Mac Forensics: Mac OS X and the HFS+ File System

Philip Craiger, PhD
Assistant Director for Digital Evidence
National Center for Forensic Science &
Department of Engineering Technology
University of Central Florida
philip@craiger.net

Paul K. Burke
Senior Digital Evidence Research Assistant
National Center for Forensic Science
paulkburke@gmail.com

ABSTRACT

There are few resources that describe a forensics analysis of an Apple Mac computer. The purpose of this paper is describe procedures to conduct a forensics examination of an Apple Mac running the newest operating system, Mac OS X, and its default file system, the Hierarchical File System Plus (HFS+). Our chapter is divided into four sections. In the first we demonstrate Target Disk Mode to create a forensic duplicate of a Mac hard drive and an on-site preview of a suspect's computer. In the second we describe the HFS+ file system and describe the data structures used to represent files and are important in the recovery of deleted files. In the third section we describe several procedures one can use to recover evidence at a physical level to recover evidence from unallocated, slack space, and virtual memory. Finally, we describe methods to recover trace evidence from Mac OS X default email, web browser, and instant messaging applications, as well as forensic procedures to recover commands issued from a terminal window.

Keywords: Mac OS X, Mac OS X forensics, digital forensics, computer forensics.

Mac Forensics: Mac OS X and the HFS+ File System

The Apple Macintosh (or Mac) was first introduced to the public in 1984. Since then it has an enjoyed a small, albeit vocal, user base – typically somewhere between 3 and 8% of the installed operating system base. It is not surprising then that there has been very little published regarding digital forensics on Macintosh computers. To partially rectify this lack of information in this chapter we present an introduction to forensics for the Macintosh.

Due to space limitations we make certain assumptions about the suspect's computer and operating system. The reason for these assumptions is that in our research we discovered that different versions of Mac OS X behave differently. Thus, these assumptions will determine what procedures are required to ensure the forensic integrity of our examination, i.e., some procedures may be proper for one version of Mac OS X, but not another. These assumptions are as follows:

- The forensic acquisition computer and the suspect's computer are running version 10.4.3 of Mac OS X (The latest version as of November 2005).
- The Open Firmware password has not been set on the suspect's computer. (Open Firmware is a processor and system independent boot firmware used in Mac, an analogue to the PCs BIOS.
- The suspect has not used encryption via the Mac OS X FileVault – a virtual image that uses 128-bit AES encryption to create a virtual encrypted volume on the hard drive.
- The suspect's hard drive is formatted with the Hierarchical File System Plus (commonly referred to as HFS+). This HFS+ file system has been the default file system since Mac OS X's inception in 2000.

- The suspect’s computer was used primarily as a personal workstation as opposed to a server. A version of Mac OS X for servers is available, and its use as a server would require a different forensic protocol.

We begin by describing a method for creating forensic duplicates of the suspect’s hard drive. This method also supports an onsite preview of the suspect’s hard drive, a common procedure used by law enforcement to determine if there is any contraband or probative evidence on the suspect’s computer prior to seizing it. We then describe the HFS+ file system, the default file system for Mac OS X installations. We describe HFS+ layout and data structures that are important for recovering deleted files. Next we describe procedures for recovering deleted files from unallocated space as well as the recovery of textual information from virtual memory. We then describe the most popular Mac applications for Web browsing, email, and instant messaging. We describe the locations of potential evidence and any special recovery procedures that may be required to read the recovered evidence. Finally we describe the forensic procedures to recover evidence for suspects who use UNIX command line utilities in the commission of their crimes. We begin our paper with a brief introduction to Mac OS X.

OS X Background

Mac OS X is the successor to the original Apple Macintosh operating system that debuted in 1984. Mac OS X has an entirely different code-base from the original Mac OS, partially based upon the NeXTSTEP operating system code-base. Mac OS X is a UNIX-based operating system that is composed of FreeBSD-based services and a Mach 3.0 microkernel. Although Mac OS X has a tightly integrated user interface that is very ‘Apple-like,’ the underlying architecture is UNIX, with all services and a command line interface that is the heart of any UNIX operating system. Consequently, a thorough forensic examination of a Mac OS X system would require an examiner who is knowledgeable with UNIX-based systems.

Imaging and Onsite Previewing

One of the most important forensic procedures is to create a forensic duplicate, i.e., a bit-for-bit copy, of a suspect's hard drive. Mac OS X provides a novel method for creating a forensic duplicate that requires putting the suspect's computer into *Target Disk Mode* [1]. When the suspect's computer is in Target Disk Mode we connect our forensic acquisition computer and the suspect's computer with a FireWire cable, and create a forensic duplicate of the suspect's hard drive without fear of writing to the suspect's hard drive. (FireWire – also referred to as IEEE-1394 – is a high-speed serial protocol developed by Apple.) Target Disk Mode works with any version of Mac OS X or OS 8/OS 9 (the predecessors to Mac OS X) with FireWire software version 2.3.3 or later [1].

Additionally, Target Disk Mode supports an onsite preview of the suspect's computer. Onsite previews are common when law enforcement is interested in seizing a computer only if there is evidence of probative value on the suspect's hard drive. An onsite preview allows the agent to conduct a search onsite to determine if such evidence exists to warrant a seizure of the suspect's system.

Creating a Forensic Duplicate

It is crucial that nothing causes the suspect's hard drive to mount in read/write mode, as the process of mounting in this mode will cause changes in numerous files on the hard drive. (In our research we found that booting a Windows 98 caused changes in over 400 files. Approximately the same number of files was modified in response to a graceful shutdown.) To prevent Mac OS X from automatically mounting the suspect's hard drive requires disabling the daemon that controls *disk arbitration*, which controls automatic mounting. Under Mac OS X this program and its location is:

```
/usr/sbin/disarbitrationd
```

Disk arbitration can be disabled by simply moving the file from its current directory and rebooting the acquisition computer. After reboot an examiner can check to the acquisition computers *systems log* to determine whether disk arbitration is running:

```
Oct 26 13:33:01 LittleBuddy mdsyncServer[173]: Could not
create DiskArb session or approval session.
```

This example demonstrates that moving the daemon prevented disk arbitration from starting. A second method to determine if the disk arbitration daemon is a running process is through the UNIX command:

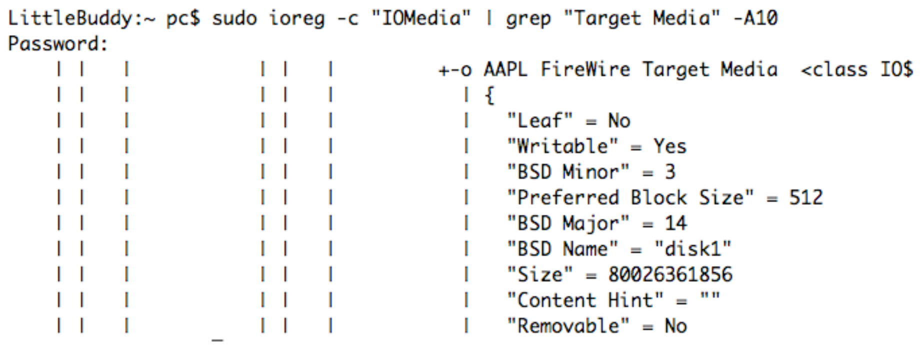
```
$ ps auxw | grep diskarbitrationd
```

This command will show whether `diskarbitrationd` is running. If the result of the command is empty then disk arbitration did not initiate. If the command returns something similar to the following we know that disk arbitration is still running.

```
$ ps auxw | grep diskarbitrationd
$ root 33 0.0 … Ss Fri11AM 0:01.23 usr/sbin/diskarbitrationd
```

The next procedure is to connect the suspect and acquisition computers with a FireWire cable, and power on the suspect's computer while holding down the "T" key on the suspect computers keyboard. This will bring the suspect's computer up into Target Disk Mode. If successful a FireWire icon will display on the monitor of the suspect's computer.

Once disk arbitration is disabled, both computers are connected via a FireWire cable, and the suspect's computer booted into Target Disk Mode, the next procedure is to identify the target (suspect's) computer from the acquisition computer. We use the Mac OS X utility `ioreg`, which will display the Input/Output registry for the Mac to determine which of the physical disks is mapped to the suspect's computers hard drive (see Figure 1 below). Within the registry output we search for the term "APPL FireWire Target Media" as demonstrated in Figure 1 below.



```
LittleBuddy:~ pc$ sudo ioreg -c "IOMedia" | grep "Target Media" -A10
Password:
    | |   |           | |   |        +-o AAPL FireWire Target Media  <class IO$
    | |   |           | |   |           | {
    | |   |           | |   |           |  "Leaf" = No
    | |   |           | |   |           |  "Writable" = Yes
    | |   |           | |   |           |  "BSD Minor" = 3
    | |   |           | |   |           |  "Preferred Block Size" = 512
    | |   |           | |   |           |  "BSD Major" = 14
    | |   |           | |   |           |  "BSD Name" = "disk1"
    | |   |           | |   |           |  "Size" = 80026361856
    | |   |           | |   |           |  "Content Hint" = ""
    | |   |           | |   |           |  "Removable" = No
```

Figure 1. Identifying Physical Device Associated with FireWire Target.

We note that the FireWire Target media, i.e., our suspect's hard drive is mapped to *disk1* (BSD Name=disk1). Under UNIX-based systems, attached physical devices are represented via files, typically located under the `/dev` directory. Therefore, we can now use the UNIX utility `dd` to create a forensic duplicate of the physical disk thusly.

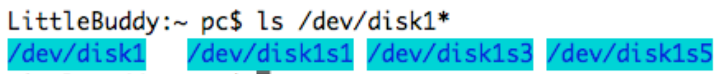
```
$ dd if=/dev/disk1 of=./evidence.dd
```

Finally we verify the integrity of the forensic duplicate by calculating hashes for the physical disk (`/dev/disk1`) and the forensic duplicate (`evidence.dd`).

```
$ md5 /dev/disk1 ./evidence.dd
MD5 (/dev/disk1) = 1cd57cb5d2e438d5be4bca9cf1664986
MD5 (evidence.dd) = 1cd57cb5d2e438d5be4bca9cf1664986
```

Previewing the Suspect's Computer

We must determine information about the volumes contained on the hard drive to preview the contents of the suspect's hard drive from a logical viewpoint. Like other BSD-based systems, volumes are represented by the nomenclature `/dev/disk{n}s{m}`, with `{n}`being a number representing the physical drive, and `s{m}` representing slice number `{m}`. Slice is the BSD nomenclature for a volume or partition. A directory listing under the `/dev` (device) directory for `disk1` shows the following.



```
LittleBuddy:~ pc$ ls /dev/disk1*
/dev/disk1    /dev/disk1s1 /dev/disk1s3 /dev/disk1s5
```

Figure 2. Volumes on Suspect's Computer

Figure 2 above shows that the suspect's hard drive contains three volumes denoted by *s1*, *s3*, and *s5*.

We can conduct an onsite preview of the suspect's computer by mounting each volume read-only and viewing the contents of the volume with Mac OS X Finder (see Figure 3 below). Before we mount the volumes we create a directory (evidence.s3) on our acquisition computer, then manually mount the volume on the suspect's hard drive we wish to view thusly:

```
$ sudo mount -t hfs -r /dev/disk1s3 evidence.s3/
```

The `-t hfs` specifies the type of file system, and the `-r` flag indicates to mount the volume read-only. We can now use the Mac Finder to view the logical files allocated on the suspect's computer (see Figure 3 below).

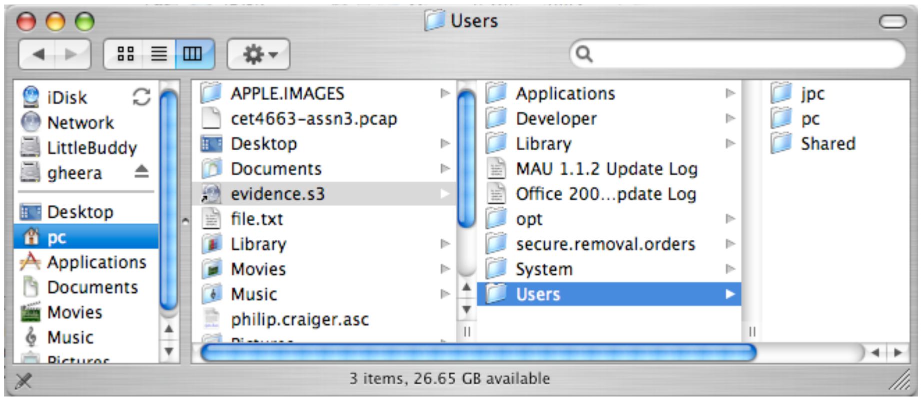


Figure 3. Logical Preview of Suspect's Hard Drive

We can now copy files from the suspect's to the acquisition computer for review without fear of causing any changes on the volume we are viewing. Note that an onsite preview provides us with a view of files in allocated space only. To access deleted files (in unallocated space) and slack space we must access to the hard drive at a physical level. In a later section we explain how to recover evidence in unallocated and slack space.

Summary

Target Disk Mode provides a very simple imaging solution for Mac computers. A second alternative not discussed here is the use of a bootable Linux CD (PowerPC version) to create a forensic duplicate. [2]

HFS+ File System

(The following explanation of the HFS+ file systems is based upon the most authoritative documentation available for the HFS+ file system. [3])

The HFS+ file system replaced the older HFS file system as the default file system for the Mac under Mac OS 8.1. HFS+ provides many improvements over HFS including:

UNICODE for file naming (support for 255 character file names), support for multiple data forks per file, and the use of a 32-bit allocation mapping table, allowing Mac OS X to address 232 (4,294,967,296) allocation units. As of Mac OS X 10.4 Apple has added support for access control lists to the file system, causing it to now be fully compatible with the Windows XP and Server 2003 file system ACLs. HFS+ now supports a maximum file size of 2^{63} bits, or the total size of the volume.

HFS+ Data Structures

There are five special files that define the HFS+ file system including: a) volume header; b) catalog file; c) extents overflow file; d) attributes file; e) allocation (bitmap) file; and f) startup file. For the purposes of this chapter it is important to understand the structure of the catalog file, as it contains metadata that is critical for the recovery of deleted files.

The catalog file describes the folder and file hierarchy on a volume. The catalog file contains metadata about all the files and folders on a volume, including information on modified, access, created times; permissions, file ID, and user information. The complete data structure for the catalog file entries is given below.

```
struct HFSPlusCatalogFile {
    SInt16              recordType;
    UInt16              flags;
    UInt32              reserved1;
    HFSCatalogNodeID    fileID;
    UInt32              createDate;
    UInt32              contentModDate;
    UInt32              attributeModDate;
    UInt32              accessDate;
    UInt32              backupDate;
    HFSPlusPermissions  permissions;
    FInfo               userInfo;
    FXInfo              finderInfo;
    UInt32              textEncoding;
    UInt32              reserved2;
    HFSPlusForkData     dataFork;
    HFSPlusForkData     resourceFork;
};
typedef struct HFSPlusCatalogFile HFSPlusCatalogFile;
```

Two fields that are important for identifying and recovering deleted files are the `dataFork` and `resourceFork` fields, each of which is of type `HFSPlusForkData`. Note that HFS+ supports multiple forks: a resource forks and a data fork. The `dataFork` field specifies information about the location and size of the data fork, that is, where the actual file contents are stored. The `resourceFork` field specifies the same only for resource information, that is, application relevant metadata. The `HFSPlusForkData` data structure is given below:

```
struct HFSPlusForkData {
    UInt64                  logicalSize;
    UInt32                  clumpSize;
    UInt32                  totalBlocks;
    HFSPlusExtentRecord     extents;
};
typedef struct HFSPlusForkData HFSPlusForkData;
```

The `logicalSize` field describes the size (bytes) of the data in the fork. The `totalBlocks` field specifies the number of blocks allocated across all extents for this fork. Finally the `extents` field is an array that stores the first eight extent descriptors.

The `HFSPlusExtentDescriptor` structure is used to hold information about a specific extent within the `extents` field. (An extent is a contiguous segment of a file.)

```
struct HFSPlusExtentDescriptor {
    UInt32                  startBlock;
    UInt32                  blockCount;
};
typedef struct HFSPlusExtentDescriptor
HFSPlusExtentDescriptor;
```

This last data structure will be important in determining the location a file starts on the media, and the number of allocation blocks used to represent the file. We will use this information later to demonstrate the manual recovery of a deleted file.

Physical Analysis (Low-Level Forensics)

A physical analysis views a hard drive at a physical level, that is, without regard to the imposed file system. A physical analysis allows us to recover evidence that cannot be recovered using logical analysis procedures, particularly deleted files and information in slack space. A common first procedure performed during a forensic examination is to recover deleted files, which will require access to the media at a physical level. Below we describe several procedures for recovering deleted evidence.

Recovering Deleted Files

As with versions of Microsoft Windows, there are several methods to delete files under Mac OS X. From the desktop, a user can use the mouse to drag-and-drop a file onto the Trash icon on the Dock. An alternative is to CTRL-Click the mouse over the file, which will bring up a menu from which “Move to Trash” can be chosen. Both of these methods are analogous to dragging and dropping a file onto the Recycle Bin under Microsoft Windows. [4]

Similar to the behavior of the Windows Recycle Bin, files placed in the Mac Trash are not deleted. Rather, the files are copied to special hidden folder and deleted from their original location. Thus, a copy of the file moved to the Trash will still exist in allocated space, and can be recovered easily by double clicking on the Trash icon and dragging the file out of the Trash.

The Trash is represented on the file system as a hidden folder, .Trash, on the root directory of the file system, as shown below in a directory listing from the command line:

```
$/.Trash pc$ ls -al
total 944
drwx------    9 pc  pc     306 Oct 30 15:05 .
drwxr-xr-x   31 pc  pc    1054 Oct 30 12:44 ..
-rw-------    1 pc  pc    6148 Oct 30 14:38 .DS_Store
-rw-r--r--    1 pc  pc  187500 Oct 27 15:41 PPriceList.pdf
-rw-r--r--    1 pc  pc  108382 Oct 27 15:43 EdIPPPL-1.pdf
-rw-r--r--    1 pc  pc  108382 Oct 27 15:43 1019EdIPPPL.pdf
-rw-r--r--    1 pc  pc   61879 Oct 27 15:41 EducationFG.pdf
```

Note that the Trash (.Trash folder) contains four files, each of which can be recovered by simply copying or moving each file to an alternate location. Copying or moving can be accomplished from the desktop via drag-and-drop, or via the command line. (Note: The *.DS_Store* file holds the information that controls the way a folder will be opened; i.e., the shape and size of the window, the position of the window on the desktop and whether file, folder or icon view has been selected. [5])

It is important to determine whether the aforementioned methods for recovering files will inadvertently change date/time stamps, a critical mistake. To determine if either of these procedures changed date and time stamps we conducted a series of simple experiments of removing a deleted file to and from the Trash using different methods and noting differences in modified, accessed, and created times. The original file metadata is as follows:

```
~ $ stat Deleted.file
234881030 966561 -rw-r--r-- 1 pc pc 0 595 "Nov 11 11:08:37
2005" "Nov 11 11:06:12 2005" "Nov 11 11:06:12 2005" 4096 8 0
Deleted.file
```

(Note that the dates/times are the accessed, modified, and created, respectively.) We then drag-and-dropped the file to the Trash from the Finder. Note there were no changes in the date/time stamps.

```
~ $ stat .Trash/Deleted.file
234881030 966561 -rw-r--r-- 1 pc pc 0 595 "Nov 11 11:08:37
2005" "Nov 11 11:06:12 2005" "Nov 11 11:06:12 2005" 4096 8 0
.Trash/Deleted.file
```

We then drag-and-dropped the file from the Trash back to its original location:

```
~ $ stat Deleted.file
234881030 966561 -rw-r--r-- 1 pc pc 0 595 "Nov 11 11:08:37
2005" "Nov 11 11:06:12 2005" "Nov 11 11:06:12 2005" 4096 8 0
Deleted.file
```

Note no changes in dates and times. It appears that Mac OS X preserves dates and time stamps when moving files to and from the Trash, regardless of which method is used. One would expect this behavior, and it appears to be implemented correctly. This consistency allows us to determine the original dates and times for the files.

philip craiger 11/14/05 9:46 AM

Deleted: Once again we moved the file to the Trash, and double-checked that times did not change:

```
~ $ stat .Trash/Deleted.file
234881030 966561 -rw-r--r-- 1
pc pc 0 595 "Nov 11 11:08:37
2005" "Nov 11 11:06:12 2005"
"Nov 11 11:06:12 2005" 4096 8 0
.Trash/Deleted.file
```

Finally, we moved the file back to its original location using the UNIX `mv` command:

```
~ $ mv .Trash/Deleted.file .
~ $ stat Deleted.file
234881030 966561 -rw-r--r-- 1
pc pc 0 595 "Nov 11 11:08:37
2005" "Nov 11 11:06:12 2005"
"Nov 11 11:06:12 2005" 4096 8 0
Deleted.file
```

Again there were no changes in date and time stamps.

Similar to the behavior of the Windows Recycle Bin, files in Trash can be purged permanently. When this occurs, the items in the Trash are 'deleted' from the file system in the traditional sense of the word. As with other file systems, the 'delete' procedure means that the operating system marks the file as deleted, and the space composing the file is marked as 'available', or free for use by the operating system (this would occur in the allocation special file under HFS+). The actual contents of the file will remain on disk until overwritten.

In order to recover files that are deleted in this manner the examiner must access the hard drive at the physical level. This means that the examiner must identify the starting block number of the file and the number of blocks required to store the file.

Recall from above that the `HFSPlusExtentDescriptor` structure holds information about a file's extents within the `extents` field.

```
struct HFSPlusExtentDescriptor {
    UInt32                  startBlock;
    UInt32                  blockCount;
};
typedef struct HFSPlusExtentDescriptor
HFSPlusExtentDescriptor;
```

The only time a file will have more than one extent is when a file is fragmented. Most files will be represented by a single extent. When a file is fragmented, each file fragment will be represented by an extent in this data structure, up to seven extents. If a file consists of more than seven extents the remaining extents will be recorded in the extents overflow special file. Once we find `startBlock` and `blockCount` we can use this information and the UNIX utility `dd` to manually recover a deleted file. For example, say we find that the `startBlock` is 4355500, and the `blockCount` is 18. The default block size under Mac OS X is 8 sectors per block (4096). Thus we can recover the deleted file as follows:

```
$ dd if=/dev/disk1 of=./evidence bs=4096 skip=4355500
count=18
```

We set the block size for `dd` (`bs`) equal to the default block size used by HFS+, skipped to the beginning of the file using the skip command (block 4355500), and then recovered the next 18 blocks, the number of blocks required to store the file. Note that this will recover all data up to the end of the last sector of the block. Therefore this procedure will also recover some of the slack space, i.e., unallocated space that lies after the end-of-file marker up to the end of the last allocation unit of the file, unless the file size is a perfect multiple of the block size.

Searching at a Physical Level

Recall that under UNIX-based operating systems everything is represented as 'file,' including attached physical devices. This means that all attached physical devices can be addressed at the physical, as opposed to logical, level by accessing a file that is mapped (associated) with the physical device. For instance, `/dev/disk0` is typically the primary hard drive.

Recall in our example that `/dev/disk1` is the device file for the suspects hard drive. Say we are interested in searching the suspect's hard drive terms relevant to explosives, particularly 'hand grenades.' To be thorough the examiner will search the entire hard drive (allocated and unallocated space) as the suspect may have deleted files to cover his tracks. We can use the UNIX commands `cat`, `strings`, and `grep` from the command line of a terminal window to filter the hard drive contents (both allocated and unallocated space, including slack space) for the keyword(s). We use `/dev/disk1` as in the input to the `strings` command, which extracts all human readable strings (in the ASCII 7-bit range), and pipe that output to the `grep` command, and search for the keyword 'grenade.' Below is an abbreviated portion of our results. It appears from our results that the suspect has part of the Anarchists Cookbook on his hard drive (based on the author 'Jolly Roger.') Note that the `–C 2` flag informs `grep` to display two lines of context, two lines before and two lines after each keyword hit. If we want to capture more of the file we can provide a larger number for the `C` flag.

```
$ sudo cat /dev/disk1 | strings -o | grep –i 'grenade' –C 5

4294967266 it should be "JACKPOT" time.
4294967243                                 ---------Jolly Roger
4294967228 Portable Grenade Launcher      by the Jolly Roger
4294967228 If you have a bow, this one is for you. Remove the
ferrule from an
```

```
4294967229 aluminum arrow, and fill the arrow with black
powder (I use grade
--
4294967276 Material Required:
4294967277 -----------------
4294967277 Grenade Container
4294967228 Steel pipe nipple, 6 in. (15 cm) long - (see table
1 for diameter)
```

Searching Virtual Memory

Even when a file has been deleted and physically wiped using a secure delete utility [6], there may be traces of the file in virtual memory on the hard drive. We can recover the evidence by searching through virtual memory, which is typically stored as a swapfile on the hard drive. We can search the swapfile for keywords thought or known to exist in the file. In our research we have discovered this procedure to be more successful for recovering textual information than binary or graphical data.

The Mac OS X swapfile is located in `/var/vm`:

```
$ ls -al /var/vm
total 131072
drwxr-xr-x    4 root  wheel       136 Oct 14 10:50 .
drwxr-xr-x   24 root  wheel       816 Oct 14 10:52 ..
drwx--x--x   18 root  wheel       612 Oct 11 11:20 app_profile
-rw------T    1 root  wheel  67108864 Oct 14 10:50 swapfile0
```

We run `strings -o` on the swapfile to extract human readable content and indicate the offset for each keyword, then use `grep` to search for an appropriate keyword, as is demonstrated below.

```
$ sudo strings -o /var/vm/swapfile1 | grep cyberterror -C 2
24057057 $)2DILQSW]dry
24057112 01]NSStrokeWidth"@@
24057264 -/pc/Desktop/cyberterrorism/Definitions of
cyberterrorism.doc
24057344 STR#c
24057360 STR#e
--
34649683 Definitions of cybert#849CA.doc
34649751 PD+W8BNMSWD
34649786 cyberterrorism
```

```
34649847
KLittleBuddy:Users:pc:Desktop:cyberterrorism:Definitions of
cybert#849CA.doc
34650027 AUsers/pc/Desktop/cyberterrorism/Definitions of
cyberterrorism.doc
34650667 PCRAIGER
34650707 craiger.pollitt.ch#FFFFFFFF.doc
```

Application Forensics

Mac OS X comes with several applications that provide a well-rounded suite of applications, including an email reader (Apple Mail), a Web browser (Safari), and a instant messaging application (iChat). Mail, Safari, and iChat all leave a significant amount of trace evidence on the hard drive. Knowing the location of this trace evidence, as well as the format of the evidence, will be crucial during an onsite preview or in executing a logical analysis at the lab. Below we describe trace evidence locations for each of these applications, as well as any special reformatting that must occur to make the evidence human readable. Additionally, below we demonstrate the recovery of evidence generated by a suspect from the UNIX command line, just in case our suspect is UNIX-savvy and used Mac OS X FreeBSD services in the commission of his crime.

Mail

Apple Mail is a full-featured email application that supports multiple POP3 and IMAP accounts and advanced filtering. Anecdotal evidence suggests that Apple Mail is the most popular email application for the Mac.

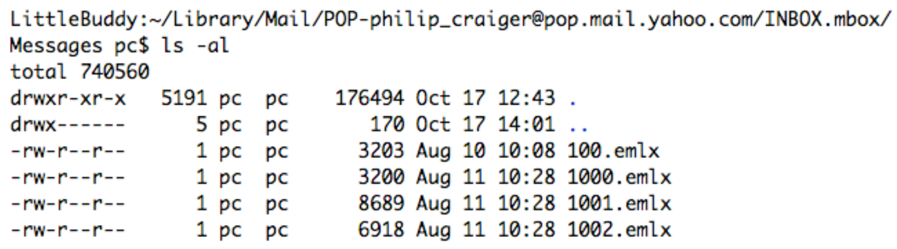
A user's emails are located in the following directory:

```
/Users/<username>/Library/Mail
```

The `/Users` directory is the Mac analogue of the UNIX `/home` directory, where individual users' files are located. Consistent with UNIX philosophy of multi-users systems, each user has his/her own directory.

Prior to Mac OS X 10.4, Apple Mail files were simple text files in the *mbox* format [7], the default format for many mail applications. *mbox* files are simple flat text files with individual emails appended they are downloaded. *mbox* files can be imported from one mail application to another – as long as the application understands the standard *mbox* format (most of which do).

As of Mac OS 10.4 Apple changed the default Apple Mail format to *emlx*. Under the *emlx* format each individual email is its own file in ASCII format (see Figure 4 below). Apparently the change from *mbox* to *emlx* was made to allow for more thorough indexing under Apple's new integrated search technology named SpotLight. [8]



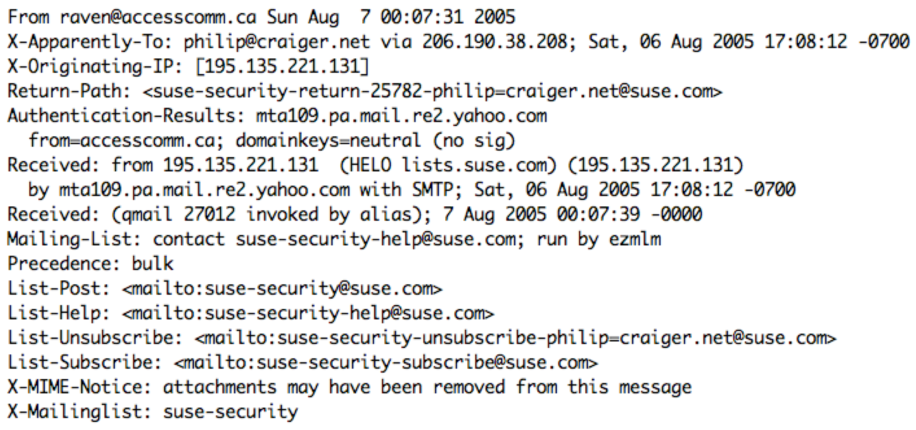
```
LittleBuddy:~/Library/Mail/POP-philip_craiger@pop.mail.yahoo.com/INBOX.mbox/
Messages pc$ ls -al
total 740560
drwxr-xr-x   5191 pc  pc    176494 Oct 17 12:43 .
drwx------      5 pc  pc       170 Oct 17 14:01 ..
-rw-r--r--      1 pc  pc      3203 Aug 10 10:08 100.emlx
-rw-r--r--      1 pc  pc      3200 Aug 11 10:28 1000.emlx
-rw-r--r--      1 pc  pc      8689 Aug 11 10:28 1001.emlx
-rw-r--r--      1 pc  pc      6918 Aug 11 10:28 1002.emlx
```

Figure 4. Individual emails under the emlx format

Because each email is a simple text file we can use the UNIX utilities to search for specific keywords with the file from the command line.

One method often used to view a suspect's email is to import the email file into another email application that supports that type of format. Consequently, we could copy the folder containing *emlx* files to our acquisition computer, and import them into our own Apple Mail application for viewing.

An alternative is to import the files into another (non-Apple Mail) email application, such as Microsoft's Outlook, Mozilla's Thunderbird, or Qualcomm's Eudora. Unfortunately, we cannot directly import *emlx* files into a secondary mail application for viewing a suspect's emails as these applications do not understand the *emlx* format. We must first use a converter, such as *emlx converter* (http://www.cosmicsoft.net/emlxconvert.html) to convert the *emlx*-based individual emails into a single *mbox* file. *mbox* is a standard flat text file containing headers and content for each email, appended end-to-end, as shown in Figure 5 below.



```
From raven@accesscomm.ca Sun Aug  7 00:07:31 2005
X-Apparently-To: philip@craiger.net via 206.190.38.208; Sat, 06 Aug 2005 17:08:12 -0700
X-Originating-IP: [195.135.221.131]
Return-Path: <suse-security-return-25782-philip=craiger.net@suse.com>
Authentication-Results: mta109.pa.mail.re2.yahoo.com
  from=accesscomm.ca; domainkeys=neutral (no sig)
Received: from 195.135.221.131  (HELO lists.suse.com) (195.135.221.131)
  by mta109.pa.mail.re2.yahoo.com with SMTP; Sat, 06 Aug 2005 17:08:12 -0700
Received: (qmail 27012 invoked by alias); 7 Aug 2005 00:07:39 -0000
Mailing-List: contact suse-security-help@suse.com; run by ezmlm
Precedence: bulk
List-Post: <mailto:suse-security@suse.com>
List-Help: <mailto:suse-security-help@suse.com>
List-Unsubscribe: <mailto:suse-security-unsubscribe-philip=craiger.net@suse.com>
List-Subscribe: <mailto:suse-security-subscribe@suse.com>
X-MIME-Notice: attachments may have been removed from this message
X-Mailinglist: suse-security
```

Figure 5: Header of an mbox Formatted Email File

The *mbox* file can be viewed with a standard text editor or imported into an *mbox*-aware mail application.

iChat

One of the most popular computer applications today is instant messaging (IM). IM applications are bundled with all major operating systems (Windows, Linux, Mac OS

X). Additionally, all major web-based mail providers, including AOL, MSN, Yahoo!, and Gmail provide instant messenger services for users.

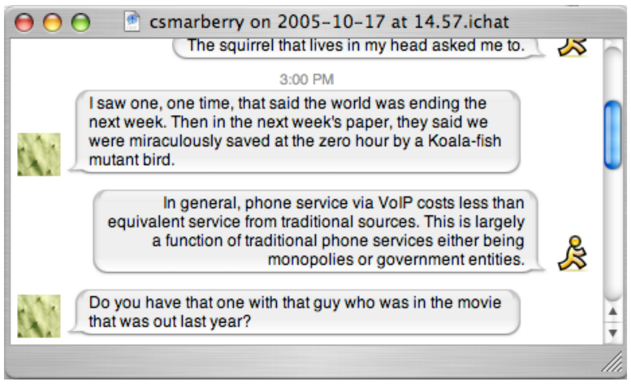
Through our research we determined that different IM applications store the trail of instant message conversations differently: Some such as certain versions of AOL AIM store the instant messages on AOL servers, not on the client's machine. Some IM applications will keep a history of previous conversations. iChat does not automatically store previous conversations; however a user preference allows users to store iChat conversations. The default location for these stored iChats is:

```
/Users/<username>/Documents/iChats
```

These files are named as follows:

```
<username> on <date> at <time>.ichat
```

We can view these files by copying them to our acquisition computer and opening them the file in iChat by double clicking on their icons. Figure 6 below shows a recovered iChat conversation from two iChat users.


csmarberry on 2005-10-17 at 14.57.ichat

The squirrel that lives in my head asked me to.

3:00 PM

I saw one, one time, that said the world was ending the next week. Then in the next week's paper, they said we were miraculously saved at the zero hour by a Koala-fish mutant bird.

In general, phone service via VoIP costs less than equivalent service from traditional sources. This is largely a function of traditional phone services either being monopolies or government entities.

Do you have that one with that guy who was in the movie that was out last year?

Figure 6: Recovered iChat

Web Browsing

Apple's bundled Web-browser, Safari, has become the de-facto standard browser for Mac users. Safari has seen increasing market share primarily because Internet Explorer is no longer bundled with the latest versions of Mac OS X.

Note that all customizations that a user makes to his system are stored in XML formatted files that end in *.plist*. We can view the possible 'domains' (applications) that have been customized by issuing the command:

```
$ defaults domains
```

We can read the customizations for one of the domains by issuing:

```
$ defaults read <domain name>
```

A suspect's web browsing history, download history, and bookmarks are often recovered and used for evidence in various criminal cases. The bookmark, download history and browsing history are stored in the following location:

```
/Users/<username>/Library/Safari
```

The files used to store this information are named `Bookmarks.plist`, `Downloads.plist`, and `History.plist`, respectively. `History.plist` stores recent Web searches performed from Safari's built-in Google search bar. `Bookmarks.plist` stores Safari bookmarks, and `Downloads.plist` stores the history of files downloaded from within Safari. One can read each of the files using the `defaults read` command. Below we read the `History.plist` file to determine what Google keywords have been of interest to our suspect:

```
~ $ defaults read ~/Library/Safari/History
{
  …
```

```
"http://www.google.com/search?q=bomb+making&ie=UTF-8&oe=UTF-
8";
lastVisitedDate = "153417328.9";
title = "bomb making - Google Search";
visitCount = 1;
…
"http://www.google.com/search?q=sex+drugs+rock+and+roll&ie=UTF
-8&oe=UTF-8";
lastVisitedDate = "153417324.8";
title = "sex drugs rock and roll - Google Search";
visitCount = 1;
…
```

The snippet above indicates that our suspect performed at least two searches with Safari's Google toolbar: once for the keywords 'bomb making,' and once for 'sex, drugs, and rock and roll.'

Safari follows a traditional Mac OS X layout with its preferences and data files. Cache files are located in:

```
/Users/<username>/Library/Caches/Safari
```

Below is a directory listing that displays the naming scheme used in the folders and files:

```
~/Library/Caches/Safari/00/00 $ ls -al
total 256
drwx------   13 pc  pc    442 Aug 21 17:05 .
drwx------   18 pc  pc    612 Mar  6  2005 ..
-rw-------      1  pc   pc     1385  Jul  27  20:04  1113936647-
3722688267.cache
-rw-------      1  pc   pc    48227  Aug  21  17:05  1199711745-
0794204939.cache
-rw-------      1  pc   pc     2163  Jul  22  19:58  1333575177-
3641883655.cache
-rw-------      1  pc   pc     9188  Jul  23  15:07  1375667714-
3116260872.cache
```

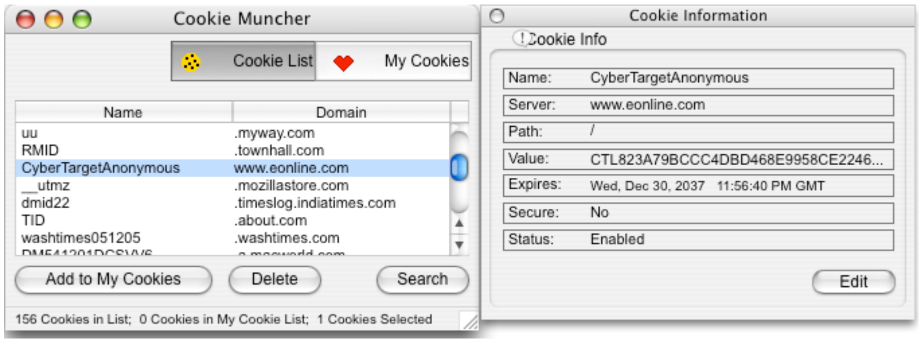
As displayed, the browser cache is separated into several files that span a set of two-digit numbered folders, each composed of another set of two-digit numbered folders.

We can view these files by copying these folders onto our acquisition computer, and opening them in Safari.

Cookie files are stored in XML format in the file:

```
/Users/<username>/Library/Cookies/Cookies.plist
```

As with the other plist files, it is possible to transfer this file between machines and fool Safari into using that version of the file instead. The latest versions of Safari also include the option to view stored cookies in the browser. By going to Preferences: Security and clicking "Show Cookies" one can view all data associated with each cookie. You can also use an external viewer such as Cookie Muncher (http://www.dotsw.com/cookiemuncher.html) to view the bookmarks. Figure 7 shows the result of Cookie Muncher displaying the contents of a Cookie.



Name	Domain
uu	.myway.com
RMID	.townhall.com
CyberTargetAnonymous	www.eonline.com
__utmz	.mozillastore.com
dmid22	.timeslog.indiatimes.com
TID	.about.com
washtimes051205	.washtimes.com

156 Cookies in List; 0 Cookies in My Cookie List; 1 Cookies Selected

Name:	CyberTargetAnonymous
Server:	www.eonline.com
Path:	/
Value:	CTL823A79BCCC4DBD468E9958CE2246...
Expires:	Wed, Dec 30, 2037 11:56:40 PM GMT
Secure:	No
Status:	Enabled

Figure 6. Cookie Muncher Displays Cookie

Evidence from the Command Line

A UNIX-savvy Mac user may open a terminal and use UNIX commands in the commission of a crime. This is most likely where an examiner will find evidence of a network intrusion. Moreover, a savvy user may download and install any sort of UNIX-based application, given that Mac OS X is UNIX-based. In fact, installing UNIX applications under Mac OS X has become easy because several packaging systems are now available, including Fink (http://fink.sourceforge.net) and DarwinPorts (http://darwinports.opendarwin.org).

The bash shell is the default shell installed with Mac OS X 10.4. One default behavior of the bash shell is to record commands issued from a terminal to a file named *.bash_history* (a hidden file) in the users default directory. .bash_history is a simple text file that contains a list of the last 1000 commands issued by the user from the command line. .bash_history acts like a queue (FIFO) where the least recent commands are purged once the HISTORY_SIZE environment variable (typically 1000) is exceeded. (Note that the bash shell does not write the commands issued from within a terminal until that terminal window is closed. Therefore, if a user has left a terminal open, there may be commands in memory that have not been written to the .bash_history file.)

To illustrate, we copied the suspect's .bash_history file from his computer to our acquisition computer, then viewed the last few lines of the file. Here are the last entries of his .bash_history file:

```
…
550  sudo nmap -sS 192.168.1.2 -O > /Volumes/leet/recon.txt
551  sudo nmap -sS 192.168.1.1 -O > /Volumes/leet/recon2.txt
552  sudo nmap -sS 192.168.1.3 -O > /Volumes/leet/recon3.txt
553  cd /Volumes/leet/
554  cat recon.txt
555  cat recon2.txt
556  cat recon3.txt
557  rm recon*
```

According to the suspect's .bash_history file he performed several port scans with OS fingerprinting (note the `-O` flag) on an internal network (note the private addresses). The suspect saved his results to another volume named 'leet', viewed the files, and subsequently deleted the files. We can recover the deleted files by accessing the volume at the physical level and searching for keywords we know will exist in the file:

```
~ $ cat /dev/disk5 | strings | grep -i nmap -C 10
…
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2005-
11-11 14:43 EST
Interesting ports on 192.168.1.2:
(The 1657 ports scanned but not shown below are in state:
closed)
PORT     STATE SERVICE
22/tcp   open  ssh
515/tcp  open  printer
548/tcp  open  afpovertcp
631/tcp  open  ipp
3689/tcp open  rendezvous
5000/tcp open  UPnP
No exact OS matches for host (If you know what OS is running
on it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=3.81%P=powerpc-apple-arwin8.2.0%D=11/11%
…
```

Above is a snippet of the evidence we recovered. We were able to recover the contents of all three files that the suspect deleted. Of course, the files could have been overwritten by the OS had we not worked quickly to recover the evidence.

There are several dozen locations and procedures for recovering evidence from a UNIX system. We point to the reader to the following for a more exhaustive description of these locations and procedures. [1] [9] [10]

Conclusion

Our paper has only scratched the surface of Mac OS X forensics. We had to limit the scope of our coverage due to space limitations and the fact that there are variations in how Mac OS X behaves that are relevant to forensic processing. Our research with different versions demonstrated that there are several changes in each version (major and/or minor) of Mac OS X that require a new plan of attack to perform forensically sound examinations. Because of these variations it is important that a forensic examiner have a document that lists the protocols for each major and minor version of Mac OS X. This helps ensure that improper procedures are not used that may taint the evidence, for instance, by writing to the suspect's hard drive.

Moreover, we could not cover many other well-known applications due to space limitations. In subsequent chapters we will cover other Mac OS X email applications (such as Microsoft's Entourage and Mozilla's Thunderbird), other web browsers (Microsoft's Internet Explorer and Mozilla's Firefox), and other popular instant messengers (AOL's AIM and Adium). Our future research efforts into Mac OS X forensics will concern developing protocols for all major and minor versions as well as research on forensics for other popular applications for Mac OS X.

References

[1] Apple Computer, Target Disk Mode. (http://developer.apple.com/documentation/Hardware/Developer_Notes/Macintosh_CPUs-G4/PowerMacG4_16Jan01/3Input-Output/Target_Disk_Mode_.html), 2002.

[2] Craiger, J.P., Recovering Evidence from a Linux System. In M. Pollitt & S. Shenoi (Eds.), Advances in Digital Forensics. International Association of Information Processing, 2005.

[3] Apple Computer, Technical Note TN1150: HFS Plus Volume Format. (http://developer.apple.com/technotes/tn/tn1150.html), 2004.

[4] Microsoft Corporation, How the Recycle Bin Stores Files. (http://support.microsoft.com/default.aspx?scid=kb;en-us;136517&Product=w95), 2004.

[5] Computer Knowledge, File Extension Details for .DS_Store. (http://filext.com/detaillist.php?extdetail=.ds_store), 2003.

[6] P. K. Burke and J. P. Craiger, Digital Trace Evidence from Secure Deletion Programs. Submitted to the International Conference on Digital Forensics 11.9, 2005.

[7] Network Working Group, RFC 4155 - The application/mbox Media Type. (http://www.faqs.org/rfcs/rfc4155.html), 2005.

[8] Apple Computer, Working with Spotlight. (http://developer.apple.com/macosx/spotlight.html), 2005.

[9] D. Farmer, D.and W.Venema, Forensic Discovery. Prentice-Hall, 2004.

[10] K. Jones, R. Bejtlich and C. Rose, Real Digital Forensics: Computer Security and Incident Response. Addisson-Wesley Professional, 2005.