Mac OS Lion Forensic Memory Acquisition Using IEEE 1394

Todd Garrison

September 17, 2011

Abstract

In Mac OS version 10.7, also known as Lion, Apple (Apple - OS X Lion - The world's most advanced OS., n.d.) has introduced several countermeasures to protect against well-known IEEE 1394 Direct Memory Access (DMA) methods that allow the forensic imaging of RAM (Dalrymple, 2011). It was found that it is possible to subvert these measures in cases where default operating systems settings are used, and the "Fast User Switching" feature is configured on the system or if a user had logged off but the system remained running; these methods work even if full-disk encryption (FDE) is enabled. There are configuration settings that can mitigate this threat which includes modifying sleep settings, disabling fast user switching, and enabling FileVault 2 encryption.

Mac OS Lion Forensic Memory Acquisition Using IEEE 1394

Introduction

The use of the IEEE 1394 protocol for capturing RAM images of running computers is not a new forensic technique, with well documented examples dating from 2004 (Hermann, 2008). Aside from being used for lawful forensic purposes, the technique also has security implications that can be used to compromise the confidentiality or integrity of information. Through testing it was confirmed that Mac OS Lion implements countermeasures against using IEEE 1394 to directly access the RAM of a computer that has been locked. This discovery prompted several questions:

1. How extensively are the preventative measures implemented?
    1.1. Do the restrictions differ with the system state, such as when the screen is locked, the user is logged out, and so on?
2. Is it possible to bypass these restrictions?
3. Are passwords or password hashes available within memory?
4. Are configuration options available that prevent the use of IEEE 1394 for compromising the security of Mac OS Lion?

Motivation

It is noteworthy that in July of 2011 Passware, Inc. issued a press announcement that their tools were capable of recovering passwords for systems running Mac OS Lion with FileVault 2 enabled (Koukoushkina, 2011). The press release did not provide any details of how this was accomplished or the circumstances applicable. The research performed here was in-part inspired by the announcement. If there are ways to ensure that FileVault 2 can be configured securely, it

is important that this information is made available to individuals and organizations that rely on this software to protect their confidential information.

Method

Test systems and software

For the sake of simplicity, the system that is being imaged will be called the *Target*, and systems that are performing and analyzing the acquired image are referenced as the *Host*. A variety of systems were used for testing, and some minor variations were discovered depending on the hardware platform (see *Discrepancies in Target Systems* below for details), though these discrepancies had little overall effect on the results. Imaging was achieved by connecting the Target and Host systems directly to each other using a Firewire/IEEE 1394 cable. To ensure that the Target system was the first device on the bus no other devices were connected to the IEEE 1394 chain.

pythonraw1394

(See *Appendix C* for usage and installation demonstration.) The pythonraw1394 libraries work on Linux systems that have the raw1394 kernel extension available (Boileau, 2006). This set of kernel extensions have been replaced by the "juju" modules in the most popular Linux distributions, such as Ubuntu (FireWire - Community Ubuntu Documentation, 2011). Only older versions of these Linux distributions that are no longer supported, have the kernel modules necessary to make this software work. Despite relying upon outdated kernel modules, the pythonraw1394 library includes several useful utilities, and has been widely tested for performing memory forensics (Schuster, 2008). Because these tools are known to work and have been used by the author to perform previous forensic RAM acquisitions, they were used in addition to more modern tools to assist in validating the results of testing.

libforensic1394

(See *Appendices A and B* for installation and usage examples.) The libforensic1394 library is a more modern library than pythonraw1394 and allows access to RAM on a target host using IEEE 1394. The libraries do not include any programs to perform the acquisition of RAM; however, some examples are provided of how to use the libraries, and the library's methods are well documented (Witherden, 2011). Consequently, it was possible to create a simple Python script to perform the capture of RAM (see *Appendix B* for Python program code).

The libraries are designed to work on both Linux and Mac OS. On Linux the libraries use the new "juju" firewire kernel modules, and hence libforensic1394 will work on up to date Linux systems. On Mac OS it is designed to use the IOKit interfaces developed by Apple.

It was discovered that using these libraries on Mac OS Lion was unstable, resulting in kernel panics anytime an error situation was encountered. On an Ubuntu 11.04 system the libraries provided consistent results, handling errors gracefully but silently.

Analyzing Captured RAM Images for Passwords

Because the passwords are stored in RAM in a clear text format, it was not necessary to create any specialized tools for extracting passwords. Through testing it was found that certain text strings were frequently in proximity to the plain text password in the RAM image. Using the Unix "strings" and "grep" commands it is possible to find the information. The following command consistently was capable of extracting the password from the RAM image:

```
strings ramdump.bin |grep --after-context=3 managedUser \
         |grep --after-context=1 password
```

Figure 1: Command to Extract Plaintext Passwords from RAM Image.

There are other strings associated with passwords and password hashes that are accessible in the RAM image file. Because this is not a discussion of how to acquire passwords this topic will not be explored further.

Mac OS Targets

The Target systems were chosen strictly on basis of availability. As a consequence, the results in this paper may be incomplete. It is believed that the results are generally applicable to Macintosh provided 64-bit Intel architecture systems, but this cannot be confirmed.

The following is a list of the various configurations used during testing. Each listing implies a unique operating system installation.

- MacBook Pro Intel Core Duo 2
    - Lion 10.7 FDE enabled, operating system installed on external USB drive.
    - Lion 10.7 FDE not configured, operating system installed on external USB drive.
    - Lion 10.7.1 FDE enabled, operating system installed on external USB drive.
    - Lion 10.7.1 FDE not configured, operating system installed on external USB drive.
    - Snow Leopard 10.6.8 FileVault 1 enabled for some accounts, system installed on internal drive. *[Snow Leopard Targets were used as a control to ensure that tools used for performing acquisition were working properly.]*
- iMac 27” i5
    - Lion 10.7.1 FDE not configured, operating system installed on primary internal drive.
    - Lion 10.7.1 FDE enabled, operating system installed on primary internal drive.
    - Lion 10.7 FDE enabled, operating system installed on external USB drive.
- MacBook Pro Intel Core Duo (32-bit architecture)
    - Snow Leopard 10.6.8 FileVault 1 not enabled, installed on primary internal drive.

Ubuntu 11.04 Host

Using Ubuntu was an arbitrary choice, any modern Linux distribution would have sufficed, although the configuration would have differed. The Host system used for capturing RAM uses an Intel Core Quad processor, and has a Motorola chipset-based PCI Firewire Card with Firewire-400 ports. Ubuntu was installed as a 64-bit operating system with default settings.

Ubuntu 8.04.4 Host

To test the pythonraw1394 library an older Linux distribution was required that still had the raw1394 kernel module available. Ubuntu 8.04.4 was chosen as it was known that this operating system had the requisite module available. The 32-bit version of the operating system was installed with default settings onto a Toshiba Laptop. The system was equipped with an AMD X2-64 processor, and Xerox-based Firewire interface.

MacOS 10.7 Host

The libforensic1394 library is designed to work with the IOKit system on Mac OS. It is possible to capture RAM images using Mac OS Lion, however, any error reading from the IEEE 1394 port from the Target system results in the Host system performing a kernel panic. Because of the experimental nature of the testing, and the additional time involved in recovering from multiple kernel panics, the use of Mac OS Lion as an acquisition host was abandoned early in the testing.

Results

Overview of IEEE 1394 Access Controls in Mac OS Lion

Through testing it was determined that attempts to access RAM using the DMA features of the IEEE 1394 device on a Macintosh computer were blocked under certain circumstances.

Access controls were activated that prevented direct RAM access when the screensaver was activated and configured to require a password on subsequent login. In many circumstances, it is possible to bypass these countermeasures (see *Table 1* below). While the access controls preventing DMA were active, attempting to capture a memory image using a Linux Host provided a dump file containing only null (0x00) bytes, and on a Mac OS Lion Host the access controls caused a kernel panic.

In Lion's pre-boot authentication state for FileVault 2 protected systems, it was not possible to capture a RAM image under any circumstance.

It is possible to configure the system in a manner that fully prevents any effort to access RAM using IEEE 1394 DMA (see section titled *Mitigating the Ability to Capture RAM*) with the exception of a booted system with a user logged-in, and no screen locking enabled; a scenario where the system is already in an insecure state.

Testing Results Depending upon Configuration and System State

Ten different operating system installations were tested using varying configurations, resulting in nine different result scenarios. The different configurations are as follows:

- Different operating systems: 10.7, 10.7.1, and 10.6.8 as a control.
- Different system states: logged-in unlocked, logged-in locked, booted not logged-in, logged out, pre-boot FDE authentication, and finally testing the proposed countermeasures that supplement Apple's countermeasures to DMA.
- Different levels of encryption: No FileVault 2, FileVault 2 enabled, FileVault 1 on 10.6.
- Different hardware: Two MacBook Pro laptops, One iMac Intel i5.

The following table shows the results of the different scenarios, with two exceptions (noted in the *Discussion* section below), hardware differences had no bearing on test results:

System State:	libforensic1394 SBP-2 mode enabled	libforensic1394 without SBP-2	pythonraw1394
Lion: system hibernated, with volume encryption key set to be deleted upon sleep, FDE enabled [3]	No access	No access	No access
Lion: User logged on, screen unlocked, FDE enabled	No access	Read/Write RAM access, passwords available as plain text.	Read/Write RAM access, passwords available as plain text.
Lion: User logged on, screen locked, FDE enabled	No access	Read/Write RAM access, passwords available as plain text. [1]	Read/Write RAM access, passwords available as plain text. [1]
Lion: User logged off, FDE enabled	No access	Read/Write RAM access, passwords available as plain text.	Read/Write RAM access, passwords available as plain text.
Lion: User logged on, screen unlocked, FDE not enabled	No access	Read/Write RAM access, passwords available as plain text.	Read/Write RAM access, passwords available as plain text.
Lion: User logged on, screen locked, FDE not enabled	No access	Read/Write RAM access, passwords available as plain text. [1]	Read/Write RAM access, passwords available as plain text. [1]
Lion: System fully booted, user not logged in, FDE Not enabled	No access	Read/Write RAM access, passwords available as SHA-2 512 bit salted hashes in binary form.	Read/Write RAM access, passwords available as SHA-2 512 bit salted hashes in binary form.
Snow Leopard, System booted, no users have logged in	Read/Write RAM access, no plaintext passwords, password hashes available as salted SHA-1 in binary form. [2]	Not tested.	Read/Write RAM access, no plaintext passwords, password hashes available as salted SHA-1 in binary form. [2]
Snore Leopard: User has logged in, or user has logged in and then logged off	Read/Write RAM access, plaintext passwords available.	Not tested.	Read/Write RAM access, plaintext passwords available.

Table 1: Results of RAM Acquisition in Various System States.

Footnotes:

[1] Requires that the "Switch User" button be clicked on the unlock screen. If this is not available, as would be the case if there is only one user configured on the system or if user switching is disabled, RAM acquisition is not possible.

[2] This is of limited utility in acquiring passwords. Because the disk is not encrypted it is much easier to extract password hashes directly from files on disk (see: Garrison, 2011) than to search a large RAM image.

[3] These are not system defaults and require settings to be changed: see following section titled "Mitigating the Ability to Capture RAM".

Mitigating the Ability to Capture RAM

The combination of setting specific sleep options, disabling fast user switching and enabling FileVault 2 will prevent the RAM acquisition methods disclosed in this paper.

Sleep Options

It is possible to prevent access to RAM while the system is not in use by configuring the system to "hibernate". This is an alternative to sleep mode, where the contents of RAM are written to the disk, the system is shutdown, and when awoken requires authentication before loading the stored contents of RAM. Mac OS offers several options for configuring the sleep mode for the operating system. The configuration settings are accessible through the command line tool "pmset".

According to the pmset manual, there are two relevant options: *destroyfvkeyonstandby* and *hibernatemode* (pmset(1) Mac OS X Manual Page, n.d.). By default neither of these settings are configured in a manner that provides protection against forensic RAM acquisition. Through testing it was determined that by changing the settings it is possible to disable the capture of

RAM against a system that has been put into sleep mode. This is particularly important for protecting laptop computers.

```
sudo pmset -a destroyfvkeyonstandby 1 hibernatemode 25
```

Figure 2: The pmset Command Used to Configure Settings.

Option	Value	Description
destroyfvkeyonstandby	1	Removes the full volume encryption key from RAM when the system is put into sleep mode and is dependent on the value of hibernatemode.
hibernatemode	25	Forces the system to immediately write RAM to disk and remove power from memory upon sleep.

Table 2: Explanation of pmset Options to Protect Against Memory Acquisition (pmset(1) Mac OS X Manual Page, n.d.)

There are some downsides to using these settings:

1. The user will be required to authenticate twice upon waking the system, once for decrypting the volume where memory was stored, and again to login to the computer.
2. Waking from disk is significantly slower than RAM.
3. Testing shows that systems using external drives, such as a USB storage device, for the operating system volume will not write memory to disk and are not protected by these settings.

Fast User Switching

The ability to capture RAM relies on the relaxation of access controls when the Switch User button is pressed on the login screen. This feature can be disabled in the System Preferences, by selecting “Users and Groups”, clicking on “Login Options”, and unselecting the “Show fast User switching menu as” check box (Jacob, 2011).

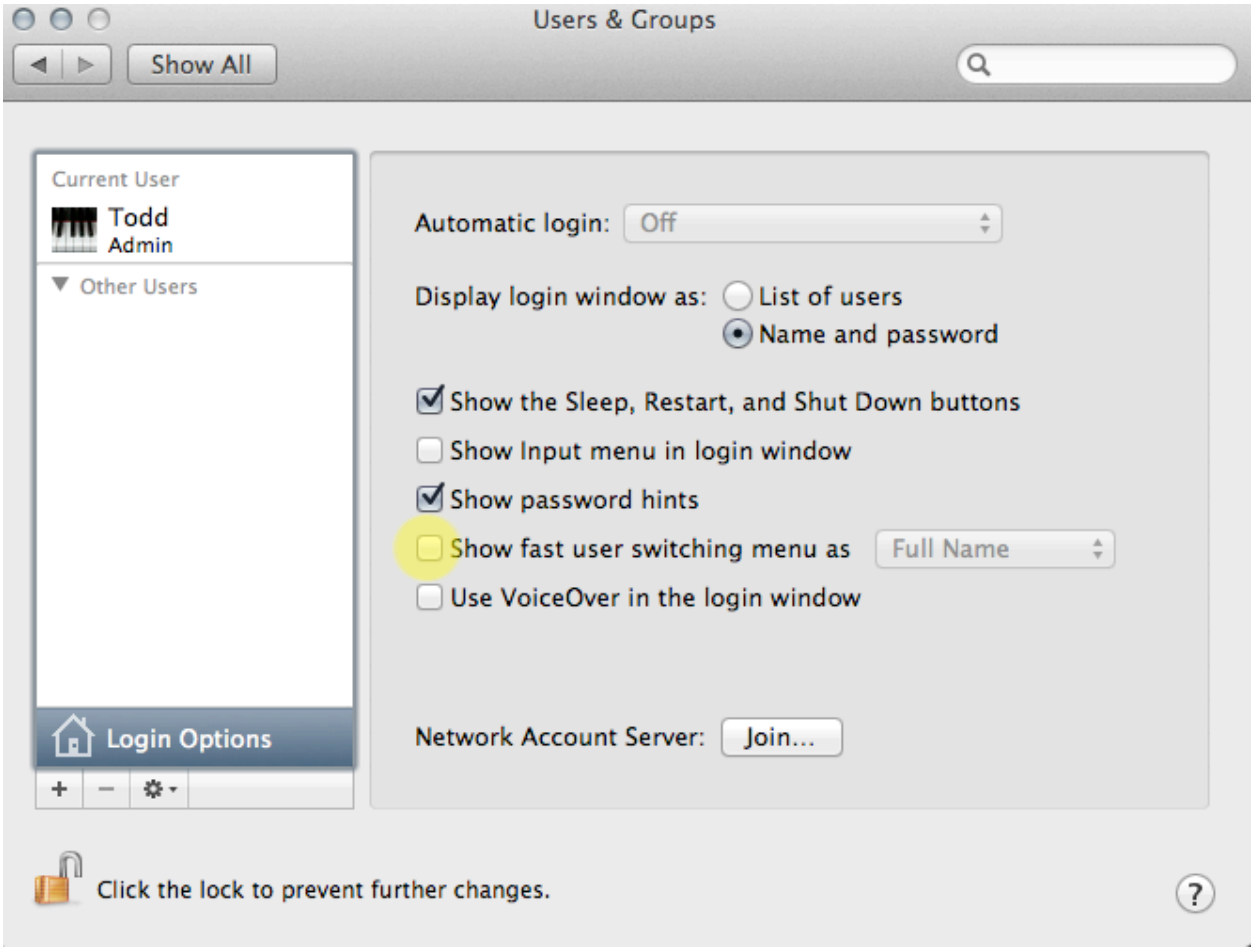

Figure 3: Disable Fast User Switching.

Discussion

Discrepancies in Target Systems

The older MacBook Pro, 15 Inch Core 2 Duo system that was used reported a RAM page size of 2048 bytes when queried through the libforensic1394 libraries. The iMac 27-inch, Late 2009 system reported a RAM page size of 4096 bytes. When attempting to access the iMac system using 4096 byte pages RAM was not accessible, however, when forcing the page size to 2048 the RAM was accessible. As a result the Python script used to perform memory dumps was modified to only use 2048 byte pages when not using SBP-2 mode, and to ignore reported values.

Systems using an external hard drive for the operating system would not hibernate to disk when the "hibernatemode" option was manually set to a value of 25, and because RAM stays active in this case, the FDE key was not removed. This implies that the suggested configuration for preventing DMA is not valid on systems using an external USB drive for the operating system. It is unknown if this limitation is present on external drives using other connection methods such as Firewire, eSata, or Thunderbolt.

Lion Full Disk Encryption

The FileVault 2 product from Apple implements full-disk encryption using AES-XTS mode encryption using a 128 bit key (OS X Lion: About FileVault 2, 2011). Further examination shows that the system implements a boot-time authentication scheme that uses a separate drive partition to store a minimal boot loader and encryption keys (Fleischer, 2011). The system is first booted from the HFS+ formatted disk partition labeled "Recovery HD." Exact details of how the boot process are not known, but some information is available from observations. The file "EncryptedRoot.plist.wipekey" on this partition, which appears to be encrypted, grows by roughly 300 kilobytes each time a new user is added. This would suggest that boot settings and encryption keys are stored in this file. It is likely that the file employs multiple layers of encryption because a list of user names is presented at boot time. It may be possible that there are methods of attacking the pre-boot state of FileVault 2. If for example, password hashes are stored in the file it might be possible to recover credentials. FireWire access appears to be disabled at the kernel level in this pre-boot state so it is unlikely that kernel debugging would work.

Thunderbolt

Apple has started shipping new computers with a new technology named Thunderbolt (Apple - Thunderbolt: Next-generation high-speed I/O technology., n.d.). Thunderbolt also likely provides DMA access, but in a different manner. It is probable that Thunderbolt represents similar risks to IEEE 1394 (Graham, 2011). Because Thunderbolt provides PCIe access the protections provided against Firewire attacks may not apply. Testing Thunderbolt was outside the scope of this paper, and at present there are no known tools for attempting these attacks.

Test Methodologies that Failed to Produce Results

Several other theories were tested to see if the DMA restrictions could be bypassed. These methodologies did not result in a successful bypass of the controls blocking DMA. Not all of these techniques were fully investigated, but they are listed nonetheless for completeness.

Emulating Firewire-Audio Device

The pythonraw1394 tools have the ability to extract and emulate the identification of another IEEE 1394 device. A Behringer FCA202 audio device was emulated without gaining access. This was tested both before and after the actual device had been connected to the system. It did not matter if the device had been used. It was also tested with the actual device in the chain between the Target and Host systems. (See *Dangers to Hardware* below for a warning about connecting multiple Firewire host systems to a single device). This also did not produce an access control bypass.

Using Hardware to Relax Access Controls

A theory that an actively connected device that requires DMA might be used to perform a RAM capture. The theory was that the access control might apply to the entire bus, and not a specific device on the bus. By hooking a Firewire peripheral device in the middle of the chain it might be possible to trick the Target system into disabling the DMA restrictions. Testing

suggests that the access control preventing DMA is applied at a device level, and not at the bus level.

There was an anomaly during the testing of this theory that could not be confirmed. At one point it appeared that DMA had been gained on a 10.7 Target with the screen locked by placing a *Presonus Firepod* device in between the Target and Host system at boot time and subsequently locking the screen. However, due care was not exercised during connecting the cabling and the device that appeared to provide access was physically damaged. It is likely, but not confirmed, that this anomaly can be attributed to another variable that was not accounted for during the test, such as the "Switch User" feature being activated. Because of the hardware failure it was not possible to confirm the results and this anomaly should therefore be considered a false positive. Other Firewire audio devices did not yield similar results.

Dangers to Hardware

Using a Firewire device connected to two computers is not a supported configuration. Additionally, care must be taken to prevent bridging power between the two computers. It is possible to physically harm hardware by connecting two computers to a Firewire peripheral device. When two computers are directly connected this risk is not present because the power leads will be automatically deactivated, however the protections that prevent power from being bridged will not necessarily work if there is a device between the two computers. The Firewire specification allows a computer to supply power to peripheral devices over the Firewire cable (1394-2008 - IEEE Standard for a High-Performance Serial Bus, 2008). There are at least four different cable specifications for IEEE 1394 A/B. Of these one in particular is of interest: the 4-pin cable. This cable omits the power leads from the pinout to create a smaller connector size. In the case that a 6-pin to 6-pin connection needs to be made, the connection can be made safely,

preventing both computers from providing power simultaneously by using a 6-pin male to 4-pin male cable and a 4-pin female to 6-pin male adapter. This is only necessary on one of the computers.

Gaining Access to Pre-Boot RAM Image

As mention above, the contents of RAM in FileVault 2's pre-boot state are of interest, and several attempts were made at acquiring such an image. This included using the *Parallels Desktop 7 for Mac* (2011) virtualization software. It does not appear that the Parallels boot loader supports the EFI volume that FileVault 2 boots from for authenticating users, and as a result, attempts to create a RAM image by booting and virtually suspending the system failed. This method was not tested extensively and may be of interest at a later date.

References

1394-2008 - IEEE Standard for a High-Performance Serial Bus. (2008). IEEE Standards Association.

Apple - OS X Lion - The world's most advanced OS. (n.d.). Retrieved September 17, 2011, from http://www.apple.com/macosx/

Apple - Thunderbolt: Next-generation high-speed I/O technology. (n.d.). Retrieved September 17, 2011, from http://www.apple.com/thunderbolt/

Boileau, A. (2006). *pythonraw1394*.

Dalrymple, J. (2011, July 26). Lion FireWire security issue misleading. Retrieved September 17, 2011, from http://www.loopinsight.com/2011/07/26/lion-firewire-security-issue-misleading/

FireWire - Community Ubuntu Documentation. (2011, May 15). Retrieved September 17, 2011, from https://help.ubuntu.com/community/FireWire

Fleischer, G. (2011, July 12). File Vault in Mac OS X Lion - k3t's weblog. Retrieved September 3, 2011, from http://fleischer.jp/k3t/blog/2011/07/file-vault-in-mac-os-x-lion.html

Garrison, T. (2011, September 7). Cracking MacOS Lion Passwords. Retrieved September 17, 2011, from http://www.frameloss.org/2011/09/05/cracking-macos-lion-passwords/

Graham, R. (2011, February 24). Errata Security: Thunderbolt: Introducing a new way to hack Macs. Retrieved September 17, 2011, from http://erratasec.blogspot.com/2011/02/thunderbolt-introducing-new-way-to-hack.html

Hermann, U. (2008, August 14). Physical memory attacks via Firewire/DMA - Part 1: Overview and Mitigation (Update) | Uwe Hermann. Retrieved September 17, 2011, from http://www.hermann-uwe.de/blog/physical-memory-attacks-via-firewire-dma-part-1-overview-and-mitigation

Homepage | Ubuntu. (n.d.). Retrieved September 17, 2011, from http://www.ubuntu.com/

Jacob. (2011, July 21). Lion Tips, a Collection | The Tech Bulletin. Retrieved September 17, 2011, from http://www.thetechbulletin.net/2011/07/21/lion-tips-a-collection/

Koukoushkina, N. (2011, July 26). Passware Proves Mac OS Lion Insecure Revealing Login Passwords in Minutes. Retrieved from http://www.lostpassword.com/pdf/pr-110726.pdf

Libraw1394. (2009, December 27). Retrieved September 17, 2011, from http://sourceforge.net/projects/libraw1394/

OS X Lion: About FileVault 2. (2011, July 26). Retrieved September 17, 2011, from http://support.apple.com/kb/HT4790

Parallels Desktop 7 for Mac. (2011). Retrieved September 17, 2011, from http://www.parallels.com/products/desktop/

pmset(1) Mac OS X Manual Page. (n.d.).*Mac OS X Developer Library*. Retrieved September 17, 2011, from http://developer.apple.com/library/mac/#documentation/darwin/reference/manpages/man1/pmset.1.html

Schuster, A. (2008, February). Memory analysis: “Acquisition (5): FireWire” - Computer Forensic Blog. Retrieved September 13, 2011, from http://computer.forensikblog.de/en/2008/02/acquisition_5_firewire.html

Technical Note TN2124: Technical Note TN2124. (n.d.). Retrieved September 3, 2011, from http://developer.apple.com/library/mac/#technotes/tn2124/_index.html

Witherden, F. (2010, September 7). Memory Forensics over the IEEE 1394 Interface. Retrieved from https://freddie.witherden.org/pages/ieee-1394-forensics.pdf

Appendix A: Building libforensic1394

The libforensic1394 library was developed by Freddie Witherden (2010) and is available at the following URL: https://freddie.witherden.org/tools/libforensic1394/.

The source code has an explanation of the steps required to build an install the library, and it is not necessary to reproduce that here. When compiling the library on a default installation of Ubuntu Linux 11.04 there are several packages required that are not installed:

- gcc
- g++
- cmake
- linux-headers-generic

```
apt-get install gcc g++ cmake linux-headers-generic
```

Figure 4: Command Required to Install libforensic1394 Prerequisites.

Appendix B: ramdump.py

The following source code was adapted from examples in the paper "*Memory Forensics over the IEEE 1394 Interface*" (Witherden, 2010). It will allow the imaging of RAM on Mac OS, Linux and Windows Targets. Minimal error checking is performed and no validation of the accuracy of the results has been performed. Because cut and pasting Python code can cause difficulties in preserving newlines and indentation, a copy of this program can be downloaded direct at the following URL: http://www.frameloss.org/wp-content/uploads/2011/09/ramdump.py.gz.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# Todd Garrison, 2011 http://frameloss.org/
# Simple script that creates a binary file
# containing the contents of another computer's RAM
# Requires the libforensic1394 library:
# https://freddie.witherden.org/tools/libforensic1394/
#

"""
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""


from forensic1394 import Bus
from time import sleep
from sys import argv

def usage():
      print ""
      print "Usage:"
      print argv[0] + " <MB> <SBP-2>"
      print ""
      print "   MB = Size of dumpfile."
```

```
      print "   SBP-2 = T or F (set to T for linux/win, F for Mac Lion
targets)"
      print ""
      print "   Example dump first 128mb from a Windows Machine:"
      print "      python " + argv[0] + " 128 T"
      print ""
      return
b=Bus()
d = b.devices()[0]
# get opts
try:
      mbs = int(argv[1])
      if mbs < 1:
            exit()
      sbp = str(argv[2])
      if sbp.upper() == "T":
            # Use this for dumping everything *other* *than* MacOS Lion
            b.enable_sbp2()
            sleep (2.0)
            pageSize=d.request_size
            print "Using " + str(pageSize) + " byte pages"
      elif sbp.upper() == "F":
            print "SBP-2 not used, forcing 2048 byte pages."
            pageSize = 2048
      else:
            exit()
except:
      usage()
      raise
      exit()

# Open the first device
print b.devices()
print "Attempting to access device 0: " + b.devices()[0].vendor_name + " " +
b.devices()[0].product_name
d = b.devices()[0]
d.open()
fileobj = open("ramdump.bin", "wb", 1024*1024*64)
try:
        for i in range(0, mbs*1024*1024/pageSize):
            # Skip the first MB
                fileobj.write(d.read(1024*1024 + i*pageSize,pageSize))
      fileobj.close()
      print "Wrote file ramdump.bin."
except IOError:
      print "Something went wrong, sorry."
```

Figure 5. Program Source Code for ramdump.py

Using ramdump.py to Perform RAM Acquisition:

The program shown above is simple to use. It only offers two options: How much RAM in Megabytes to write, and whether to use the SBP-2 mode of Firewire.

Testing showed that Mac OS Lion passwords were recovered within the first Gigabyte of memory.

The SBP-2 mode of Firewire works provides a method of mapping physical memory addresses as opposed to logical memory addresses. This mode of access is disabled on Lion and it is unknown if there is a method of enabling access. Because SBP-2 is not availaon on Mac OS Lion, it is likely that only the first four Gigabytes of RAM can be acquired. To capture against a Windows 7, or Linux 2.6 system it is necessary that SBP-2 is enabled, the limitation in amount of RAM should not exist. Neither of these conditions were tested.

A file named "ramdump.bin" is written during the capture; it is a binary file containing the contents of RAM. The following shows the ramdump.py program being used on a Linux Host to acquire the RAM of a Mac OS Lion Target:

```
tag@ubuntu11:~$ sudo -s
[sudo] password for tag:
root@ubuntu11:~# ./ramdump.py 1024 f
SBP-2 not used, forcing 2048 byte pages.
[<forensic1394.device.Device object at 0x1078250>]
Attempting to access device 0: Apple Computer, Inc. Macintosh
Wrote file ramdump.bin.
root@ubuntu11:~# strings ramdump.bin |grep --after-context=3 managedUser \
>         |grep --after-context=1 password
password
password1
shell
--
password
shell
root@ubuntu11:~#
```

Figure 6: Example usage of ramdump.py to Capture RAM and Retrieve Password

Appendix C: Building and Using pythonraw1394

During the testing, the Ubuntu 8.04.4 Linux distribution was used for testing with pythonraw1394. This operating system implements the required raw1394 Linux kernel module. The site for distributing the pythonraw1394 software was not online at the time of performing the research. In order to gain access to the software the source code was extracted from the Helix 3 bootable forensics distribution and recompiled.

An copy of this library and associated programs as compiled for Ubuntu 8.04.4 is being mirrored at the following URL: http://www.frameloss.org/wp-content/uploads/2011/09/pythonraw1394-1.0.tgz.

Required Libraries and Programs

To get pythonraw1394 working on a default installation of Ubuntu 8.04.4 there are a few programs that need to be installed:

- libraw1394-dev
- libdc1394-dev
- gcc
- g++
- linux-headers-generic
- python-all-dev

This can be accomplished by running the following command:

```
apt-get install libraw1394-dev libdc1394-dev gcc g++ \
        linux-headers-generic python-all-dev
```

Figure 7: Command to Install Required Programs for pythonraw1394.

The header file "/usr/include/libraw1394/raw1394.h" also needs to be modified, by removing all of the lines that contain the word "deprecated". After this the software can be compiled and used.

Once built the memory acquisition can be performed as demonstrated by the following example:

```
tag@tag-ubuntu8:~$ cd ./pythonraw1394/
tag@tag-ubuntu8:~/pythonraw1394$ sudo -s
[sudo] password for tag:
root@tag-ubuntu8:~/pythonraw1394# modprobe raw1394
root@tag-ubuntu8:~/pythonraw1394# ./businfo
Firewire initialized, with 1 ports available:
Enumerating port & node tree...
Port(number=0, generation=2, busid=1023, localid=1, nodeCount=2, name='ohci1394')
Node(number=0, nodeid=0xffc0)
ConfigROM(
 . . . # extraneous information removed in example # . . .
  0 (Immediate Value), 3 (Module Vendor ID): 0xa27 (Apple Computer, Inc.)
 . . . # extraneous information removed in example # . . .
Node(number=1, nodeid=0xffc1)
ConfigROM(
 . . . # extraneous information removed in example # . . .
  0 (Immediate Value), 3 (Module Vendor ID): 0x80d (Toshiba)
 . . . # extraneous information removed in example # . . .
)
root@tag-ubuntu8:~/pythonraw1394# ./1394memimage 0 0 ./Lion-ram.bin -1024M
1394memimage v1.0 Adam Boileau, 2006. <adam@storm.net.nz>
Init firewire, port 0 node 0
Reading 0x3fdd8000 (1046368KiB) at 6578 KiB/s...
1073741824 bytes read
Elapsed time 159.39 seconds
Writing metadata and hashes...
root@tag-ubuntu8:~/pythonraw1394# strings ./Lion-ram.bin |grep --after-context=3 \
> managedUser |grep --after-context=1 password
password
password1
shell
--
password
shell
root@tag-ubuntu8:~/pythonraw1394#
```

Figure 8: Example usage of pythonraw1394 to Capture RAM and Retrieve Password