Global Information Assurance Certification Paper

Practical Security Techniques for Macintosh OS X Laptop Users

David H Hickman

August 13, 2002

GIAC Security Essentials Certification (GSEC) Practical Assignment V 1.4

Abstract

The purpose of this paper is to provide a tutorial on methods to use a Macintosh OS X laptop in a secure manner. The papers in circulation concentrate on the security flaws in the operating system and are aimed at a security professional or Unix administrator. Laptop usage of OS X is common today. A typical laptop user does not have the need to share files, run a web server or be multi-user on their laptop. This tutorial concentrates on disabling those services and introduces methods to work with the two threat vectors that a laptop will most often face, physical and the network. It begins with a system install, which shows the user items that can easily be practiced by a novice. As the tutorial progresses, the built in encryption system is combined with inexpensive USB removable media to provide a place to store critical data such as vpn keys, documents, etc. The final part of this tutorial illustrates the use of the techniques with the combination of ssh tunnels to provide secure email and web surfing with the laptop on an untrusted network.

The information in this document requires that the user skills to be progressive. It begins with tasks that most OS X users will be able to perform. The portion of the document that deals with ssh requires the user to be able to install and run a Unix based system which has pop3 and smtp capabilities. Since OS X appeals the many people in the open source movement, there are increasing numbers of people with these skills.

All of the information presented was tested with OS 10.1.5, and was tested with the beta versions of OS 10.2. The final release of OS 10.2 is August 24, 2002.

Out of the box - New Hardware and Operating System.

As we all know OS X is a FreeBSD derived operating system[1]. This gives Macintosh users a very stable and proven operating system. Like any good tool if used or managed incorrectly, it can become a very unstable and insecure system.

Out of the box, it is a fairly secure system. However, it is a good idea to reinstall the operating system to meet the users specifications. Before the installation is started, the decision of installing OS 9 needs to be made. If there

is a real need for OS 9/ classic, install it first on its own partition. Keep in mind that booting into OS 9 can will allow unrestricted access to all HFS+ file systems.

The OS X installation process is a simple process. An experienced Unix administrator will probably find the install limited but very easy. Partitioning of the hard drive is probably the most complex portion of the install. A novice should probably stick to one partition for classic and one partition for OS X. The user is presented with a choice of file systems, HFS+ or UFS. Until the file system performance increases one will need to install HFS+ as their operating system partition.[2] If the user does Unix development, Apple says "UFS may be preferable for developing UNIX-based applications within Mac OS X."[3]
This can be accomplished by creating an additional UFS partition for the Unix development environment.

Once the system is installed and configured, the first thing that will be noticed is the computer auto logged in as the user that was created in the install. This is bad... Disable it.

-----> Goto AppleMenu - System Preferences - Login - Login Window Tab.

Uncheck "Automatically log in"

If needed: click on the lock to enter the administrator account info to make the necessary changes.

Leave the System Preferences window open and

----> Click on "Show All" - Users

It is now time to create the non-privileged day to day user. This user will take advantage of a BSD security feature for administrative access called the wheel group. The account that created on the original install is an account that is a member of the wheel group. The wheel group is a privileged group that can execute many things such as shutdown, reboot, etc, and only the wheel group has the ability to su to root. This is common on a BSD system but not very common on a Linux system.

How does this help the end user? Unless they are performing a task that requires the gui to be an administrator, here is no reason for a common user to have this level of access, all it can do is lead to problems. The most damage a non-privileged user can do a OS X system is trash their account or anything that the user owns on the system, requiring the deletion of their home directory and possibly some cleaning on the public areas of the system. Most OSX Carbon and Cocoa applications use the OSX security model. So if it needs

administrative access it will prompt for a username and password. If one uses a program that has problems installing, the user will need to log out and log in as the administrator to complete the install.

How to create a non-privileged user:

> Create the user -----> Click New User - Follow directions.
>
> Under the password tab do not click “Allow user to Administer this computer” and do not leave a hint.
>
> Close the window.

Configure the network and run software update from the system preferences. Once the system is up to date, restart as necessary and log in as the regular user.

Section 2 - Laptop Security Concepts

There has been many papers written on OSX and UNIX security. It is a fact that the BSD subsystem in OSX needs a little work.[4] Luckily, most OS X laptop users do not have the same security issues as other Unix based systems. Most Unix based computer are run 24 hours a day, 7 days a week as a workstation, server or other various purposes. Laptop users typically use their computer to perform day to day tasks such as web surfing, checking email, etc. They usually will not have multiple users on the computer and will rarely have the need to serve files from their laptops. If the user does use their laptop as a portable server then they need to harden the machine more like a server, which is out of the scope of this document. Check out reference 4 for details on a more secure server install.

A default OS X is fairly secure for a laptop user. The user needs to verify that they are not running any services or sharing files.

> ----> Run Apple Menu - System Preferences - Network - AppleTalk - Verify that AppleTalk is not enabled on the active network interface.
>
> Go to the Sharing tab in the System Preferences. In the File and Web tab, verify that File and Web sharing is disabled. Also make sure that ftp access is disabled.
>
> From the Application tab, make sure everything is disabled.

Now that the services are disabled, enable the screen saver lock.

----> Apple Menu - System Preferences - Screen Saver - Activation Tab – Click “Use my user account password.”

It is now time to look at the threats that a laptop user will have to face. The most common threat against a laptop is through the physical vector. As consumers we want smaller laptops, which makes them easy to steal. As a result we need to make the laptops harder to steal by watching our environment and using locking mechanisms on the computer. Thomas Palmer does a great job on discussing the various physical threats to a laptop and how to mitigate them.[5] In a nutshell, one needs to keep physical control of the laptop as much as it is possible. When not possible, secure the machine with alarms, encryption, etc. The author of this document subscribes to the school of thought that Mr. Murphey was an optimist and the laptop will eventually get tampered with or even worse, stolen. So the user needs to prepare for the worst.

We all carry insurance on our laptops... Well most of us should. The cost of the laptop replacement is mitigated through insurance. The problem is the data is usually much more valuable. So the age-old concept of backing up the data should be followed. Since there are so many different ways to accomplish this, it is beyond the scope of this document to discuss backup methods. The whole idea is to do pick a method and do it on a timely basis.

Since we are preparing for the worst, we need to take steps to make the criminal recovery of the data as difficult as possible. This can be accomplished through system passwords, removable media and cryptography.

Since OS X is a Unix based operating system, it has a fundamental security system in place. Unfortunately, it can be easily defeated if the system is booted into System 9, booted from the cdrom, or even worse booted into Target mode. Target mode is activated when one holds the “t” key down upon boot. It turns a firewire Macintosh into a very expensive firewire hard drive.[6] This is a nice feature when backing up your laptop but it also makes it a tempting target to someone who sees it sitting unattended and wants to tamper with it.

The easiest way to fix this problem is to run the Apple Open Firmware Password program.[7] This program sets a password on the Open Firmware and also disables all keys but the option key during boot. This also prevents booting into single user mode. Keep in mind that this procedure is easily defeated by the skillful person, but it will stop a casual person from tampering with the laptop. The URL for this program is listed in the program download section of the paper.

At this point the user needs to determine the value and need of their data. Questions that need to be asked are... Do I need this document on my laptop? What can happen to my company or me if this data fell in the wrong hands? If one starts to get paranoid after asking these questions, then a solution needs to be implemented. Two possible solutions that can be used together are

encrypted dmg files and USB “keychain” flash drives.

A dmg file is a mountable image file. Macintosh OS has a nice feature that supports mounting loopback filesystems. In other words, this means that an image file can be mounted and treated as a filesystem. OS X adds the ability to use AES 128 bit encryption on the fly with the dmg file. An encrypted dmg file located on the laptop’s hard drive, is a good place to store documents that need to be encrypted but the user can tolerate the document being lost if the laptop is stolen. The author suggests using a 10 meg documents.dmg file in the users home directory. When there is a need for a document file, the dmg is mounted by double clicking on it. When done with the file, the dmg is unmounted by right-clicking on it and choosing eject (This can also be accomplished by dropping the volume in the trash.)

Here are the quick steps to create a 10 meg encrypted dmg file.

In the finder run to /Applications/Utilities/Disk Copy
-----> Image - New Blank Image

Choose a location and give the file a name.

Under Volume Name: give it the name that it will mount as.

The default size is 10 MB.

The default format is Mac OS Extended.

In Encryption choose AES-128.

The program will then ask you for a password. Give it something long and hard to guess. Do not forget the password.

Uncheck - “Remember password (add to Keychain)”

Any time you mount this volume remember to not allow the keychain to store the password.

We have covered setting an Open Firmware password and building a encrypted dmg file. The final piece to cover is removable media. The current laptop models that Apple ships do not include removable media. Some models have a cd r/w drive, but it can be a chore to use that drive for anything besides backups. Luckily the cost of flash memory has dropped to the point that USB “keychain” flash drives have become affordable. These little drives are the size of a keychain and plug into the USB port on your computer. 64mb drives are readily available for under $100. The drives are easy to use. The Macintosh will auto-mount the drive once it is inserted into the USB port. It will usually come up

as a PC file system. The drive should be reformatted to HFS. This eliminates the ability for a pc user to easily read the drive. This can be accomplished with the Disk Utility program. It is located at /Applications/Utilities/Disk Utility. If there is a need for higher security for the users documents, the user should create an encrypted dmg file on that drive.[8] This will be the method used later in this document for storing the user's ssh keys to have secure email and web surfing.

Section 3 – Secure Email and Web Surfing

Up until now we have covered methods that can be easily implemented by a novice OS X user. Most laptop users use their computers in many places and on foreign networks. There exists a need to access email and if bandwidth permits, surf the web in a secure manner.

The user needs to meet the following criteria.

1. To have root access on a functional unix based computer that can
    a. Send and receive email through smtp.
    b. Has a working pop3 server installed on it.

2. The machine must be reachable from the external Internet through ssh.

3. The network has either static ips or resolvable dynamic dns.

The instructions in the rest of this document refer to a Linux computer running the Debian distribution. The scripts and instructions will need to be modified for the particular flavor of Unix implemented.

Now that we have a way to safely store data away from the computer we can look at a method to securely access email on the road. If the user meets the above criteria, great! Otherwise read on and they might be able to justify an upgrade.

We all use email from time to time. When one is on a wired network they usually do not care what goes out over the wire. But since the user is on a laptop, they can not always trust the network that they are connected to. It is also common to see a OS X user running a wireless 802.11, since all modern Macintoshes are easily equipped with an airport card. It is well known that WEP encryption is easily broken.[9] So we need a way to access email in a secure manner. A method to accomplish this is to have a remote system and ssh into it. If one likes the command line mail programs, great, but most users will want to use the gui based programs like Apple Mail or Eudora. SSH has the abilities to create secure tunnels for pop3 and smtp between the laptop and remote computer. This will allow the laptop to use 127.0.0.1 as its mail servers and a

secure network connection between the laptop and the remote computer is assured. The only requirement of the laptop's network is that ssh and possibly dns is allowed. This method can also be used for secure web surfing for the laptop but bandwidth constraints can quickly become an issue.

In order to redirect ports 25 and 110, root access on both the laptop and the home computer is necessary. This is going to require the creation of dsa key pairs to access the remote system. It is up to user to password protect the keys. It becomes a usability vs. security decision. Earlier in this document an encrypted dmg file was created on the keychain drive. This will make a great place to store the ssh key directories. At this point, the keychain is required for any secure access to the server using ssh keypairs. If the laptop is stolen, the critical private key is not located on the computer. Also if the keychain is lost, the keypair can be revoked on the server. Unless AES has a critical flaw, it would take quite a long time for the encrypted dmg to be cracked. Thus allowing plenty of time to replace the keychain and replace the keys.

The root account must be enabled on the laptop to create the ssh tunnels. Run the terminal program. It is located at /Applications/Utilities/terminal. At the terminal prompt su to the administrative account. It is assumed that the administrative account is administrator.

su to administrator

[localhost:~] reguser% su administrator
Password:
[localhost:/Users/reguser] administrator%

--→ sudo passwd root

[localhost:/Users/reguser] administrator% sudo passwd root

The system will then ask for the administrator password and then ask twice for a root password.

Password:
Changing password for root.
New password:
Retype new password:

Once the root password has been set su to root.

-----> sudo –

[localhost:/Users/reguser] administrator% su -
Password:
[localhost:~] root#

Now the root account should work as expected.

Since the root account is working it is a good idea to verify that the wheel group is doing its job and only letting its members access to root. Open a new terminal as the regular user.

```
[localhost:~] reguser%
```

Type su –

```
[localhost:~] reguser% su -
su: you are not listed in the correct secondary group (wheel) to su root.
[localhost:~] reguser%
```

Two passwords are now required for a user to access root on this laptop.

```
[localhost:~] reguser% su administrator
Password:
[localhost:/Users/reguser] administrator% su -
Password:
[localhost:~] root#
```

The following assumptions are made for the rest of this document. Change the names as needed.

1. You have a usb keychain drive formated as the volume KEYCHAINDRIVE.

2. You have an encrypted dmg file on the keychaindrive named keys.dmg formated as the volume KEYS.

3. You have both of the volumes mounted and they are located at /Volumes/KEYCHAINDRIVE amd /Volumes/KEYS.

4. You have a scripts directory in the path. ~/scripts

5. Your regular user is named reguser and the administrative user is named administrator.

6. The local computer is named OSXLAPTOP

7. The remote computer is called LINUXREMOTE

The root ssh public keypair needs to be built. Su to the root account and follow the example. The commands that need to be entered are underlined.

```
[localhost:~] root# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/var/root/.ssh/id_dsa): Press Enter
Just press enter here unless you know what you are doing.
Enter passphrase (empty for no passphrase):
```

If a password is entered here, it will be required to be entered everytime a script needs to build a tunnel. Keep in mind that the laptop does not have any running services/shares, no multiple users and the ssh keys are going to be stored on the usb drive in an encrypted dmg file. In other words, the laptop and the usb drive has to be stolen, as well as the AES encryption cracked on the dmg file.

Let us continue:

```
Enter same passphrase again:
Your identification has been saved in /var/root/.ssh/id_dsa.
Your public key has been saved in /var/root/.ssh/id_dsa.pub.
The key fingerprint is:
a3:2a:8d:8b:73:74:48:fa:b2:aa:c0:91:c1:79:ae:37 root@localhost
[localhost:~] root#
```

We are going to move the .ssh directory to the encrypted dmg file.

```
[localhost:~] root# mv .ssh /Volumes/KEYS/.rootssh

[localhost:~] root# ln -s /Volumes/KEYS/.rootssh .ssh
```

Now to add the public key to the server so the scripts will be able to ssh in as root.

```
[localhost:~] root# scp .ssh/id_dsa.pub root@LINUXREMOTE:laptop.pub

root@LINUXREMOTE's password:
id_dsa.pub        100% |*****************************|   604      00:00

[localhost:~] root# ssh root@LINUXREMOTE
root@LINUXREMOTE's password:

LINUXREMOTE:~# cat laptop.pub >> .ssh/authorized_keys2

LINUXREMOTE:~# exit
```

Verify that the remote computer can be accessed via a key, since a password was not configured during the key creation, there will be no password prompt.

[localhost:~] root# **ssh root@LINUXREMOTE**

If all goes well you should get a prompt. Delete the public key.

LINUXREMOTE:~# **rm laptop.pub**

LINUXREMOTE:~# **exit**

If there is ever a need to reissue the keys, rebuild the dmg on the keychain with a new password and repeat the above steps to reissue the dsa keys.

In order to build the tunnels, root access is required on the laptop. For convenience the /etc/sudoers file can be configured to allow the user reguser the ability to use ssh with sudo. The author believes this is an acceptable risk, since the laptop does not have any services/shares, is not multi-user and the keys are stored on a physically separate encrypted file. If one feels that this is not an acceptable risk, then skip this next step and install the following scripts as root.

Enable the regular user to sudo ssh.

[localhost:~] root# **visudo**

Add the following line to the end of the file.

reguser ALL=NOPASSWD:/sw/bin/ssh

Save the file. Test this by bringing up a shell for the reguser by typing:

[localhost:~] reguser% **sudo ssh LINUXREMOTE**

The response should be the same as if it we issued as root. Remember if the user is not comfortable with this then install and run the following scripts as root.

Decide where the scripts are going to be installed, this location should be in the path. Open a shell and move to that directory. There are going to be three simple scripts created.

getmail.sh - This is the typically used tunnel script. It accepts the minutes the tunnel is to be open as its argument.

8hr_getmail.sh - This script is the same as getmail.sh, except it does not accept any arguments and defaults to 8 hours for the tunnel time. A typical use for this would be a work day.

killtunnels.sh - This script kills any tunnels you have built by killing the sleep processes on the remote server tied to the ssh tunnels.

getmail.sh

```
#!/bin/sh
sudo ssh -f root@LINUXREMOTE -L 110:LINUXREMOTE:110 tunsleep $1m
sudo ssh -f root@LINUXREMOTE -L 25:LINUXREMOTE:25 tunsleep $1m
```

8hr_getmail.sh

```
#!/bin/sh
sudo ssh -f root@LINUXREMOTE -L 110:LINUXREMOTE:110 tunsleep 480m
sudo ssh -f root@LINUXREMOTE -L 25:LINUXREMOTE:25 tunsleep 480m
```

killtunnels.sh

```
#!/bin/sh
sudo ssh root@LINUXREMOTE killall -9 tunsleep
```

Be sure to make the scripts executable by chmod 755 the files.

In order for the scripts to work, a symlink on the remote machine to the sleep program called tunsleep, has to be made.

```
LINUXREMOTE:~# ln -s /bin/sleep /bin/tunsleep
```

Now that everything looks like everything is present, it is time to test this setup. Run the getmail.sh with 10 as the option. If necessary, su to root and/or enter passwords as needed.

```
[localhost:~/scripts] reguser% getmail.sh 10
[localhost:~/scripts] reguser%
```

Telnet to 127.0.0.1 ports 25 and 110. The responses will vary due to programs installed on the server.

SMTP

```
[localhost:~/scripts] reguser% telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 LINUXREMOTE ESMTP Sendmail 8.12.5/8.12.5/Debian-1; Mon, 12 Aug 2002
15:43:19 -0500; (No UCE/UBE) logging access from: LINUXREMOTE(OK)-
root@LINUXREMOTE [###.###.###.###]
```

QUIT
221 2.0.0 LINUXREMOTE closing connection
Connection closed by foreign host.
[localhost:~/scripts] reguser%

Pop3
[localhost:~/scripts] reguser% telnet 127.0.0.1 110
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
+OK Qpopper (version 4.0.3) at localhost starting. <29889.1029185185@localhost>
quit
+OK Pop server at localhost signing off.
Connection closed by foreign host.
[localhost:~/scripts] reguser%

Test the killtunnels.sh by running it and then telneting to 127.0.0.1 on ports 25 and 110. Everything is working if a connection refused message is returned.

Once everything has been verified configure the email program to use 127.0.0.1 as its incoming and outgoing mail server.

To use this configuration all that needs to be done is to run the getmail.sh script for the desired time period. Run the email program. It is important to remember to run the killtunnels.sh script once the user is ready to terminate the network connection. If forgotten, it may be neccesary to su to root and kill the stray ssh processes.

Earlier it was mentioned that web surfing can also be secured on the laptop's network. If the remote machine has a web proxy such as Squid, http://www.squid-cache.org, installed, tunnels can be easily installed to allow secure web surfing on the laptop's network. The following script assumes the Squid proxy is installed to respond to port 8080 on the remote machine.

The script surf.sh works just like the getmail.sh script.

Surf.sh
```
#!/bin/sh
sudo ssh -f root@LINUXREMOTE -L 8080:LINUXREMOTE:8080 sleep $1m
```

Under the System Preferences program - Network, there is a tab for proxies. Check Web Proxy and Secure Web Proxy. Enter 127.0.0.1 and 8080 as the port. Save the settings. Like the mail scripts, run the surf.sh script with the desired time in minutes. When done run the killtunnels.sh script.

Conclusions

In the history of Unix, Apple's OS X is a new and and significant chapter.

Within one year of its release it has become the largest volume Unix distribution on the planet. Only time will tell what the impact placing a Unix operating system in consumers hands will do. Hopefully this paper has introduced some techniques that can be practiced by laptop consumer and the power users. Apple provides in the operating system, methods to encrypt files and use them as file system. When used properly this can make a laptop a safer place to work. Since OS X is based on open source, there has been an outpouring of software ported to the platform. This allows the use of industry standard protocols such as ssh to create a easy to use tunnels for use on unsecured networks.

Software Downloads

Apple OpenFirmware Password Program
http://www.apple.com/downloads/macosx/apple/openfirmwarepassword.html

References

1. Apple Computer Inc, "The Power of Unix". URL: http://www.apple.com/macosx/jaguar/unix.html (12 August 2002).

2. Jordan - jordan@appletechs.com, "Speeding up OS X for G3 machines", 21 May 2002. URL: http://www.appletechs.com/archives/00000027.html (12 August 2002).

3. Apple Computer Inc, "Mac OS X 10.0: Choosing UFS or Mac OS Extended (HFS Plus) Formatting", 25 June 2002. URL: http://docs.info.apple.com/article.html?artnum=25316 (12 August 2002).

4. Deal, Daniel, "Mac OS X 10.1.4: Security Analysis and Recommendations", 4 June 2002. URL: http://rr.sans.org/mac/osx_analysis.php(12 August 2002).

5. Palmer, Thomas, "Basic Travel Security Revisited", 6 August 2001. URL: http://rr.sans.org/travel/sec_revisited.php (10 August 2002).

6. Apple Computer Inc, "Macintosh: How to Use FireWire Target Disk Mode", 18 July 2002. URL: http://docs.info.apple.com/article.html?artnum=58583(12 August 2002).

7. Apple Computer Inc, "Open Firmware Password 1.0.2: Information and Download", 2 February 2002. URL: http://docs.info.apple.com/article.html?artnum=120095 (12 August 2002).

8. Gilmore, Dan, "Encrypting Removable Mac Drive", 8 August 2002. URL: http://www.siliconvalley.com/mld/siliconvalley/business/columnists/dan_gillmor/ejournal/3825075.htm (13 August 2002).

9. The Schmoo Group, "Air Snort Homepage" URL; http://airsnort.shmoo.com/ (13 August 2002).

10. "OpenSSH Homepage", URL: http://www.openssh.org (10 August 2002)