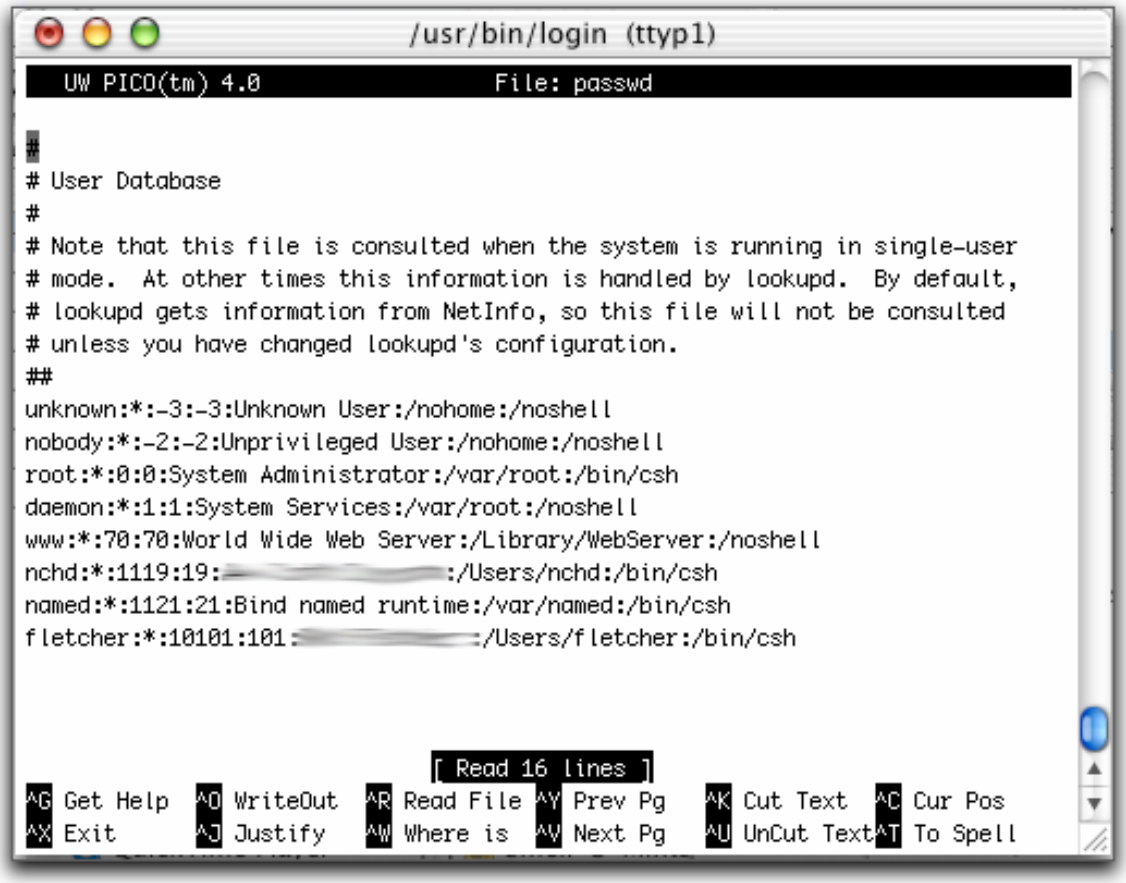
Now lets take a closer look at the user accounts on the system. User accounts are normally stored in the NetInfo (Apple 7, p.1) database. The database itself is composed of several files located in /var/db/netinfo/local.nidb as shown below. Examining the contents of Store.832 and Store.864, reveals that user accounts have been added to the system. In this case, three user accounts have been added: fletcher, nchd and named with home directories located at /Users/fletcher, /Users/nchd and /var/named respectively.



This information is corroborated by the directory listing of the /Users directory as shown below. In addition, there appears to be another user account called runtime that is not referenced in NetInfo. These home directories will be examined in further detail later.



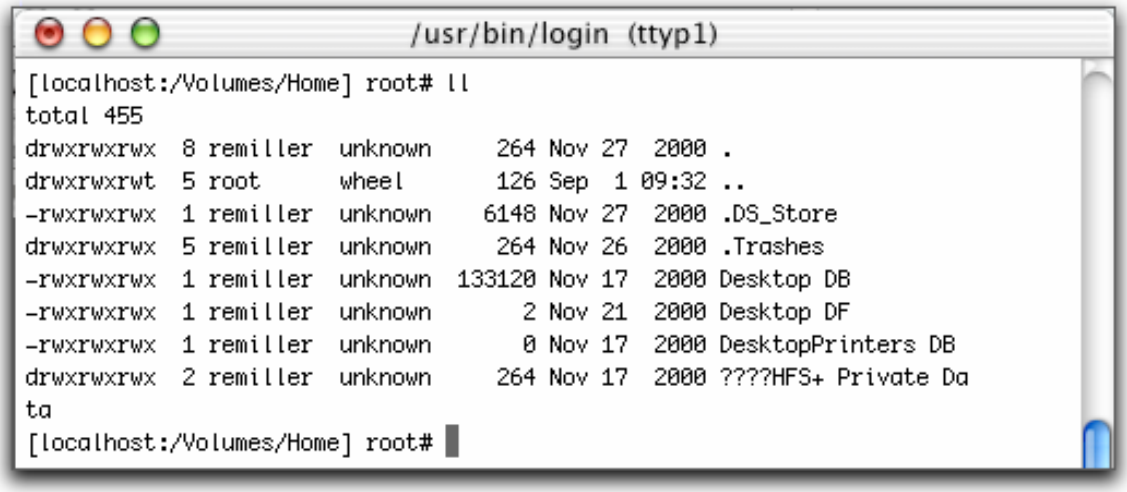
It has already been noted that the /etc/group, /etc/passwd and /etc/master.password files have been altered. The /etc/ passwd file is shown below, the real names of the fletcher and nchd accounts have been obfuscated. As noted in the comments, this file is only consulted when the system is booted into single-user mode. Despite this, all three of the user accounts found in NetInfo have been duplicated here as well as in the /etc/master.password file. Similar entries have been added to the /etc/group file as well.



Moving on with an examination of the overall directory structure produced some valuable information as outlined below:

- The /Application directory revealed that only default applications have been installed.
- The presence of the /Developer directory indicated that the Developer Tools were in fact installed. This was also confirmed by the Developer.pkg receipt in the /Library/Receipts folder. In addition, this explains the presence of the /etc/gdb.conf file as gdb is part of this package.
- There was no System Folder directory indicating that Mac OS 9.x (Apple 4, p.1) was not installed on the system, which prevented the usage of the Classic environment (Mac OS 9.x running as a process in Mac OS X for backwards compatibility) or the ability to boot into the older operating system.
- There was a folder called /Home on the boot volume disk2.img. This apparently was the mount point for the second hard disk represented by disk1.img. This is not the way that Mac OS X normally attempts to mount volumes.

- In the /usr directory, there were two folders that are not part of the normal installation of Mac OS X. They are /usr/X11R6 and /usr/Xfree86. Examination of these folders reveals the X11R6-darwin.tar file in the Xfree86 directory and an apparent remnant of this tar in the form of /usr/X11R6/bin/xauth.
- The file system on disk1.img ended up being mounted to /Volumes/Home on the forensic system and inspection of this volume shows it to be virtually empty, with only standard files present in the directory listing as shown below. Examination of the subfolders on this volume showed them to be empty.



Looking at the home directories of the user accounts on the system revealed more information. For the fletcher account, the home directory was /Users/fletcher and the following information was obtained from this directory:

- The user documentation for Mac OS X was stored in a tar file that was located here as well as some fragments of the files located in the archive.
- There were only the standard preferences in the ~/Library/Preferences folder. There was no Internet Explorer history file and the favorites.html file is the default.
- There were no files left in the .Trashes folder.
- The .cshrc file was the standard one that ships with the OS.
- A .ssh directory existed and within was a known host file that contained a single host and the associated key in it.
- The .tcsh_history file contained the command history of the fletcher account. The command history for fletcher appeared to be pretty standard. Essentially, the fletcher account used ssh, played with X11 and worked with named as shown below:

```
#+0974903327
sftp
```

```
#+0974903331
cd /usr/sbin
#+0974903336
dir named*
#+0974903342
alias
```

For the nchd account, the home directory was /Users/nchd and the following information was obtained from it:

- There was only one file in the directory and there were no sub-directories. This suggests that this account was not setup using the GUI tools of Mac OS X because they create a default directory structure as found in the fletcher home directory. In addition, the lack of any preference files suggests that this account was never used from the GUI.
- Using the *file* command on the only visible file called random.f indicates that it was an English text file. Examination of the file shows that it was a FORTRAN program used to generate random numbers called Corky_Randomizer.
- The .cshrc file was the standard one that ships with the OS.
- A .ssh directory existed and within it were two authorized_key files.
- The .tcsh_history file contained the command history of nchd. Besides setting up ssh, the user of nchd did not appear to be familiar with the system's syntax as shown below:

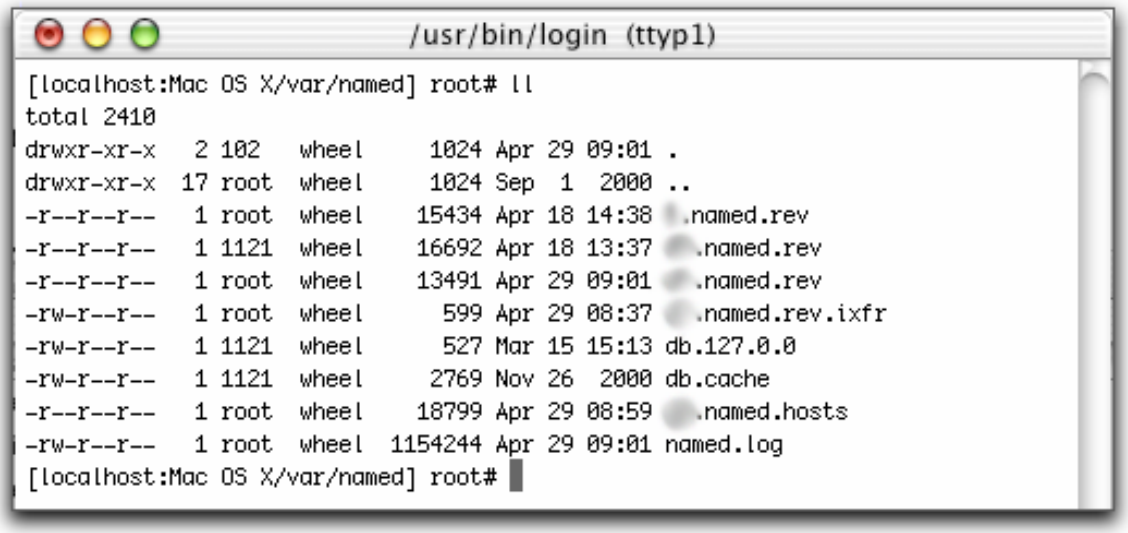
```
#+0975428631
ps -A
#+0975428636
ps -a
#+0975428639
ps
#+0975428652
man ps
#+0975428687
ps -e
#+0975428691
ps -x
#+0975428699
ps -E
#+0975428704
ps -N
#+0975428718
ps -C
#+0975428724
ps -h
#+0975428730
ps -j
#+0975428736
ps -l
#+0975428747
ps -mrS
#+0975428753
ps -T
#+0975428764
ps uvwx
#+0975428778
ps -L
#+0975428798
```

```
ps -aL
#+0975428813
ps -ax
#+0975428825
ps -ax | more
```

For the runtime account, the home directory was /Users/runtime and the following information was obtained from it:

- There were only the default user support folders and associated files.
- There were only the basic default preference files.
- There were no hidden files or directories nor any command history files.
- Overall, it appeared that this account was never used.

For the named account, the home directory was /var/named, which is not a default location for home directories in Mac OS X; suggesting again that it was not setup using the GUI. The only files in the directory are shown below with information specific to the organization masked.



Clearly, as the comments in the /etc/passwd file for this account indicate, this account is for the runtime for the named service. Examination of these files revealed that they were the DNS database files for Bind for the organization's domain. The named.log will be analyzed with the rest of the system logs later in the report.

The root account's home directory is normally located at /var/root by default under Mac OS X and this was the case with the target system. The following information was obtained from it:

- Within the ~/Library/Preferences folder there existed several preference files and folders which clearly indicated that this account was used from the GUI.
- In the Preferences folder there was an Internet Explorer history file and the contents of this file are shown below. Each link was verified and shown to point to the content that the link title suggests.

History

My Apple Start Page powered by Excite
http://livepage.apple.com/

Sunday, December 10, 2000

Sys Admin
Sys Admin
Search results
Mac OS X Developer Documentation
Developer Tools Overview
ADC Developer Documentation
Technical Notes
Compiler Release Notes
Developer Tools Overview
Developer Tools
Mac OS X
http://www.apple.com/developer/macosx/

Friday, December 8, 2000

Apple - Mac OS X
Apple

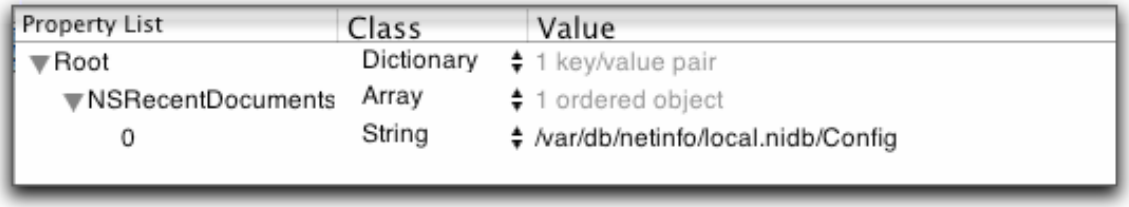
Monday, November 27, 2000

localhost

- The com.apple.finder.plist preference file revealed what folders were last used in the GUI as shown below.

Recen:Folders	Array	10 ordered objects
0	String	file://localhost/Applications/
1	String	file://localhost/Applications/Utilities/
2	String	file://localhost/Applications/Utilities/Help%20Viewer.app/
3	String	file://localhost/Users/Public/
4	String	file://localhost/Developer/
5	String	file://localhost/Developer/Sources/
6	String	file://localhost/Developer/Applications/
7	String	file://localhost/Developer/Tools/
8	String	file://localhost/Applications/Utilities/NetInfo%20Manager.app/
9	String	file://localhost/Applications/Utilities/Multiple%20Users.app/

- The preference file for mail existed in form of the com.apple.mail.plist file. Examination of this file showed that the built-in mail program was setup for the SA's organizational account.
- The com.apple.TextEdit.plist preference file revealed what file the text editor was used last on as shown below.



Property List	Class	Value
Root	Dictionary	1 key/value pair
NSRecentDocuments	Array	1 ordered object
0	String	/var/db/netinfo/local.nidb/Config

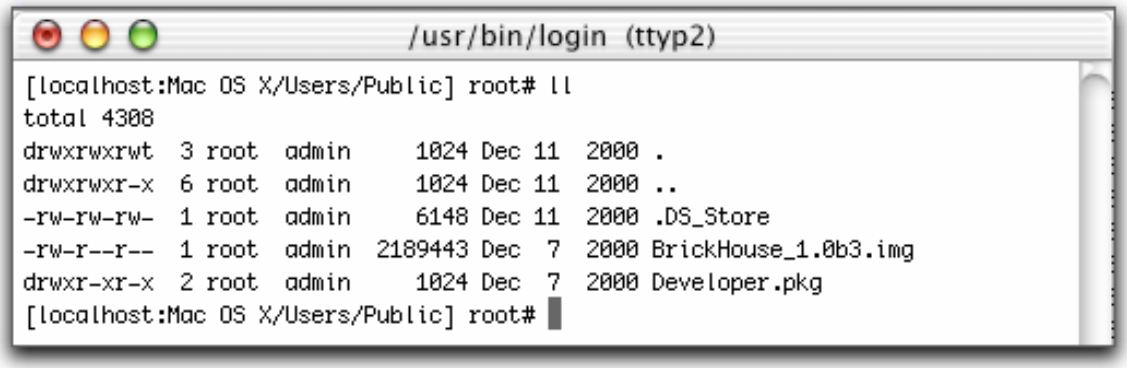
- There were no files in the .Trashes folder.
- The .cshrc file was the standard one that ships with the OS.
- A .ssh directory existed and within it were two known host files that corresponded to two hosts in the organization. In addition, there were identity files for the local system.
- There were two folders called MailAccounts and Mailboxes. In the ~/MailAccounts directory the mail of the SA from Saturday, November 18, 2000 through Monday, December 4, 2000 was found. In the ~/Mailboxes directory, the system messages root mail from Friday, November 17, 2000 through Wednesday, December 20, 2000 were found.
- The .tcsh_history file contained the command history of the root account. Perusing this file showed mostly system maintenance and named configuration and testing such as that shown below (some information has been replaced with xxx's).

```
#+1015446230
cd /etc
```

```
#+1015446233
dir *named*
#+1015446249
scp named.conf* root@xxxxxxxx:/etc
#+1015446269
cd /var/named
#+1015446278
dir *named*
#+1015446288
rm named.log*
#+1016230354
cd /var/named
#+1016230362
grep xxxxxx *
#+1016230381
vi db.127.0.0
#+1016230418
grep xxxxxx *
#+1016230430
ls -al
#+1016230445
shutdown
```

Due to the shear volume of information left in this account, it appeared that most of the usage of this system was done through root probably using ssh. The fletcher account was used sparingly by the SA, the runtime account was never really used and probably was intended for the purpose that the named account was setup for, which was simply as a service runtime account for Bind. The nchd account appears to have been setup for a second party (based on the real name–an individual not affiliated with the organization) and it was only used remotely probably through ssh.

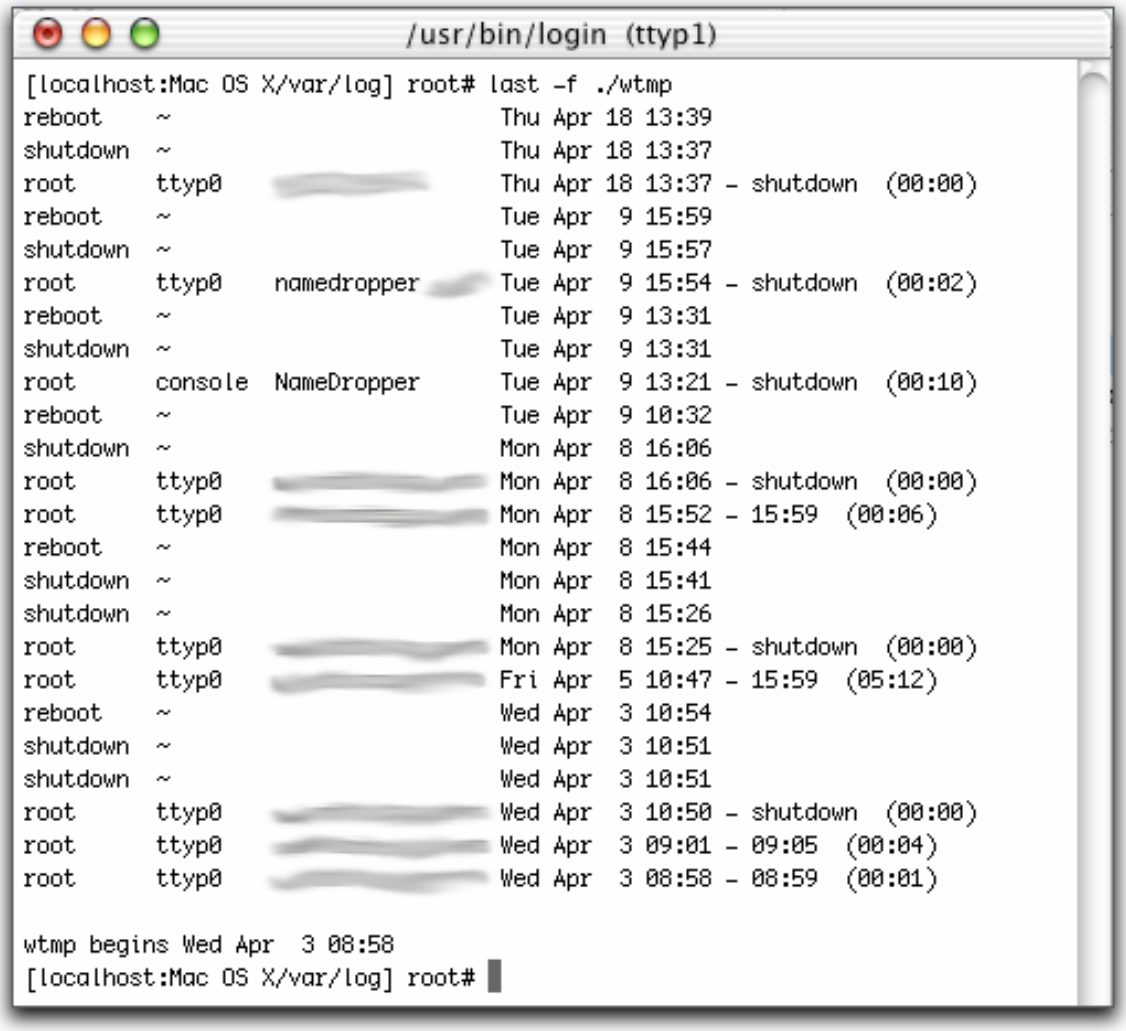
There was a Public folder that is created by default by Mac OS X in the Users directory. Investigating the contents of this folder reveals two objects as shown below.



BrickHouse_1.0b3.img (Hill, p.1) was a shareware program used to manipulate the built-in firewall of Mac OS X. Examination of the image verified that this was indeed the BrickHouse program. There is no evidence that this program was ever installed on the system. The Developer.pkg object is the installation package for the Developer Tools that was noted as being installed previously based on the /Library/Receipts directory. Examination of the contents of this package verified that it was the original package from Apple.

Moving on now to a review of the system level log files showed that the /var/log folder had the last open log files and several archived logs for the following services: ftp, lookupd, lpr, mail, netinfo, system.log and wtmp. Further investigation of these files and their associated archives showed that all except the wtmp logs were completely empty despite being configured otherwise. Normally, this would be cause for concern indicating that there was a compromise and that the intruder had covered their tracks by clearing the log files. However, since this was a beta of an operating system that expired over a year prior to its last boot date, it is more likely that the lack of logging was either an artifact of the lack of maturity of the code or the built-in expiration of the Public Beta.

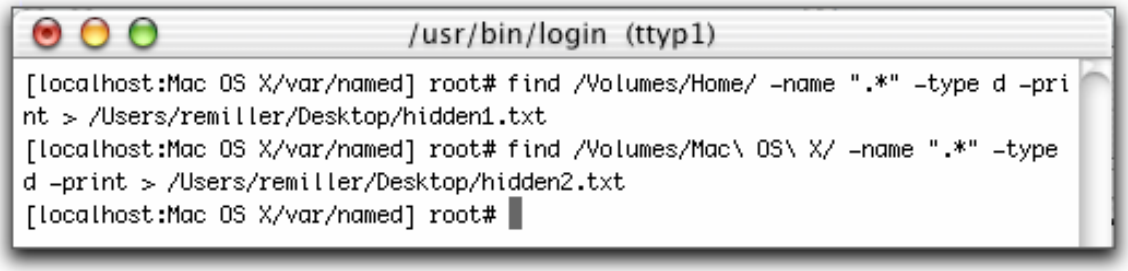
There was information logged to the wtmp files. Examination of this file using the *last* command revealed the following (hostnames and IP addresses have been obfuscated):



Extracting the archived wtmp log files and examining them as above revealed all login, shutdown and restart activity dating back to September 6, 2001. In that time, the root account was used almost exclusively from a variety of organizational systems. The fletcher account was used much less and the nchd account was only used once in this timeframe from an external network (a local ISP) and then only for a single minute. This information verifies the assumptions made based on the amount and type of information obtained from the home directories.

Returning to the named.log found in /var/named, inspection of the 1.1 MB file revealed a variety of information from Bind. This information varied from unapproved update warnings to cache cleaning notices. Virtually all of the unapproved update warnings were generated from organizational systems and the rest of the information was typical process notices. There were no unusual entries found in a search of this log file.

Next, a search for hidden directories was performed. The following commands were issued to extract this information from each disk image as shown below:



Examination of the hidden1.txt reveals only the following hidden directory on disk1.img:

```
/Volumes/Home//.Trashes
```

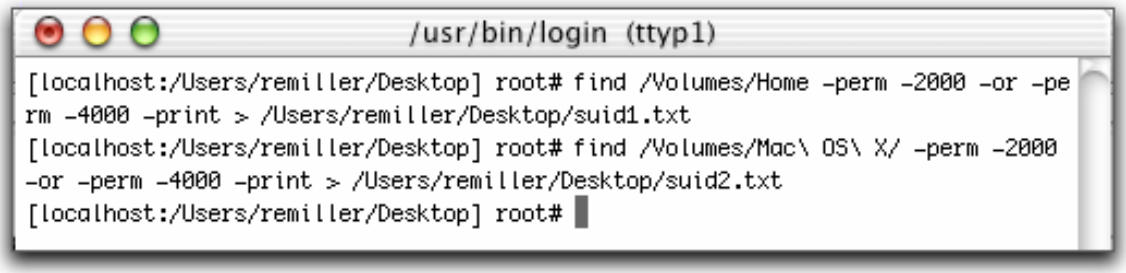
This directory was investigated and found to contain neither files nor any sub-directories.

Examination of hidden2.txt reveals the following hidden directories on the system disk:

```
/Volumes/Mac OS X//private/var/root/Library/Internet Search Sites/.Downloads
/Volumes/Mac OS X//private/var/root/.ssh
/Volumes/Mac OS X//private/var/root/.Trash
/Volumes/Mac OS X//.vol
/Volumes/Mac OS X//.ssh
/Volumes/Mac OS X//Users/fletcher/.Trash
/Volumes/Mac OS X//Users/fletcher/.ssh
/Volumes/Mac OS X//Users/nchd/.ssh
/Volumes/Mac OS X//.Trashes
```

In our original search through the directory structure, some of these hidden directories were missed. Going back over these directories revealed that these directories are either empty or contain legitimate files (e.g., .ssh contained identity files.)

A search for all SUID and GUID files was performed. The following commands were used to find these files and the resulting output was analyzed.



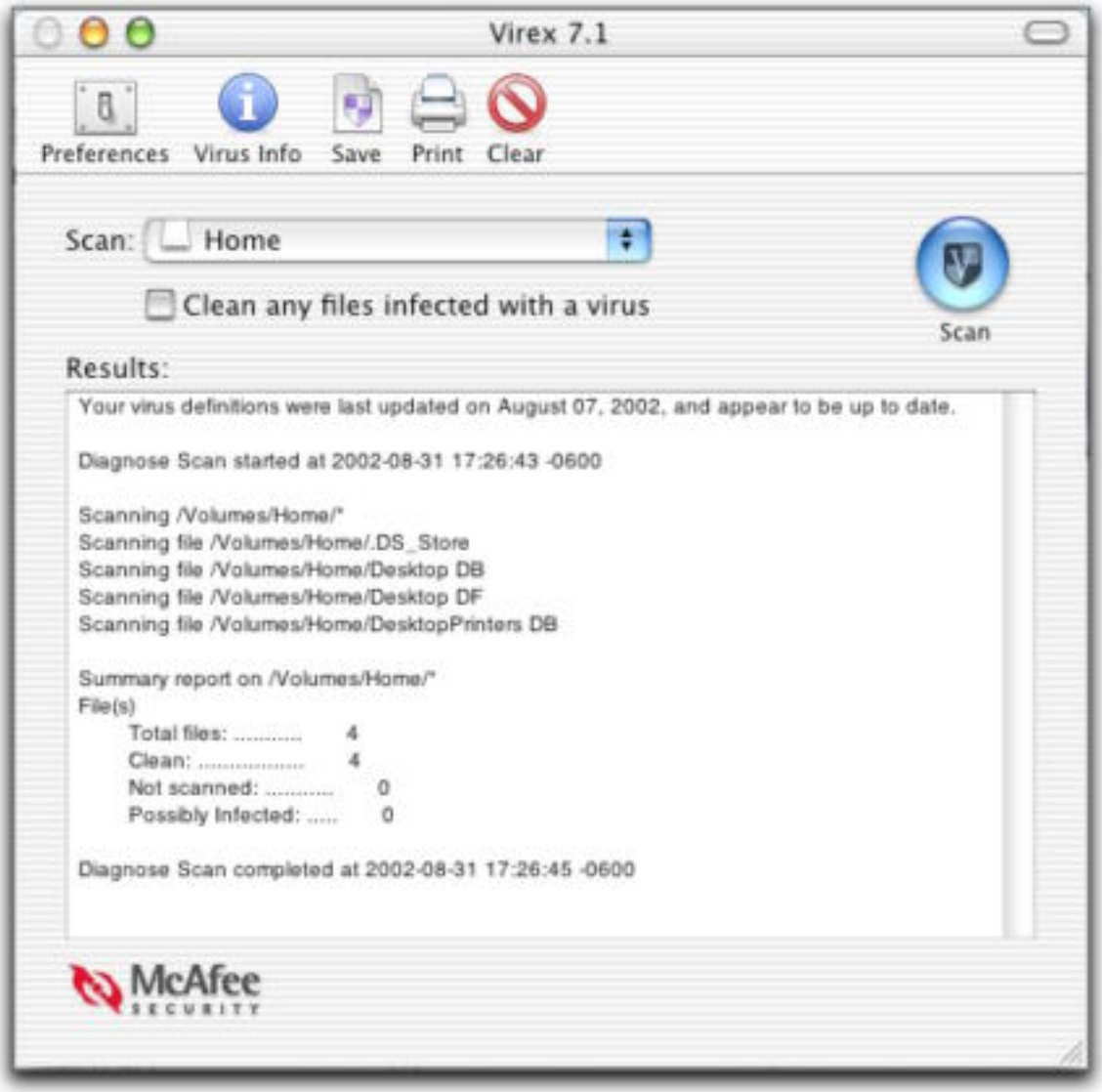
Suid1.txt was empty, which is not a surprise considering the contents of that volume. Suid2.txt is shown below. Examination of each file showed the following files had either the SUID or GUID bit set:

```
/Volumes/Mac OS X//Applications/Utilities/Disk Utility.app/Contents/MacOS/Disk
Utility
/Volumes/Mac OS X//Applications/Utilities/NetInfoManager.app/Contents/MacOS
/NetInfoManager
/Volumes/Mac OS X//Applications/Utilities/Print Center.app/Contents/MacOS/Print
Center
/Volumes/Mac OS X//Applications/Utilities/Terminal.app/Contents/MacOS/Terminal
/Volumes/Mac OS X//Applications/Classic.app/Contents/Resources/TruBlueEnvironment
/Volumes/Mac OS
X//System/Library/Frameworks/NSLCore.framework/Versions/A/Resources/NSLPlugins/slp
dLoad
/Volumes/Mac OS X//System/Library/Frameworks/PrintCore.framework/Versions
/A/Libraries/PrintServer
/Volumes/Mac OS X//System/Library/StartupItems/SecurityServer/enable
/Volumes/Mac OS X//System/Library/Filesystems/AppleShare/afpLoad
/Volumes/Mac OS X//System/Library/Filesystems/hfs.fs/hfs.util
/Volumes/Mac OS X//System/Library/Filesystems/cd9660.fs/cd9660.util
/Volumes/Mac OS X//bin/ps
/Volumes/Mac OS X//bin/rcp
/Volumes/Mac OS X//sbin/ping
/Volumes/Mac OS X//sbin/route
/Volumes/Mac OS X//sbin/shutdown
/Volumes/Mac OS X//usr/bin/lpr_ot
/Volumes/Mac OS X//usr/bin/login
/Volumes/Mac OS X//usr/bin/crontab
/Volumes/Mac OS X//usr/bin/quota
/Volumes/Mac OS X//usr/bin/ksu
/Volumes/Mac OS X//usr/bin/rlogin
/Volumes/Mac OS X//usr/bin/rsh
/Volumes/Mac OS X//usr/bin/su
/Volumes/Mac OS X//usr/bin/at
/Volumes/Mac OS X//usr/bin/atq
/Volumes/Mac OS X//usr/bin/sudo
/Volumes/Mac OS X//usr/bin/chpass
/Volumes/Mac OS X//usr/bin/chfn
/Volumes/Mac OS X//usr/bin/atrm
/Volumes/Mac OS X//usr/bin/batch
/Volumes/Mac OS X//usr/bin/chsh
/Volumes/Mac OS X//usr/bin/passwd
/Volumes/Mac OS X//usr/bin/top
/Volumes/Mac OS X//usr/bin/latency
```

```
/Volumes/Mac OS X//usr/libexec/load_webdav
/Volumes/Mac OS X//usr/libexec/chkpasswd
/Volumes/Mac OS X//usr/libexec/load_hdi
/Volumes/Mac OS X//usr/sbin/netstat
/Volumes/Mac OS X//usr/sbin/sendmail
/Volumes/Mac OS X//usr/sbin/DirectoryService
/Volumes/Mac OS X//usr/sbin/sliplogin
/Volumes/Mac OS X//usr/sbin/traceroute
/Volumes/Mac OS X//usr/sbin/pppd
```

Comparing these files with information obtained from the installation disk of Mac OS X Public Beta indicates that these files have not been altered and that they had the SUID or GUID bit set by default in the original distribution.

Finally, each disk image was scanned for malware with McAfee Virex 7.1 using the August 7, 2002 virus definitions as shown below. The scans indicated that there was no malware as defined by the virus definitions on either volume.



MACTime Analysis

While in single-user mode to reduce GUI interactions, the MACTimes for each disk image were extracted using *mac-robber* as follows:

```
localhost# disktool -c 0
localhost# hdiutil mount /Users/remiller/Desktop/GCFA/Part\ I/disk1.img
localhost# cd /Users/remiller/Desktop/GCFA/Tools/mac-robber-1.00
localhost# ./mac-robber /Volumes/Home > /Users/remiller/Desktop/mactimes1.txt

localhost# hdiutil mount /Users/remiller/morgue/disk2.img
localhost# ./mac-robber /Volumes/Mac\ OS\ X/ >
/Users/remiller/Desktop/mactimes2.txt
```

The extracted MACTimes were processed into a timeline using the *mactime* component of TASK as follows with the group and password files being defined from the original image as well as the time zone as determined previously.

```
localhost# cd /Users/remiller/Desktop/task-1.00/bin
localhost# ./mactime -b mactimes1.txt -g /Volumes/Mac\ OS\ X/etc/group -p
/Volumes/Mac\ OS\ X/etc/passwd -y -z MST7MDT > mactimes1a.txt
localhost# ./mactime -b mactimes2.txt -g /Volumes/Mac\ OS\ X/etc/group -p
/Volumes/Mac\ OS\ X/etc/passwd -y -z MST7MDT > mactimes2a.txt
```

The complete timeline for disk1.img is shown below. Clearly, the first entry is not a valid time and probably is representative of bad MACTime properties on the installation disk for the indicated file. The remaining files on the disk have MACTimes most likely related to when the disk was first initialized (Nov 17, 2000) and then when the Finder created trash cans for different users (noted by the specific id number of the directories in .Trashes.) The other entries are normal hidden system files produced by Mac OS X.

```
1969 Dec 31 17:00:00      0 m.. -rwxrwxrwx root 99    /Volumes/Home/DesktopPrinters DB
                          0 .a. -rwxrwxrwx root 99    /Volumes/Home/DesktopPrinters DB
                          0 ..c -rwxrwxrwx root 99    /Volumes/Home/DesktopPrinters DB
2000 Nov 17 02:46:05    264 mac drwxrwxrwx root 99 19 /Volumes/Home/HFS+ Private Data
2000 Nov 17 09:28:04 133120 mac -rwxrwxrwx root 99 17 /Volumes/Home/Desktop DB
                          0 mac -rwxrwxrwx root 99 18 /Volumes/Home/DesktopPrinters DB
2000 Nov 17 10:04:02    264 mac drwxrwxrwx root 99 21 /Volumes/Home/.Trashes/102
2000 Nov 21 08:27:56      2 mac -rwxrwxrwx root 99 16 /Volumes/Home/Desktop DF
2000 Nov 22 08:43:51    264 mac drwxrwxrwx root 99 23 /Volumes/Home/.Trashes/0
2000 Nov 26 23:54:53    264 mac drwxrwxrwx root 99 24 /Volumes/Home/.Trashes/1121
                        264 mac drwxrwxrwx root 99 20 /Volumes/Home/.Trashes
2000 Nov 27 10:15:32   6148 mac -rwxrwxrwx root 99 25 /Volumes/Home/.DS_Store
```

Turning to the timeline for disk2.img, the following items were the very first entries. The associated times clearly represent default times from the installation disk and do not represent the actual installation date.

```
1969 Dec 31 17:00:00 5120 m.. drwxrwxrwt root wheel /Volumes/Mac OS X/lost+found
                     5120 .a. drwxrwxrwt root wheel /Volumes/Mac OS X/lost+found
                     5120 ..c drwxrwxrwt root wheel /Volumes/Mac OS X/lost+found
1996 Jan 05 01:07:52  776 ma. -r--r--r-- root wheel 63832 /Volumes/Mac OS
X/Developer/Applications/ProjectBuilderWO.app/Contents/Resources/English.lproj/Viewer.nib
/data.classes
```

The next several hundred entries show only modify and access flags for system files dated from January 5, 1996 through September 6, 2000. Beginning on November 17, 2000 the change flag for entries began to be set. By extracting the entries for that day, the following timeline was generated (the file size, group ownership and inode information were removed and the paths were truncated in order to better display the timeline in this document):

```
2000 Nov 17 02:36:00 m.. dr-xr-xr-x root /Volumes/Mac OS X/dev
                     ma. dr--r--r-- root /Volumes/Mac OS X/.vol
2000 Nov 17 02:36:55 .a. dr-xr-xr-x root /Volumes/Mac OS X/dev
2000 Nov 17 02:46:46 .a. -rw-rw-r-- root /…/MinimalSystem.bom
2000 Nov 17 02:46:47 m.. -rw-rw-r-- root /…/MinimalSystem.bom
2000 Nov 17 02:46:49 ma. -rw-rw-r-- root /…/Receipts/Essentials.pkg/Essentials.loc
```

```
                     .a. -rw-rw-r-- root /…/Receipts/Essentials.pkg/Essentials.bom
2000 Nov 17 02:46:50 m.. -rw-rw-r-- root /…/Receipts/Essentials.pkg/Essentials.bom
2000 Nov 17 02:46:55 ma. -rw-rw-r-- root /…/Receipts/BSD.pkg/BSD.loc
                     ma. -rw-rw-r-- root /…/Receipts/BSD.pkg/BSD.bom
2000 Nov 17 02:48:37 ..c -rwxr-xr-x root /…/CFMSupport/._CarbonLibLite
                     ..c -rw-r--r-- root /…/CFMSupport/BridgeLibs/._AE.bridge
                     ..c -rw-r--r-- root /…/CFMSupport/._StdCLib
```

where "…" represents "Volumes/Mac OS X/Library"

What is apparent here is that volume information was first accessed. Then the receipts for the install itself were accessed 10 minutes later. Finally, system files began to have their change flag set indicating the beginning of the actual OS installation. This process goes on for nearly an hour with thousands of files being created in the process. After about an hour, the modified and accessed flags started showing up again in the timeline as shown below. (Note that the file sizes, group information and inode information were removed and the paths were truncated as shown in order to better display the timeline.)

```
2000 Nov 17 03:47:42 m.c drwx------ root /…/private/var/cron
                     ..c -rw-r--r-- root /…/private/etc/.AppleTimeZoneCache
                     ..c -rw-r--r-- root /…/private/var/db/.ApplePrefsCache
2000 Nov 17 04:16:48 m.c drwxr-xr-x root /…/private/var/msgs
2000 Nov 17 04:16:49 m.c drwxr-xr-x root /…/private/var/backups
2000 Nov 17 04:16:50 m.c -rw-r--r-- root /…/private/etc/dumpdates
2000 Nov 17 08:45:55 m.c -rw-r--r-- root /…/System/Library/._Appearance
                     m.c drwxr-xr-x root /…/System/Library/Appearance
                     mac -rw-r--r-- root /…/System/Library/Appearance/._Sound Sets
                     m.c drwxr-xr-x root /…/System/Library/Appearance/Sound Sets
2000 Nov 17 08:48:45 m.c -rw-r--r-- root /…/private/var/db/.AppleSetupDone
2000 Nov 17 08:48:49 ..c drwxr-xr-x 102 /…/Users/fletcher/Library/Favorites
                     ..c drwxr-xr-x 102 /…/Users/fletcher/Library/Preferences/Explorer
                     ..c lrwxr-xr-x 102 /…/Users/fletcher/Library/Favorites/Documents
                     ..c drwxr-xr-x 102 /…/Users/fletcher/Documents
                     ..c lrwxr-xr-x 102 /…/Users/fletcher/Library/Favorites/Home
```

where "…" represents "Volumes/Mac OS X/"

The end of the installation process shows the cron, timezone cache and preference cache files being set. About 25 minutes later, the system modifies and changes some system related directories and files and apparently sits idle for four and half hours. At this time, a file indicating that the setup is completed is modified and changed (.AppleSetupDone) and we begin to see activity in the fletcher account with regards to the basic directory structure being created. This suggests that the first account created on the system was the fletcher account, which represents the "administrator" account under Mac OS X.

The next significant activity that takes place on the system occured on December 7, 2000. As shown below, two files were created in the /Users/Public directory. Both of these files were previously investigated and discussed. The timeline shows that they were downloaded together on this date and that the Developer Tools were subsequently installed as shown by the last entry. The installation took about 10 minutes. (Note that the file sizes, group information and inode information were removed and the paths were truncated as shown in order to better display the timeline.)

```
2000 Dec 07 10:12:19 m.c -rw-r--r-- root /…/Users/Public/BrickHouse_1.0b3.img
2000 Dec 07 10:16:19 m.c -rw-r--r-- root /…/Users/Public/Developer.pkg/Developer.bom
                     m.c -rw-r--r-- root /…/Users/Public/Developer.pkg/Developer.info
                     ma. -rw-r--r-- root /…/Library/Receipts/Developer.pkg/
                                          Developer.info
2000 Dec 07 10:25:49 m.c -rw-r--r-- root /…/Users/Public/Developer.pkg/post_install
```

where "…" represents "Volumes/Mac OS X/"

Timeline entries related to the second-party account nchd were extracted using *grep*. The results are shown below. (Note that the file sizes and inode information were removed and the paths were truncated as shown in order to better display the timeline.)

```
2000 Nov 26 17:50:45 m.. -rw------- nchd     wheel /…/nchd/.ssh/authorized_keys2
2000 Nov 26 18:20:48 m.. -rw------- nchd     wheel /…/nchd/.ssh/authorized_keys
2000 Nov 27 15:22:35 ..c -rw------- nchd     wheel /…/nchd/.ssh/authorized_keys
2000 Nov 27 15:22:43 ..c -rw------- nchd     wheel /…/nchd/.ssh/authorized_keys2
2000 Nov 27 15:26:35 .a. -rw-r--r-- nchd     wheel /…/nchd/random.f
2000 Nov 27 15:26:40 m.c -rw-r--r-- nchd     wheel /…/nchd/random.f
2000 Nov 28 08:53:35 m.c drwxr-xr-x nchd     wheel /…/nchd
2000 Nov 28 09:07:08 m.c -rwxr-xr-- nchd     wheel /…/nchd/.cshrc
2000 Dec 13 10:08:01 .a. -rw------- nchd     wheel /…/nchd/.ssh/authorized_keys2
2001 Sep 10 13:27:55 .a. -rw------- nchd     wheel /…/nchd/.ssh/authorized_keys
                     .a. -rw------- nchd     wheel /…/nchd/.tcsh_history
                     .a. -rwxr-xr-- nchd     wheel /…/nchd/.cshrc
2001 Sep 10 13:28:18 .a. -rw------- nchd     wheel /…/nchd/.ssh/prng_seed
                     m.c drwxr-xr-x nchd     wheel /…/nchd/.ssh
2001 Sep 10 13:28:25 m.c -rw------- nchd     wheel /…/nchd/.ssh/prng_seed
2001 Sep 10 13:28:55 m.c -rw------- nchd     wheel /…/nchd/.tcsh_history
                     .a. drwxr-xr-x nchd     wheel /…/nchd/.ssh
                     .a. drwxr-xr-x nchd     wheel /…/nchd
```

where "…" represents "Volumes/Mac OS X/Users"

Clearly, the first apparent use of this account was through ssh on November 26, 2000 when the ssh authorization keys were created. The next day, the previously investigated random.f file was created. Activity from this point on appears to be rather limited with the final activity occurring almost a year later on September 10, 2001.

The last time the system was booted is illustrated below. The last file to be utilized according to the timeline was the named.log file on April 29, 2002. This date corresponds with the last boot date as determined from the dates on the various log files in the /var/log directory as previously noted. (Note that the file sizes and inode information were removed and the paths were truncated as shown in order to better display the timeline.)

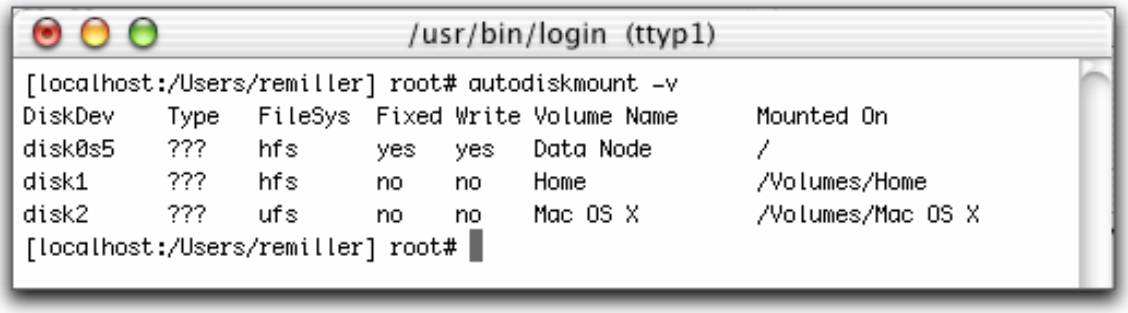
```
2002 Apr 29 09:01:05 m.c drwxr-xr-x root     wheel /…/var/run
                     m.c drwxr-xr-x 102      wheel /…/var/named
                     m.c -r--r--r-- root     wheel /…/var/named/xxx.named.rev
                     m.c -rw------- root     wheel /…/var/db/netinfo/local.nidb/Clean
                     mac -rw------- root     wheel /…/var/db/netinfo/local.nidb/Index
                     m.c drwx------ root     wheel /…/var/db/netinfo/local.nidb
                     m.c -rw-r--r-- root     wheel /…/var/named/named.log
2002 Aug 24 15:18:17 .a. lrwxr-xr-x root     wheel /Volumes/Mac OS X/mach
```

where "…" represents "Volumes/Mac OS X/Private"

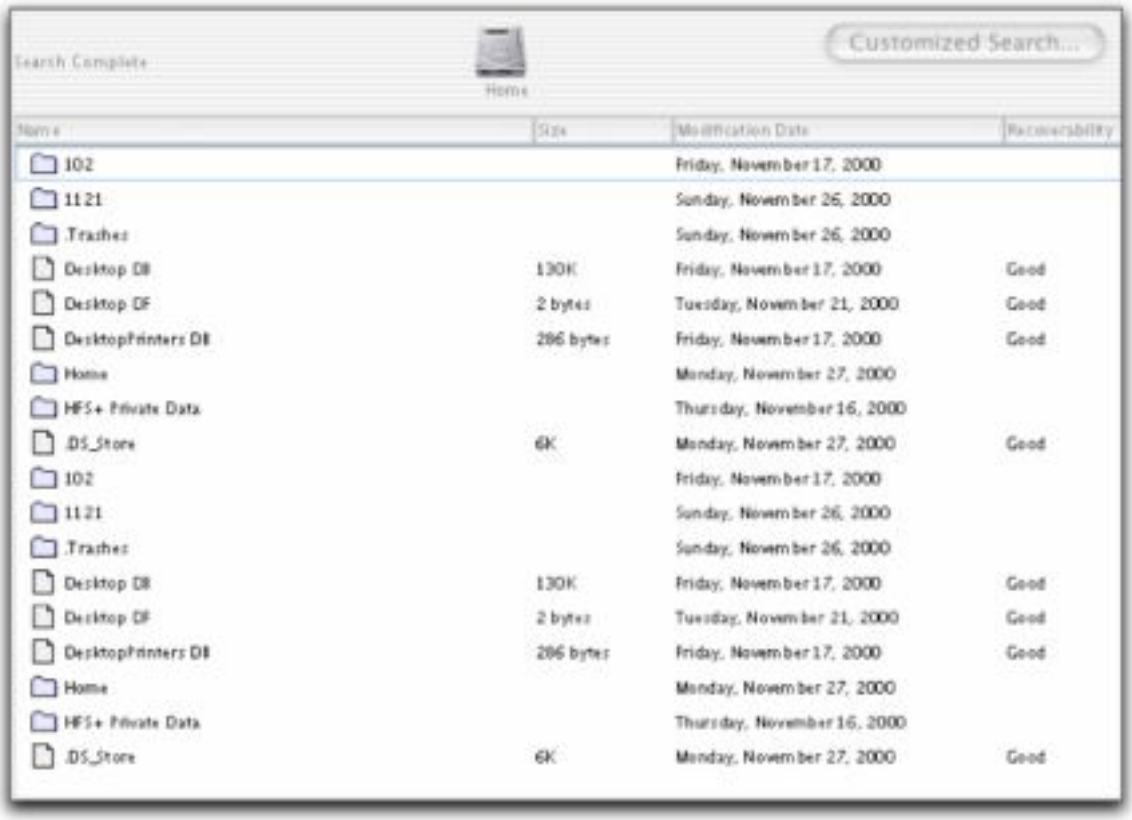
The final entry shown above is a symbolic link that had an access time equivalent to the time that the MACTimes were extracted from the disk image. All the subsequent entries are also symbolic links that were accessed in the process of MACTime extraction. This is most likely due to the fact that Mac OS X does not have the capability to mount images with the noatime parameter as previously noted. Therefore, the last real entry in the timeline is the named.log on April 29, 2002.

File Recovery

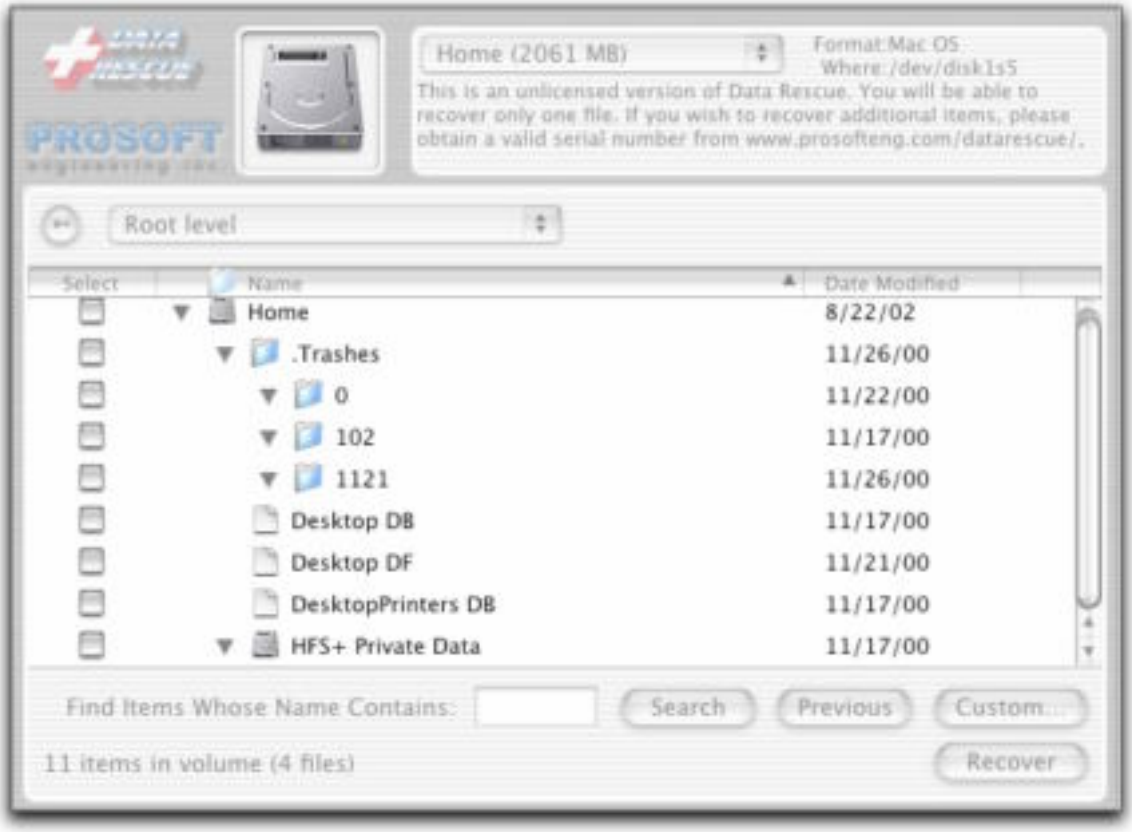
As evident by the screen shot below, the drive images have two different file systems associated with them. The first image has a HFS file system and the second image has a UFS file system. This will necessitate the usage of different tools and techniques to recover deleted files from the two images.



For the HFS file system, two tools were utilized to try and recover deleted files. The first was Norton Utilities for Macintosh UnErase 7.0. UnErase scanned the entire disk image using both a Catalog Tree Search as well as a File Type Search with all 130 file types and resource forks selected. These searches found 18 items on disk1.img. Of these, nine were duplicates as shown below leaving nine actual recoverable items. The modification date represents the most likely date that the files were deleted.



The demo version of ProSoft Data Rescue X 10.0 was also used to scan the drive for recoverable files. Using a Thorough Scan, the program found eight recoverable items as shown below with the modified date being the most likely date that the files were deleted.



Since both applications found essentially the same files, UnErase was used to recover the files and folders. These items were examined and found to either be empty folders or the duplicates of the default hidden files that Mac OS creates on each volume (i.e., Desktop DB, Desktop DF and DesktopPrinters DB). A strings inspection of the Desktop DB file which keeps track of files and applications on the disk (Apple 5, p.1) revealed some very interesting information as shown below:

```
@(#)$Header: IRIX 6.2:1232792120 built 03/09/96 at borg:/vince/6.2-mar09/root $
@(#) Copyright (c) 1983 Regents of the University of California.
 All rights reserved.
```

This and other information found by strings in this file strongly suggests that this disk was previously used in an SGI workstation. This would also provide an explanation as to why neither UnErase nor Data Rescue were able to recover any files besides the default initialization files; the file system previously on the disk was probably XFS and neither application understands that file system.

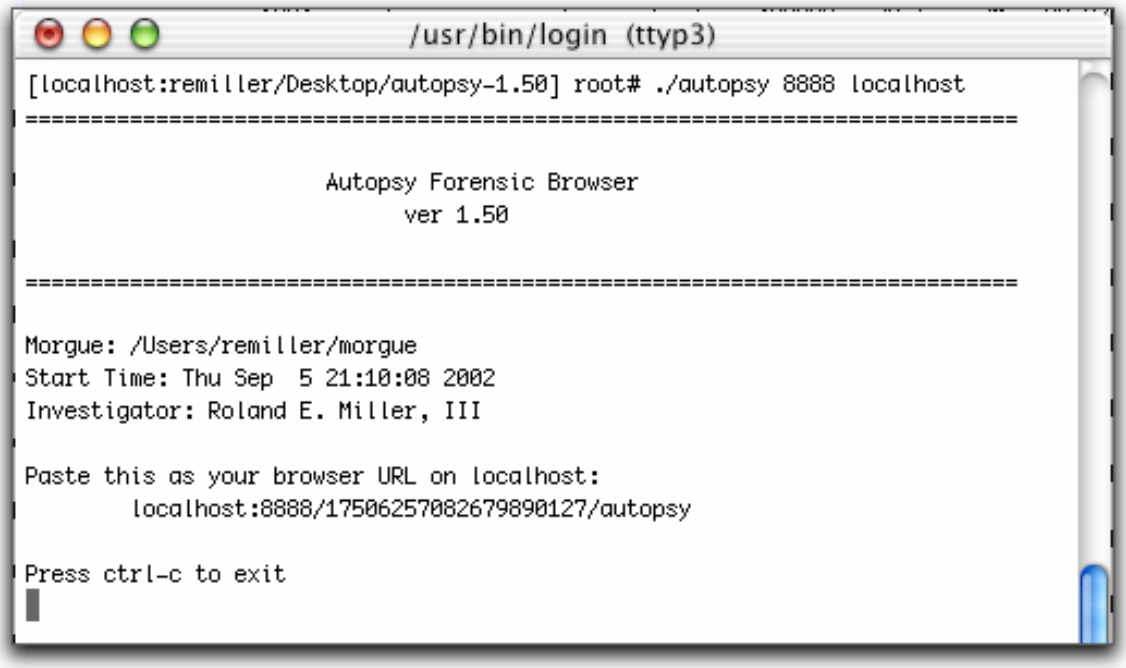
For the UFS formatted disk image, neither UnErase nor Data Rescue support the UFS file system. However, with the assistance of the author of TASK, it was determined that the OpenBSD setting (i.e., the FFS file system) in TASK would work for UFS. Therefore,

TASK and Autopsy were used to examine the image for deleted files. It should be noted here that TASK does not support the HFS or the HFS+ file systems.

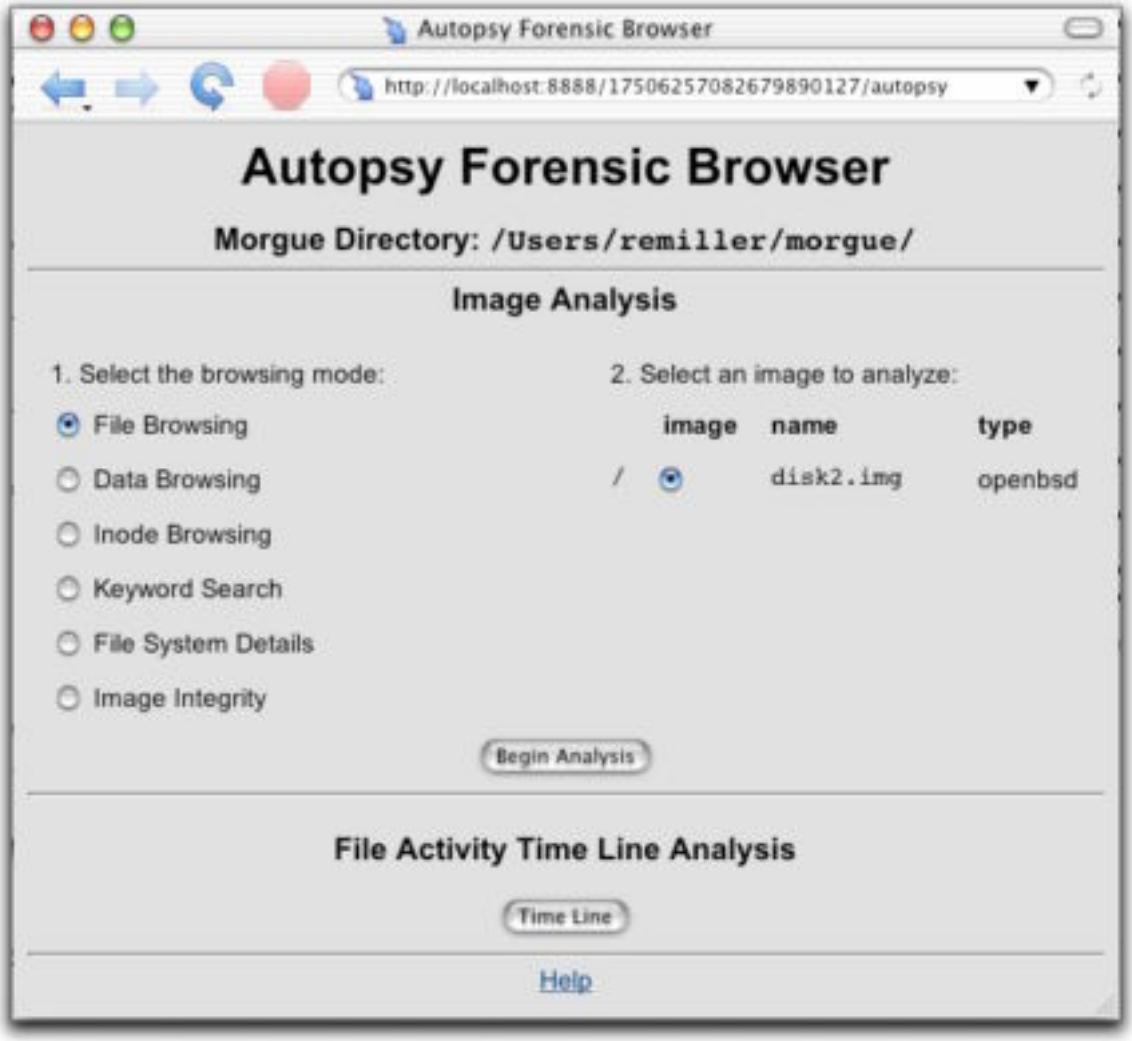
Autopsy was used to interface with TASK in the analysis of the UFS formatted disk image. Following the basic Autopsy README, an fsmorgue file was created and placed in the morgue directory that was specified when Autopsy was compiled along with the disk image and the original MD5 hash in a file called md5.txt. The fsmorgue file had the following initial contents:

```
disk2.img       openbsd         /       MST
```

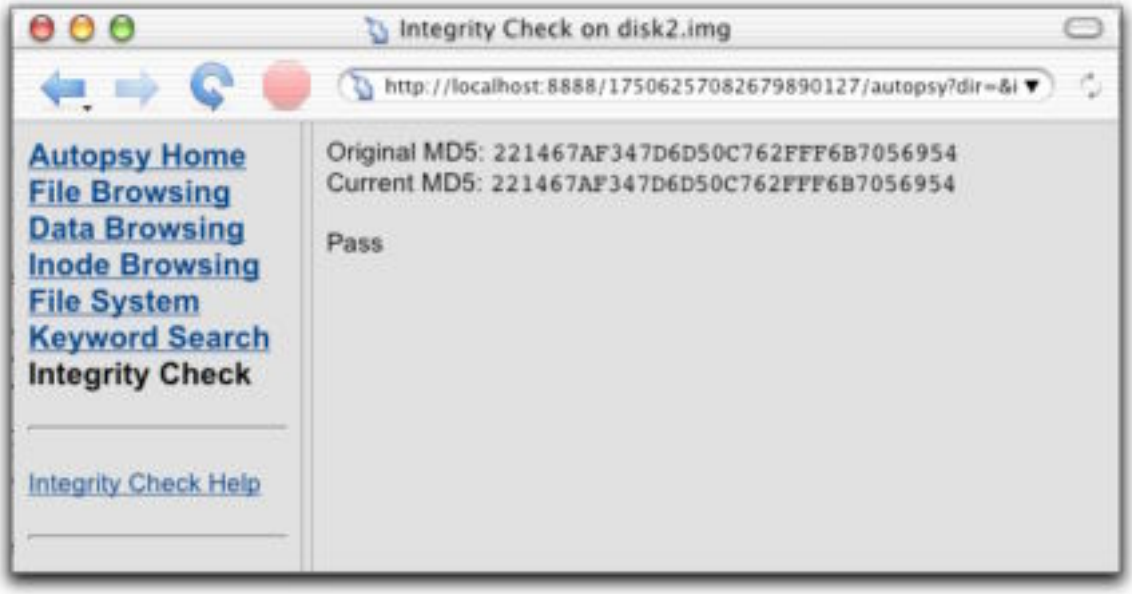
The autopsy daemon was then started as shown below:



The URL shown above was pasted into OmniWeb 4.1 (Omni, p.1) and the following page was displayed:



The first step was to verify image integrity by calculating and comparing the MD5 hash to the one obtained originally on the forensic imaging system. The Image Integrity option was selected along with the disk2.img and the Begin Analysis button was pressed. The following page was generated after several seconds, which verified that the current image being analyzed was an identical copy of the original hard disk:



Next, selecting File Browsing and then clicking on All Deleted Files brings up the page where deleted files, as uncovered by subcomponents of TASK, can be further analyzed as shown below.

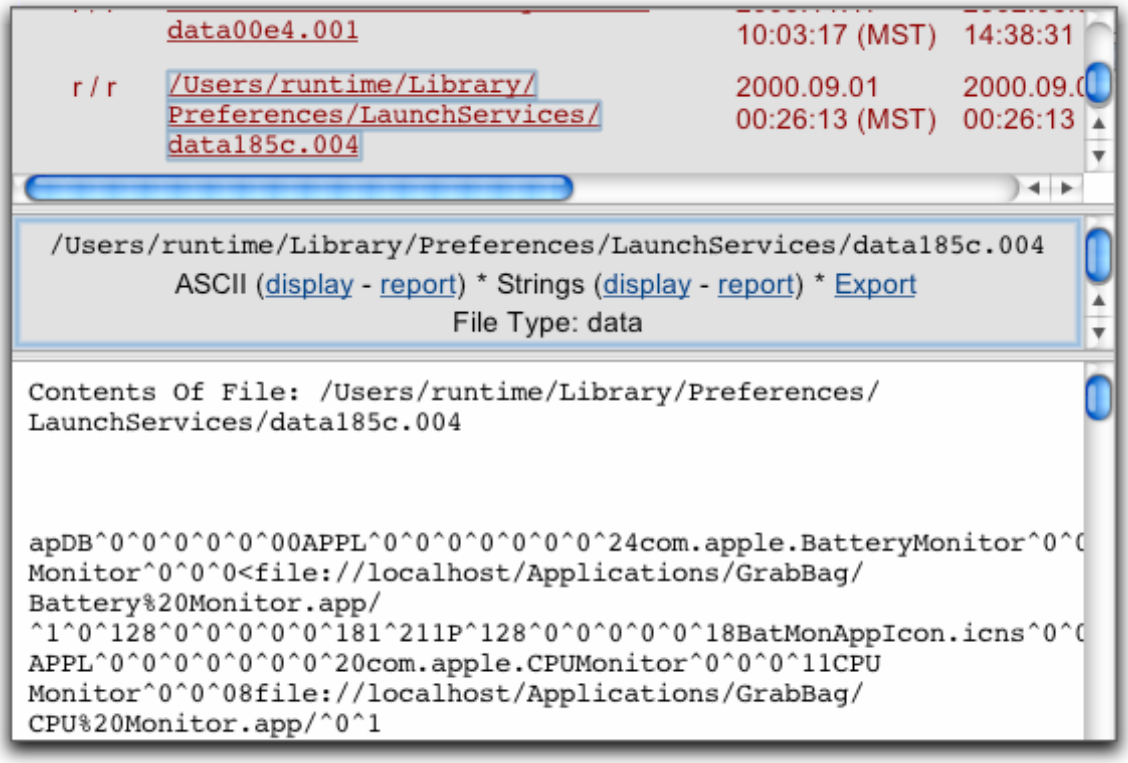
The entire list of known deleted files partially shown on the right above is given below with the modified time, accessed time, UID, GUI and inode number removed. The changed time represents the most likely time that the files were deleted.

Type dir / in	Name	Changed	Size
r / r	/private/var/db/netinfo/local.nidb/TempIndex	2002.04.29 08:01:05 (MST)	2656

r / r	/private/var/db/netinfo/local.nidb/Store.928	2002.04.29 08:01:05 (MST)	2656
r / r	/private/var/db/netinfo/local.nidb/Store.960	2002.04.29 08:01:05 (MST)	4
r / r	/private/var/log/wtmp.0	2002.04.18 12:39:55 (MST)	1008
r / r	/private/var/log/system.log.0	2002.04.29 02:21:25 (MST)	0
r / -	/private/var/mail/_JJBQ.NameDrop	2002.04.29 02:21:01 (MST)	0
r / -	/private/var/root/Library/Preferences/ByHost/cf#00149.000	2001.05.18 06:38:21 (MST)	0
r / -	/private/var/root/Library/Preferences/ByHost/com.apple.finder.NameDropper.plist	2001.05.18 06:38:21 (MST)	0
r / -	/private/var/root/Library/Preferences/Apple Help Prefs/tmp/HelpSearchResultsb07	0000.00.00 00:00:00 (GMT)	0
r / r	/private/var/root/Library/Preferences/Apple Help Prefs/tmp/._HelpSearchResultsb07	2000.12.07 15:16:03 (MST)	2463
r / -	/private/var/root/Library/Preferences/Apple Help Prefs/TestForLockedVolume	2002.04.29 02:15:18 (MST)	0
r / -	/private/var/root/Library/Preferences/Apple Help Prefs/._TestForLockedVolume	2002.04.29 02:15:18 (MST)	0
r / r	/private/var/root/Library/Preferences/Explorer/._64666A6E	2000.12.20 12:27:31 (MST)	82
r / r	/private/var/root/Library/Preferences/Explorer/64666A6E	2000.12.20 12:27:11 (MST)	3614
r / r	/private/var/root/Library/Preferences/Sherlock Prefs/._64666A6E	2000.11.27 12:23:19 (MST)	520
r / r	/private/var/root/Library/Preferences/Sherlock Prefs/64666A6E	2000.11.27 12:23:19 (MST)	1152
r / r	/private/var/root/Library/Fonts/data011a.001	2000.11.22 08:43:56 (MST)	168
r / -	/private/var/run/sshd.pid	2002.04.29 08:01:05 (MST)	0
s / -	/private/var/run/ndc	2002.04.29 08:01:05 (MST)	0
r / r	/private/var/run/ntp.drift.TEMP	2002.04.29 07:39:25 (MST)	9
r / -	/private/var/spool/mqueue/tfg3T9L1O04679	2002.04.29 08:01:06 (MST)	0
r / -	/private/var/tmp/locate.list.3982	2002.04.27 03:34:05 (MST)	0
r / -	/private/var/tmp/console.log	2002.04.27 03:34:05 (MST)	0
r / -	/private/var/tmp/sort0151400027	2002.04.13 03:33:55 (MST)	0
r / -	/private/var/tmp/sort0275500027	2000.12.16 04:33:55 (MST)	0
r / -	/private/var/named/named.pid	2002.04.29 08:01:05 (MST)	0
r / r	/private/var/named/xxx.named.rev.log	2002.04.29 08:01:05 (MST)	13491
r / -	/private/etc/nologin	2002.04.18 12:38:46 (MST)	0
r / -	/private/etc/nologin	2002.04.18 12:38:46 (MST)	0
r / r	/Library/Preferences/cf#00037.000	2002.04.18 12:38:37 (MST)	1311
r / r	/System/Library/Fonts/data00af.001	2002.04.18 12:39:48 (MST)	9528
r / r	/usr/share/man/whatis.db.tmp	2002.04.27 03:34:55 (MST)	1E+5
r / -	/Users/fletcher/Library/Preferences/LaunchServices/data0139.004	2000.12.11 09:17:31 (MST)	0
r / r	/Users/fletcher/Library/Preferences/Explorer/data0139.005	2000.11.17 08:49:12 (MST)	15948
r / r	/Users/fletcher/Library/Preferences/ByHost/cf#0012a.000	2002.04.27 03:34:55 (MST)	30
r / r	/Users/fletcher/Library/Preferences/ByHost/com.apple.finder.NameDropper.plist	2002.04.27 03:34:55 (MST)	30
r / r	/Users/fletcher/Library/Preferences/cf#00129.003	2001.05.01 14:43:47 (MST)	7739
r / -	/Users/fletcher/Library/Preferences/cf#005a5.003	2001.05.02 02:15:14 (MST)	0
r / r	/Users/fletcher/Library/Fonts/data00e4.001	2000.11.17 10:03:20 (MST)	168
r / r	/Users/runtime/Library/Preferences/LaunchServices/data185c.004	2000.12.11 09:32:18 (MST)	13847
d / -	/Developer/Documentation/Carbon/graphics/ColorSync Manager/ColorSync_Manager/Enumerations	0000.00.00 00:00:00 (GMT)	0

r / -	/Developer/Documentation/Carbon/runtime/CodeFragmentManager/Code_Fragment_Manager/APIIndex.html_bak	0000.00.00 00:00:00 (GMT)	0

Each file that was listed as having some recoverability (i.e., non-zero size) was further investigated using Autopsy. The screenshot below shows the data for the data185c.004 file located in the runtime home directory. This particular file appears to be a document listing the file type and application assignments and their associated icons.



A similar analysis of the other recoverable files did not turn up anything of interest. An extended search of deleted files that were already allocated through use of the corresponding inode also failed to provide any additional information.

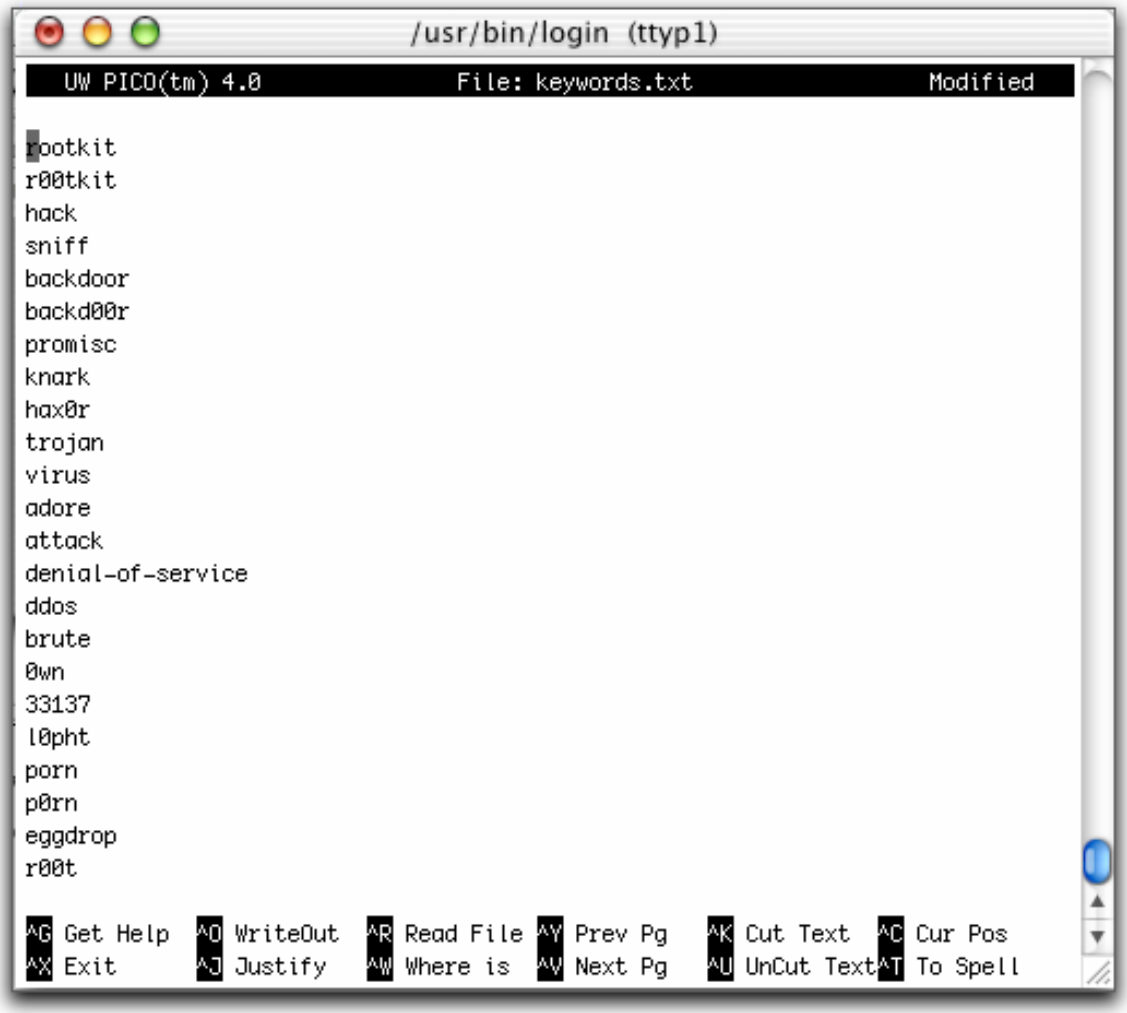
String Search Analysis

While in single-user mode to reduce potential GUI interactions, strings for each disk image were extracted using the Mac OS X built-in BSD command *strings* as follows:

```
localhost# disktool -c 0
localhost# hdiutil mount /Users/remiller/Desktop/GCFA/Part\ I/disk1.img
localhost# strings - /Volumes/Home > /Users/remiller/Desktop/strings1.txt

localhost# hdiutil mount /Users/remiller/morgue/disk2.img
localhost# strings - /Volumes/Mac\ OS\ X/ > /Users/remiller/Desktop/strings2.txt
```

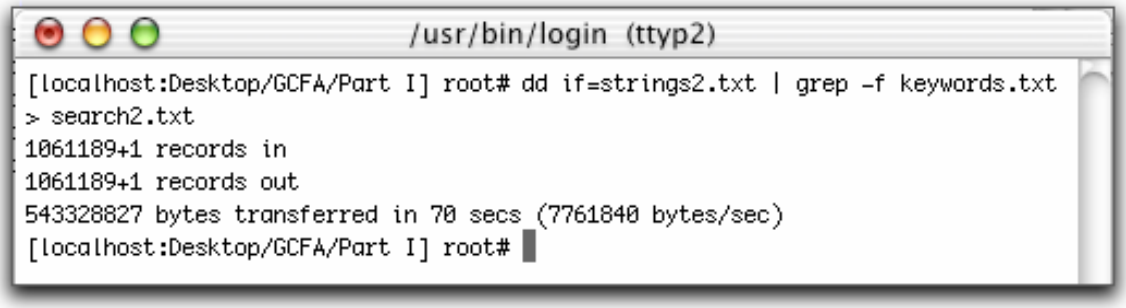
The strings1.txt file was massive (i.e., 1.43 GB.) Extracting this much information from a seemingly empty disk that had no recoverable files would have been cause for concern if it wasn't for the information previously pulled out of the Desktop DB file that clearly suggests that the disk was previously used on another system running IRIX.

A list of the keywords shown below was created and saved. These are general keywords that could be associated with a compromised system (i.e., rootkits, sniffers, and backdoors) or with potential misuse of the system based on policy (i.e., pornography, virus creation, and network attacks.) If a more specific incident had been suspected (e.g., child pornography) then a more specific list of keywords would have been used.



This file was used to extract only these keywords from the strings files created for disk2.img since the information obtained from disk1.img would more than likely provide

evidence that was not from the target system. The command given and the associated output is shown below:



Inspection of the search2.txt file provides the following insights:

- Virtually all of the "hack" references are comments in the code of default applications or example files as shown below:

```
instead of setting up a fresh yyin.  A bit of a hack ...
                        /* various hacks, don't look :) */
    struct internal_state {int dummy;}; /* hack for buggy compilers */
```

- Most of the "promiscuous" references appear to relate to the operating system's networking code as shown below.

```
K:networkcomm:OpenTransport:Open_Transport:DataTypes:dl_promiscon_req_t.ht
ml
```

- "Virus" references appear to come from a "virus dictionary" as shown below:

```
The INIT 1984 virus, which is programmed to trigger on every Friday the
13th, is a dangerous Macintosh virus
```

- References to "attack" or "denial of service" appear simply as guidance in the code as shown below:

```
# NOTE: There is a potential for a denial of service attack if this is
set.
```

Overall, the information contained in this file appeared to be from portions of legitimate applications and files. There was no sign of any malware, sniffer programs, pornography, rootkits or the keywords associated with any of these categories.

Conclusions

The evidence obtained from the target system clearly suggests that its primary usage was as a DNS server. The most substantial change to the startup sequence of the system was the addition of the Bind service. A runtime account was setup for named and all of the data and configuration files were placed on the system in support this role. The last file to be altered on the system was the named.log file, which itself indicated that the system was servicing domain name requests for the organization just before it was shutdown on April 29, 2002.

There appear to have been only two accounts setup on the system that were associated with real users. Both users appeared to have been unfamiliar with the conventions of Mac OS X and probably had a strong traditional UNIX background based on their command history and the types of configuration changes that were made to the system. The setup of the second disk (as /Home) and the fact that it appeared to have come from an SGI workstation also reinforces this conclusion.

It appears that ssh was used to log into the system remotely for the most part. The root account was the account of choice in the later months of usage that were examined. Usage of the GUI portion of Mac OS X appeared to be limited and only relegated to the applications that shipped with the operating system. Only the developer tools and possibly Xfree86 were ever installed. A shareware firewall utility was found but apparently not installed. A simple FORTRAN program was found on the system but not compiled. There were no deleted files recovered that were of interest. There was no indication that the system was ever compromised or used for illicit purposes.

Finally, regarding the small needles found in the case, a reinvestigation of the room where the computer was seized revealed a small fern in the window that appeared to be dying. The rust colored water ring found on the case matched the planter that the fern was in. The SA confirmed that this plant spent some time on top of the case of the target system while it was in his possession.

Regarding the secondary reason for the evaluation of this particular system, the usage of Mac OS X both as a target system as well as an analysis system was essentially successful. Forensic evaluation of a Mac OS X system is very similar to the analysis of a traditional UNIX system with the addition of some analysis aspects more closely related to Windows. The largest hurdle to analyzing a Mac OS X system was the lack of support for the HFS and HFS+ file systems in open source tools (this is critical considering that HFS+ is the default file system for Mac OS X.) Because of this, commercial utilities were needed to perform parts of the forensic analysis. This patchwork procedure to retrieve data from an HFS+ system worked, but is certainly not ideal.

Using Mac OS X as an imaging platform actually worked very well. Since Apple hardware supports most current drive interface technologies (IDE, SCSI, USB and FireWire) out of the box, the platform lends itself to this particular task very well. While using Mac OS X as the analysis platform worked, it presented a few unique challenges.

Not all of the command line tools that ship with Mac OS X perform exactly the same function as their UNIX/Linux counterparts. In addition, the inability to lock out access time changes to an image could cause problems. Other than these minor issues, the forensic techniques and tools used with UNIX and Windows worked well under Mac OS X.

References

Apple Computer, Inc. "Apple – Mac OS X." September 15, 2000. URL: http://www.apple.com/macosx.

Apple Computer, Inc. "Apple – Mac OS X – Technologies – Darwin." September 15, 2000. URL: http://www.apple.com/macosx/technologies/darwin.html.

Apple Computer, Inc. "Darwin – Open Source." September 15, 2000. URL: http://developer.apple.com/darwin/.

Apple Computer, Inc. "Developer Mac OS 9." September 15, 2000. URL: http://developer.apple.com/macos/macos9.html.

Apple Computer, Inc. "Mac OS: Rebuilding Desktop File and Icon Recovery." AppleCare Document, Article ID: 10182. May 12, 1992.

Apple Computer, Inc. "Mac OS X Developer Tools Update." September 15, 2000. URL: http://developer.apple.com/tools/macosxtools.html.

Apple Computer, Inc. "Mac OS X Server: What Is NetInfo?" AppleCare Knowledge Base, Article ID: 60038. February 2, 1999.

@stake, Inc. "@stake Research Labs – Tools." September 15, 2000. URL: http://www.atstake.com/research/tools/index.html.

Hill, Brian. "BrickHouse." June 4, 2001. URL: http://personalpages.tds.net/~brian_hill/brickhouse.html.

Network Associates. "McAfee – Products – Virex." September 15, 2000. URL: http://www.mcafeeb2b.com/products/virex/.

The Omni Group. "The Omni Group – Applications – OmniWeb." September 15, 2000. URL: http://www.omnigroup.com/applications/omniweb/.

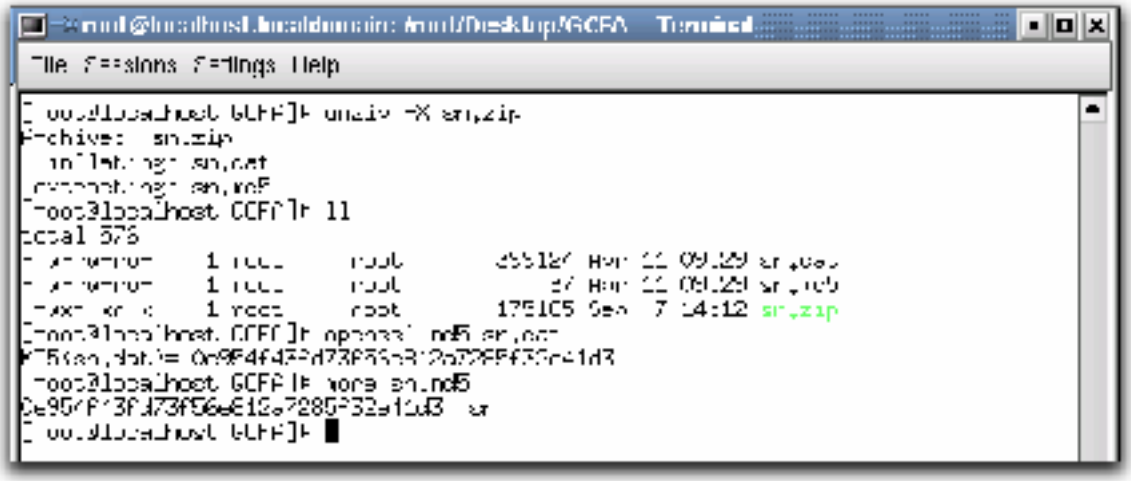
Prosoft Engineering, Inc. "Prosoft Data Rescue." September 15, 2000. URL: http://www.prosofteng.com/index.php?datarescue.

Symantec Corporation. “Norton Utilities for Macintosh.” September 15, 2000. URL: http://www.symantec.com/nu/nu_mac/.

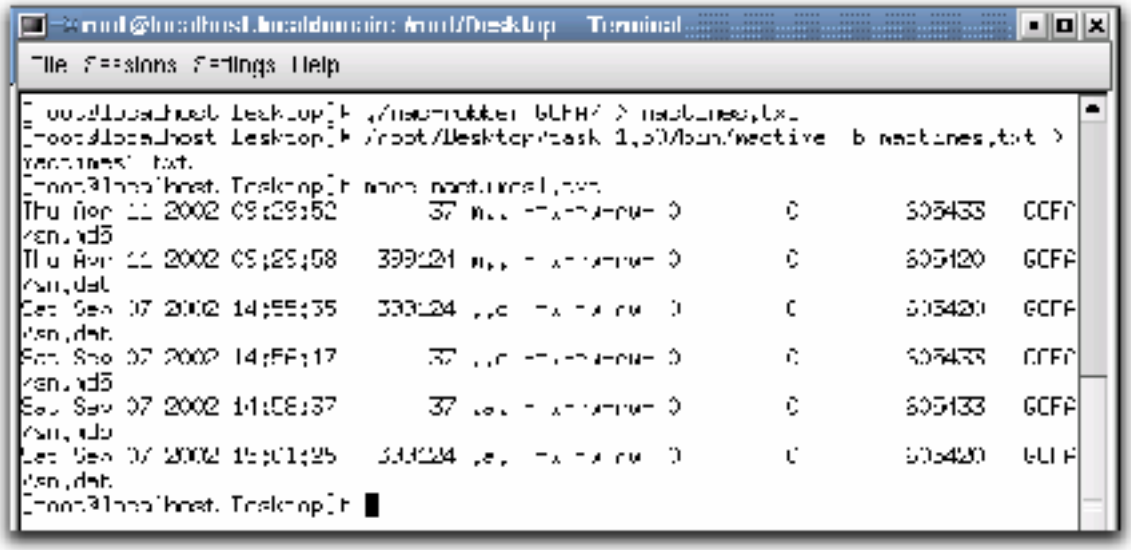
Part 2 – Analysis of an Unknown Binary

Binary Details

The unknown binary was downloaded from the GIAC site in a file called sn.zip. The file was unzipped with the option to retain the original UID and GID for the files. The zipped archive contained two files, one called sn.dat and the other called sn.md5. The MD5 hash of the sn.dat was calculated and compared to the value contained in the sn.md5 file. The extraction of the two files as well as the MD5 hash calculation is shown below. Note that the calculated hash value equals the value given in the sn.md5 file.



The size of the binary was listed as 399124 bytes and the original owner was root and the original group was root. A MACtime analysis of the binary using *mac-robber* and *mactime* from TASK is shown below. As indicated, the binary was last modified on April 11, 2002 at 09:29:52 and was subsequently marked as changed and accessed when it was unzipped for analysis. The last modification time would be the local time zone of the originating system, which is unknown.



Next, *strings* was run on the binary and the output was directed into a text file as shown below.

The resulting strings.txt file was examined for keywords related to the program. Several interesting strings were found in the binary. The first set is shown below:

```
\*       The END            */
priv 1.0
ADMsniff %s <device> [HEADERSIZE] [DEBUG]
ex    : admsniff le0
 ..ooOO The ADM Crew OOoo..
cant open pcap device :<
init_pcap : Unknown device type!
ADMsniff %s  in libpcap we trust !
credits: ADM, mel , ^pretty^ for the mail she sent me
The_l0gz
```

These strings represented some potentially important clues as to what the binary is and where it came from. However, no conclusion could be drawn at the time regarding the binary or its origins.

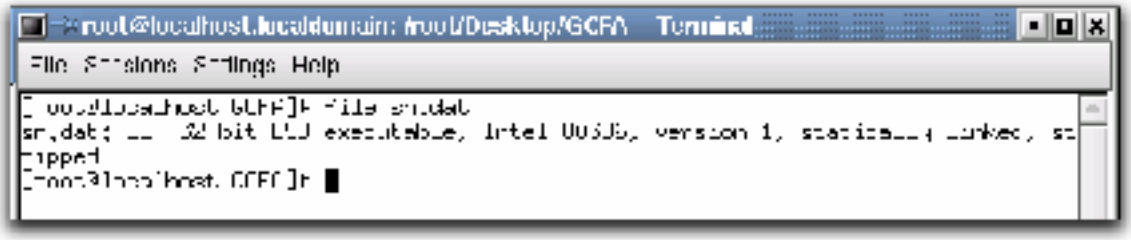
The rest of the strings that were extracted appeared to deal with normal programming and compiling functions. However, in the middle of this type of data was the following personal information:

```
1997-12-20
+45 3325-6543
+45 3122-6543
keld@dkuug.dk
Keld Simonsen
ISO/IEC 14652 i18n FDCC-set
C/o Keld Simonsen, Skt. Jorgens Alle 8, DK-1615 Kobenhavn V
ISO/IEC JTC1/SC22/WG20 - internationalization
```

This information was also potentially valuable and was investigated further. Again, no conclusions were drawn from this information.

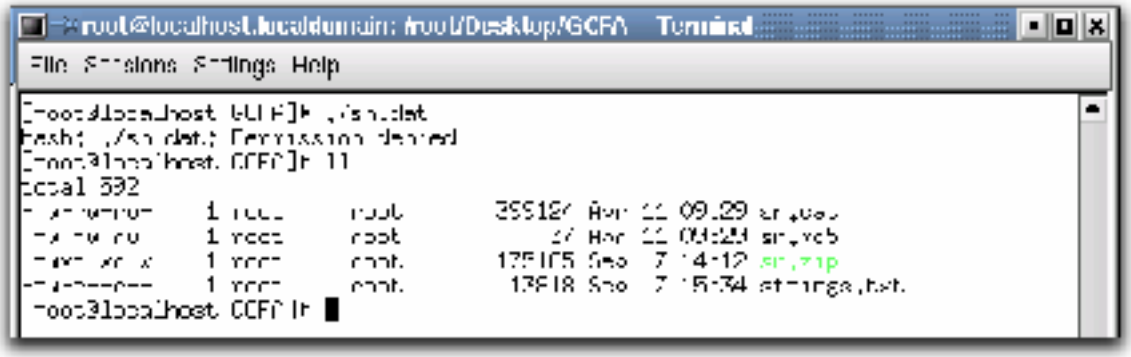
Program Description and Forensic Details

Now that the binary was assumed to have been delivered unaltered, as verified with the MD5 hash inspection, the first step was to run the *file* command to determine what kind of file it was. The results of this command are shown below:

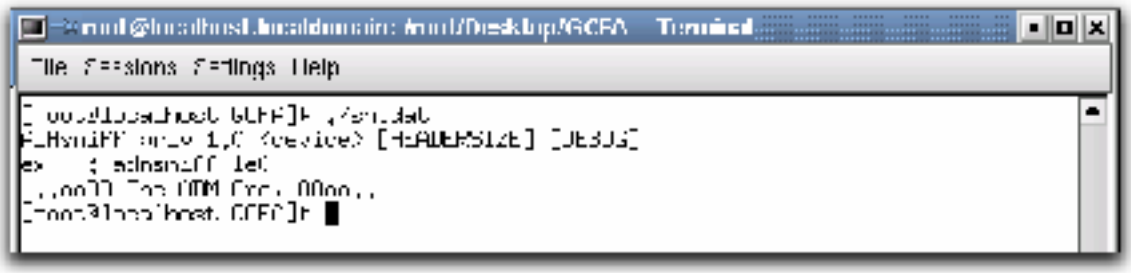


As indicated, the file appeared to be an ELF executable that was statically linked so all functions were included in the binary. In addition, it has been stripped of debug symbols. The fact that the binary was statically linked and stripped made the analysis more complicated.

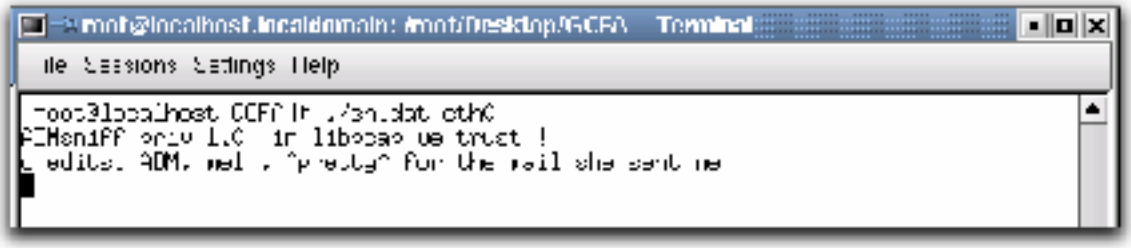
The next step chosen was to go ahead and execute the program. The first attempt to execute the unknown binary sn.dat is shown below.



Clearly, execute permissions were not set originally on the binary. After the execute permission was enabled, the binary was executed as shown below.



Clearly, the first set of text that was pulled out using *strings* appears when the program executes. Following the example given and rerunning the program with the appropriate Ethernet interface specified (in this case eth0) produced the following output. The binary continued to execute as shown above until the break signal was sent.

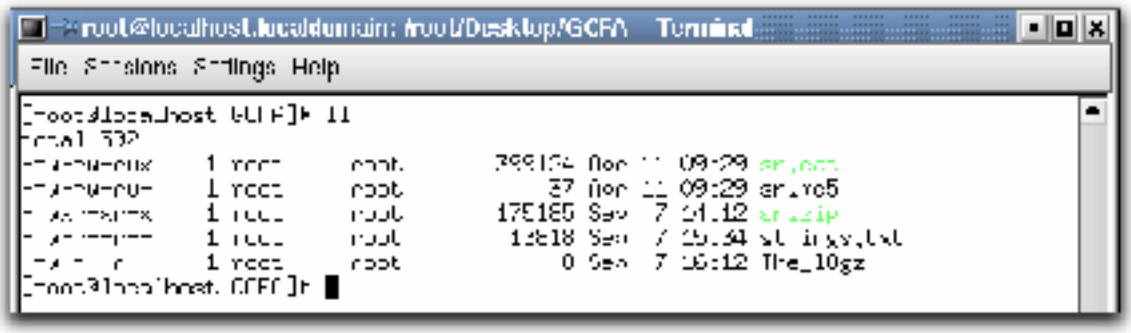


As a test, an incorrect interface was specified, and the following output occurred.

In both cases, there were references to libpcap, which is a packet capture library used in sniffer programs (TCPDUMP, p.1). This information and the apparent name of the program (ADMsniff) strongly suggested that this binary was a packet sniffer.

Interestingly, after the program had executed, a new file appeared in the directory with the binary as shown below. "The_l0gz" file appeared to be solid evidence pointing towards the execution of the binary after the fact (i.e., a forensic footprint.) Since the binary was statically linked, it should not reference any system files when it executes.

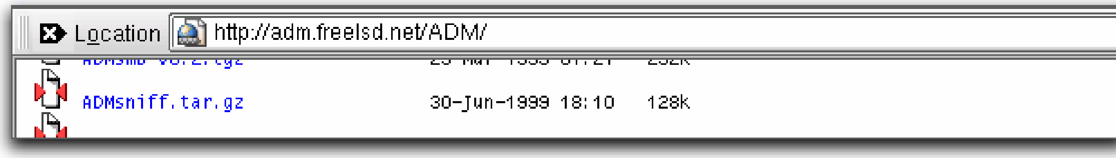
This fact and further analysis of the binary's interaction with the system were investigated further in the next section.



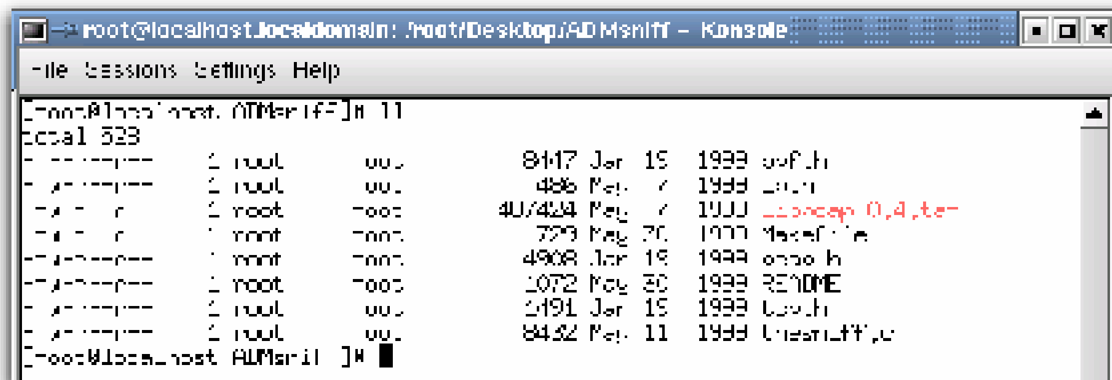
Program Identification

At this point, there was enough evidence to warrant trying to find the source code for this binary. Based on running the binary as well as the information derived from *strings*, it appeared that the best keyword to search for initially was "ADMsniff." Running this search in Google produced the following hits:

The first three hits clearly related to ADMsniff version 0.8. The header and strings extracted from the binary indicated that it was probably version 1.0, so this isn't the correct version of the code. The fourth hit looked promising from the simple fact that it appeared to be the ADM Crew's "official" web site since it contained several ADM programs. Clicking on this link brought up the following directory listing, which contained an archive of the program ADMsniff.



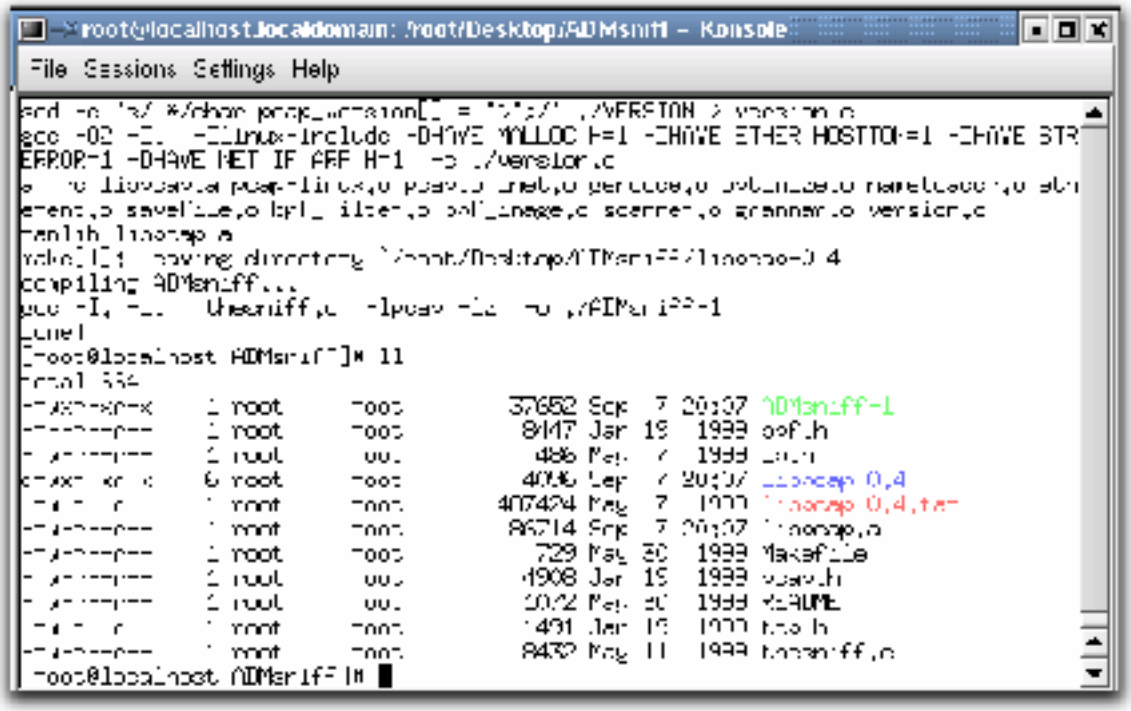
This archive was downloaded from the site. The archive was then extracted and untarred. The items produced from this process are shown below.



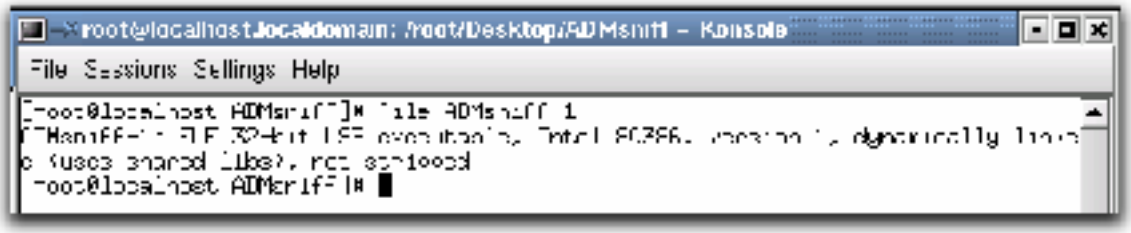
The README file indicates that ADMsniff was indeed a packet sniffer, as shown below:

```
                    ___        ______     _      _
                  /     \     |  _   \   |  \   /  |
                 |  / \  |    |  | \  |  |   \_/   |
                 | |___| |    |  |_ / |  |   \_/   |
..oO  THE        |  ---  |    |       /  |  |   |  |        CreW Oo..
                 '''   '''    '''''''    ''''   ''''
                              presents

                          ADMsniff v0.1b



                   by:    antilove
                          zlib support by plaguez


        ADMsniff is a libpcap-based sniffer, designed to be
        portable and powerful.
```

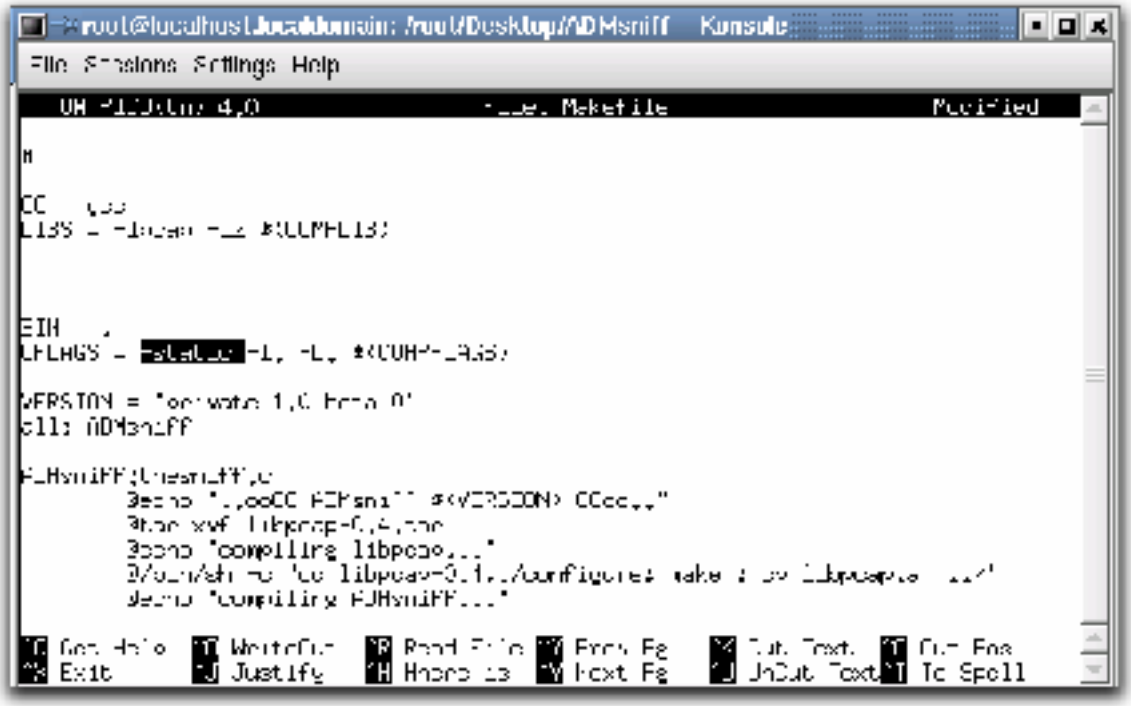
The presence of the libpcap code also appeared to back this up. After reviewing the README for basic instructions, the *make* command was invoked. The resulting binary was created as shown below:



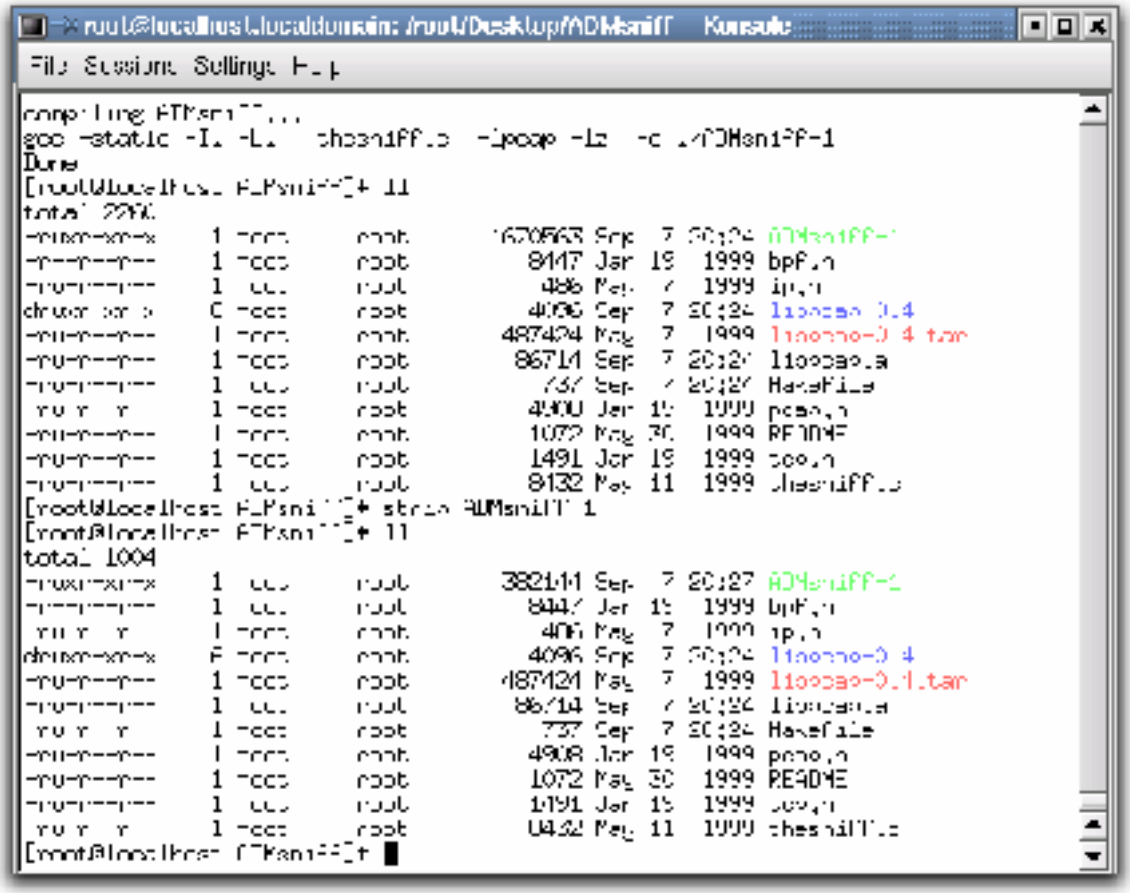
The size of the resulting binary was approximately an order of magnitude too small compared to the size of the unknown binary. Checking the ADMsniff-1 binary with the *file* command revealed the probable cause for this as shown below. ADMsniff-1 was a dynamically linked binary that did not have the symbols stripped as was the case with the unknown binary.



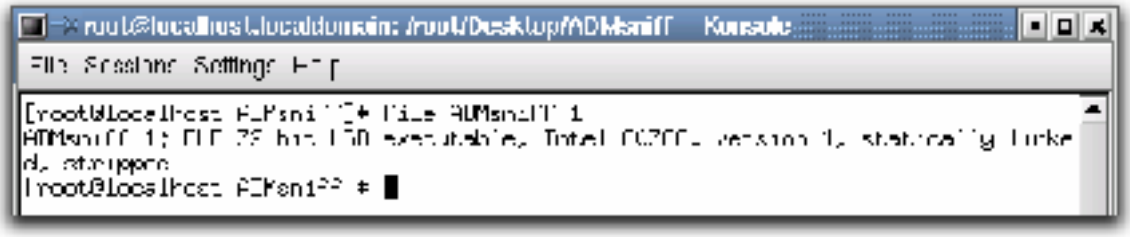
Clearly, the Makefile needed to be adjusted so that the source code would compile statically. In addition, the symbols needed to be stripped afterwards. To accomplish the first adjustment, the Makefile for ADMsniff had the "–static" flag added as shown below:



The make command was run again with this adjustment. The resulting binary was then stripped of symbols using the *strip* command. This process is shown below as well as the resulting binary.

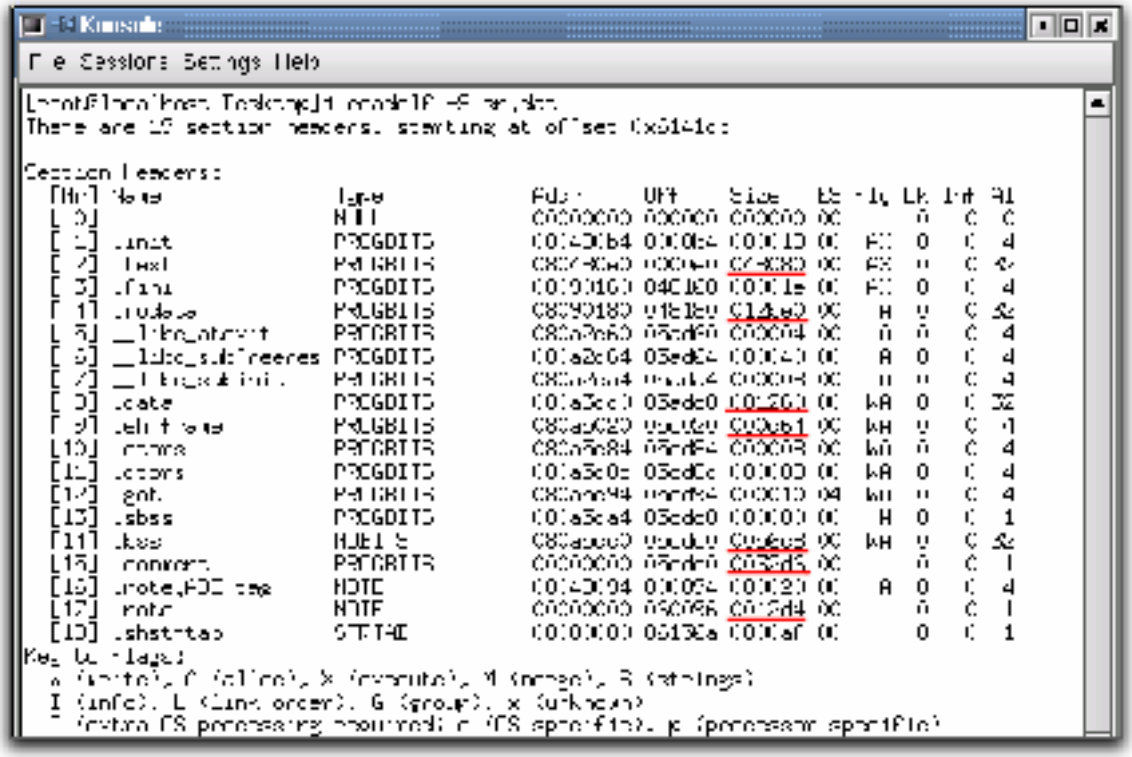


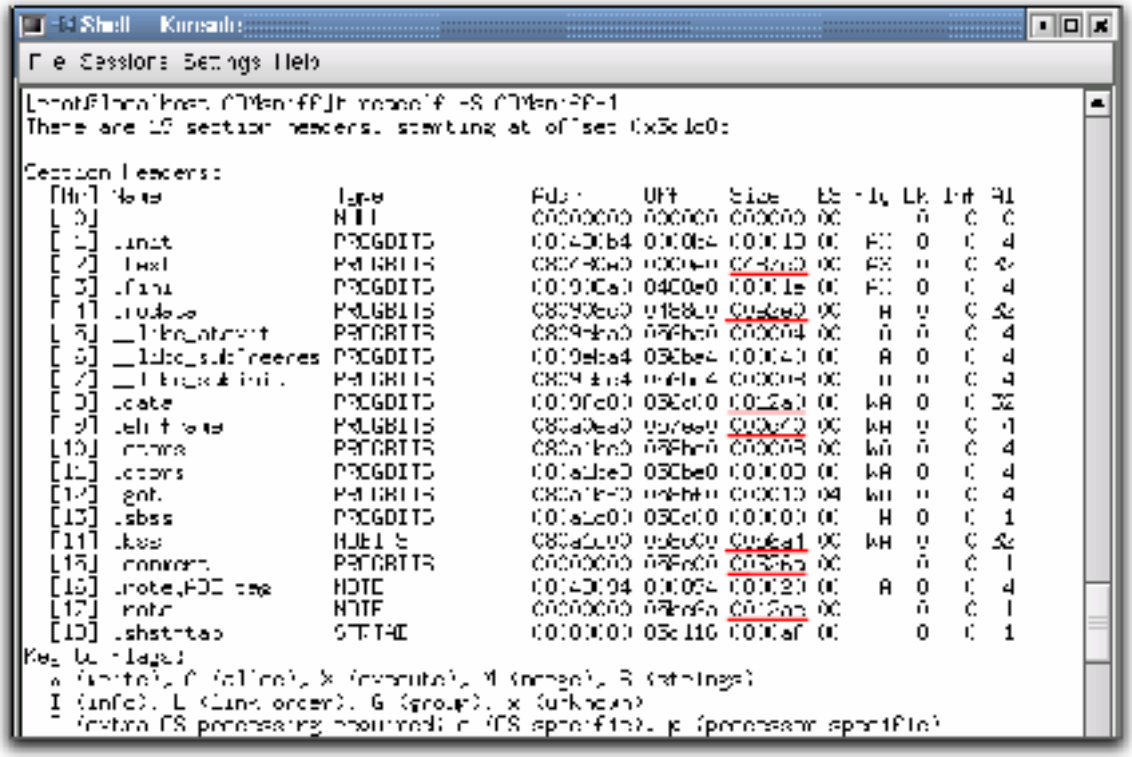
At this point, the binary ADMsniff-1 was statically linked and stripped as verified below using the *file* command. Unfortunately, this binary was not the exact same size as the unknown binary and therefore would not have matching MD5 hashes.



In an effort to explain the size difference of 16,980 bytes, several avenues were explored. The Makefile had an option for compressing the log file created by the sniffer. This was enabled and the code was recompiled and stripped. The resulting binary was much larger than the unknown binary (428220 bytes). Therefore, this option was not responsible for the size difference.

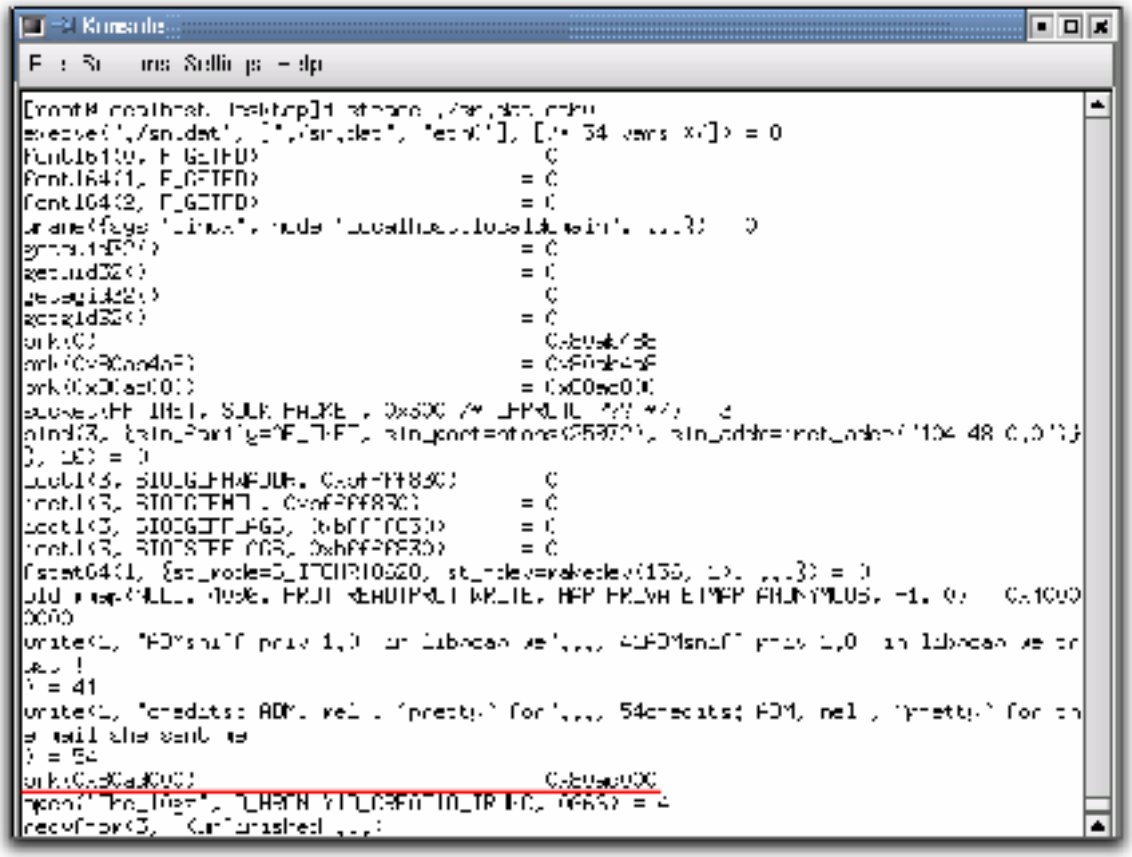
In a further effort to determine the differences between the two binaries, the *readelf* utility was run with the –S option on the unknown binary (sn.dat) and the known binary (ADMsniff-1.) This option displays the section header information. The output of running this command on each binary is shown in the next two figures below. Differences in the sizes of each section header between the two binaries were underlined in red.

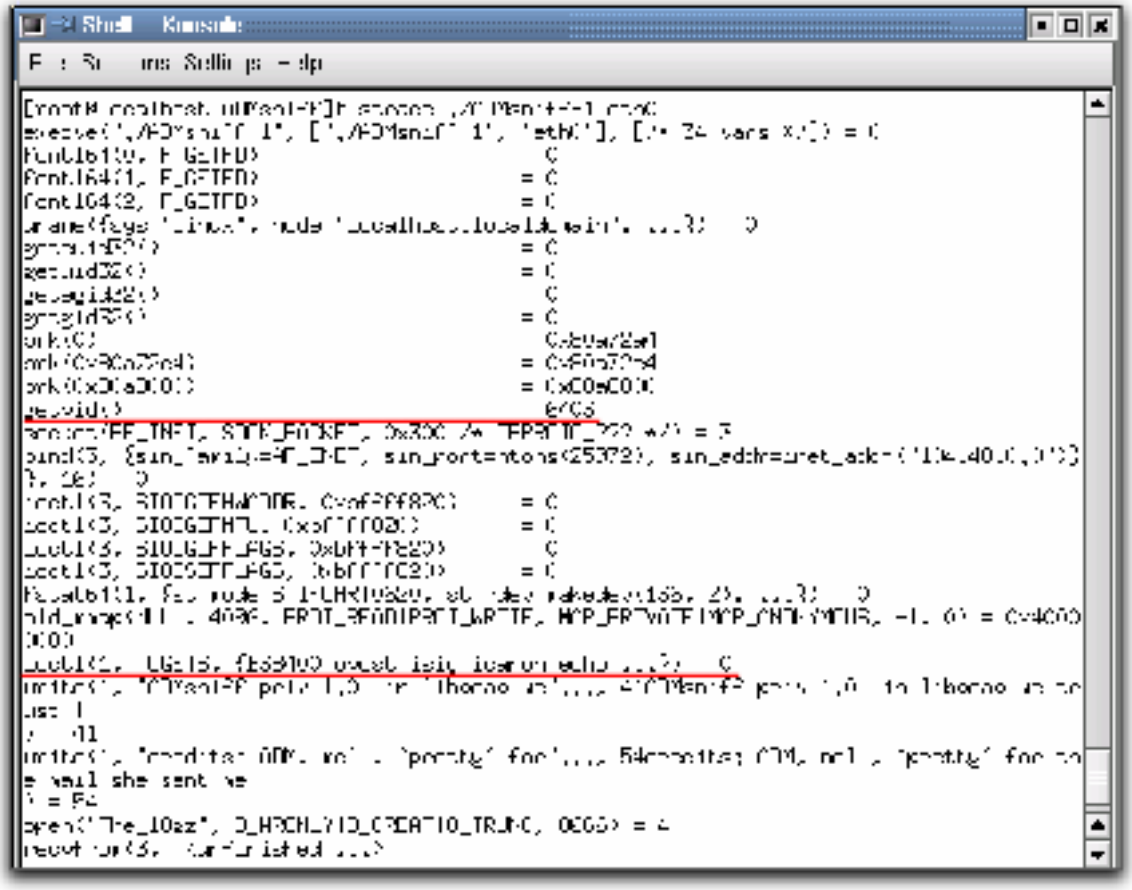




As shown, the section headers called .text, .rodata, .data, .eh_frame, .bss, .comment and .note are different sizes between the two binaries. Of these section headers, only .text was listed as execute. Further investigation of these differences exceeded the capabilities of the author in terms of programming knowledge and technical skill.

Next, *strace* was run on each binary to determine how the two binaries interacted with the system as well as to determine if there were any differences. The output is shown in the next two figures below with the differences underlined in red.





The unknown binary sn.dat sent a *brk* that ADMsniff-1 did not. In addition, ADMsniff-1 queried the pid and made a *ioctl* call that the unknown binary did not. Otherwise, the two programs performed the same system calls in the same order. This output also showed that neither binary altered the file system other than to open the file "The_logz", which has already been noted. Also, neither binary appeared to reference, manipulate or use any system files. This was expected since they are both statically linked.

Other utilities and options (including *gdb*) were used in an effort to uncover the differences between the two binaries. The source code of ADMsniff was also examined to determine any reason for the differences. In addition, the *strings* command was run on the ADMsniff binary and the output was processed through *diff* with the strings extracted from the unknown binary. Each of these avenues was of no help primarily due to the lack of c programming/debugging and binary decompilation skills on the part of the author.

Finally, the second set of information pulled out of the unknown binary using *strings* ended in a dead-end. Research of the information showed it to be related to an ISO standard on language and culture conventions used in programming (ISO/IEC, p.102).

Strings associated with this standard appeared in both the unknown binary as well as the compiled ADMsniff binary.

Legal Implications

Assuming that the unknown binary is ADMsniff (which is a sniffer application) and nothing more, its execution by an outsider or an insider without proper authorization would then clearly constitute an illegal wiretap. This would be in violation of the Federal Wiretap Act (18 U.S.C. §2510) and the Pen Registers and Trap and Trace Statute (18 U.S.C. §3121) in the United States (Salgado, p.1-29). Violation of the Wiretap Act can be penalized through a fine and or imprisonment of not more than five years (Privacy, p.1). Violation of the Pen, Trap and Trace Statute carry a fine and or imprisonment of not more than one year (Privacy, p.1).

Investigation of the unknown binary and of the alleged source code clearly showed that proof of the binary's execution comes in the form of the "The_l0gz" file. This file is created in the same directory as the binary when it is executed. If both the binary and this file were found on a system, then it would represent clear evidence that the binary was executed and that the statutes noted above could have been violated.

Interview Questions

If the person responsible for installing and executing this binary (still assumed to be ADMsniff and nothing more) could be interviewed, then the following questions could be used based in part on the evidence uncovered about the binary.

- I know that you sometimes use the root account to get things straightened out while the SAs are busy. There isn't anything wrong with that; you were just trying to help out, right? I noticed that you were on the system on April 11, 2002, can you tell me what you were working on then?

- Management is on my back to find out what this is all about. If I don't come up with an explanation, they are going to bring in the authorities and press for a conviction. At that point it is out of my hands and you know how the Feds treat these matters now days, so can you please explain to me what happened here and hopefully no one will get fired?

- I noticed that you like to play with hacker tools. I also like to know what I'm up against out there and using their tools is the best way to find out, right? I see that you were playing around with a sniffer. Can you tell me how it worked out? Were you able to find anything out about our network?

- We found out what you were doing on the system. Using our computers to hack into those other companies systems was bad enough, but why did you go after that

government computer? You're looking at spending a very long time in prison for that one alone. What were you thinking?

- Management thinks that you have been messing with the system. I think that they are going to turn it over to security soon. If you tell me exactly what you did, I can try to explain to them that you were just testing out some tools on the system and that it isn't a problem. Will you let me help you out?

- I found your tools on the system and was trying to figure out what they do to see if I could use them elsewhere. Do you mind helping me out by showing me what they can do?

Conclusions

The unknown binary sn.dat most likely is some form of the packet sniffing program ADMsniff. The program is used to capture and log network traffic. It was last used on April 11, 2002 at 09:29:52 in an unknown time zone. An exact identification of the binary was not possible due to the lack of programming and decompliation skills on the part of the author. It is possible that the unknown binary performs other actions in addition to those of the unaltered ADMsniff program, but this cannot be confirmed or denied.

A forensic footprint of this binary in the form of a log file was determined. This footprint could be used as evidence that the unknown binary was executed. Since the binary is most likely a sniffing program, its unauthorized use would violate two United States statutes and this forensic footprint could be used as evidence to that effect in any legal proceedings.

Additional Information

For additional information on the following topics, please see the associated URL.

- Sniffers FAQ – http://www.faqs.org/faqs/computer-security/sniffers/.
- Packet capture with libpcap – http://www.cet.nau.edu/~mc8/Socket/Tutorials/section1.html.
- Packet sniffing programs – http://www.cotse.com/tools/sniffers.htm.
- Laws governing electronic communication – http://www.gcwf.com/firm/groups/privacy/electronic_comm.html.
- Wiretap FAQ – http://www.privacyfoundation.org/resources/wiretap.asp.

References

ISO/IEC. "Information Technology – Specification Method for Cultural Conventions."

June 11, 1999. URL: http://www.cl.cam.ac.uk/~mgk25/volatile/ISO-14652.pdf.

Privacy Foundation. "Wiretap FAQ: A Guide to Federal Law on Electronic Surveillance." September 15, 2002. URL: http://www.privacyfoundation.org/resources/wiretap.asp.

Salgado, Richard P. "Forensics and Incident Response: The Law Enforcement Perspective." 8.4: Systems Forensics, Investigation, and Response. Orlando: SANS Institute, 2002.

TCPDUMP. "TCPDUMP Public Repository." September 15, 2002. URL: http://www.tcpdump.org.

Part 3 – Legal Issues of Incident Handling

The Wiretap Statute

The Federal Wiretap Statute (18 U.S.C. §2510) expressly makes it illegal in the United States to intercept the content of a communication that is directed over wire, oral or electronic means (USDOJ, p.1). There are a number of important exceptions to this statute, the first of which is the provider exception as stated below:

> 2 (a) It shall not be unlawful under this chapter for an operator of a switchboard, or on officer, employee, or agent of a provider of wire or electronic communication service, whose facilities are used in the transmission of a wire or electronic communication, to intercept, disclose, or use that communication in the normal course of his employment while engaged in any activity which is a necessary incident to the rendition of his service or to the protection of the rights or property of the provider of that service, except that a provider of wire communication service to the public shall not utilize service observing or random monitoring except for mechanical or service quality control checks. (USDOJ, p.1)

This exception has been applied to systems administrators and clarified by the case of *United States v. Mullins*, 992 F.2d 1472, 1478 (9th Cir. 1993) (Salgado, 1-32). This establishes the ability of systems administrators to track hackers within their own network by intercepting communications for the expressed overall purpose of protecting the network.

This exception is very specific in that it does not allow for the interception of any and all communication as noted by *United States v. McLaren*, (957 F. Supp. 215, 219 (M.D. Fla. 1997) (Salgado, 1-32). To abide by this provision, systems administrators must limit their interceptions only to traffic specifically tied to the investigation of an incident, which has been defined legally as a "substantial nexus." To collect other traffic or to randomly sample traffic could be in violation of the Wiretap Statute and consequently punishable by a fine and or a maximum of five years of imprisonment (Privacy, p.1).

Another important exception to the Wiretap Statute is stated below:

> 2 (d) It shall not be unlawful under this chapter for a person not acting under color of law to intercept a wire, oral, or electronic communication where such person is a party to the communication or where one of the parties to the communication has given prior consent to such interception unless such communication is intercepted for the purpose of committing any criminal or tortious act in violation of the Constitution or laws of the United States or of any State. (USDOJ, p.1)

This is a prior consent exception. One example of obtaining prior consent would be the use of banners on a system. Bannering a system would provide for notification to the user that further usage of the system would indicate consent to monitoring of any and all communications. Once past the banner, the user of the system essentially has given prior consent to monitoring as defined in the exception above.

The use of banners represents another way for systems administrators to monitor traffic on their networks that would not necessarily be covered by the provider exception noted first. However, it is only in effect if one party to the communication sees the banner and accepts it. When individuals access systems and networks through means that do not present a banner, then this exception is not in effect. At this point there must then exist a "substantial nexus" in order for the systems administrator to be monitoring the communications legally.

Conclusions

Systems administrators can legally monitor communications on their network in two specific situations. The first involves the investigation of an intruder where the systems administrator is attempting to prevent further damage from the intruder's activities. The communications that are monitored must be directly related to the investigation and meet the legal requirement of a "substantial nexus." The second situation involves the use of banners whereby one party to the communication gives prior consent to the monitoring. Banners have to be acknowledged in some fashion by one party to the communication in order for this situation to exist.

Systems administrators who monitor communications under conditions that do not fall into either of the above situations may be performing an illegal wiretap. Conviction of illegally monitoring a communication carries with it a potential fine and or imprisonment of up to five years.

References

Privacy Foundation. "Wiretap FAQ: A Guide to Federal Law on Electronic Surveillance." September 15, 2002. URL: http://www.privacyfoundation.org/resources/wiretap.asp.

Salgado, Richard P. "Forensics and Incident Response: The Law Enforcement Perspective." 8.4: Systems Forensics, Investigation, and Response. Orlando: SANS Institute, 2002.

United States Department of Justice. "18 U.S.C. 2511. Interception and Disclosure of Wire, Oral, or Electronic Communications Prohibited." April 24, 2000. URL: http://www.cybercrime.gov/usc2511.htm.