Securing Mac OS X

A guide to security hardening for Apple Mac OS 10.3

Paul Day, pd(at)csse.uwa.edu.au

November 2004

Abstract: This paper discusses numerous methods of securing Apple Mac OS 10.3 and drawbacks to currently accepted methods of security. It covers both security from a local user's perspective and a network perspective.

Executive Summary

This paper discusses numerous methods of securing Apple Mac OS 10.3 and drawbacks to currently accepted methods of security. It covers both security from a local user's perspective and a network perspective.

The following steps in securing MacOSX are suggested:

Local Security

- Enable the login window, disable auto-login, enable password authentication on wake, regularly change your password and recognise remaining security drawbacks of the login window.
- Enable the screensaver and screensaver locking.
- Set Keychain Access to automatically lock access, change your Keychain password so it doesn't match your login password and recognise remaining security drawbacks of the Keychain.
- Enable automatic Apple Software Update, update Fink and Darwin Ports using cron and enable or use third-party software update utilities.
- Enable FileVault or use encrypted disk images, openssl for file encryption or GnuPG for file encryption and recognise remaining security drawbacks of FileVault and encrypted disk images.
- Enable an OpenFirmware password and recognise its remaining security drawbacks.
- Disable Firewire direct memory access and recognise its security drawbacks.
- Disable Safari's auto-open after download.
- Remove other local users and disable extra system accounts.
- Fix file permissions and scan for susceptible file permissions.
- Remove Classic support.
- Put Bluetooth in invisible mode, turn on authentication, force encryption, disable auto-accept of files, disable file shares, do not pair with unknown devices and recognise inherent security drawbacks of Bluetooth.

Network Security

- Disable and understand usage of services in the Sharing preferences pane, xinetd, hostconfig and System Starter.
- Disable unneeded Directory Access methods.
- Configure a firewall, possibly through the Sharing preferences pane or a third-party application but preferably using a script inserted into System Starter.
- Tweak kernel settings for optimal network security.
- Use and secure SSH by locking down sshd, using SSH keys instead of a password and forward/tunnel X11 and other IP services through SSH.

Contents

1 - Introduction

This document covers numerous methods to harden Mac OS X, from both a local user and network perspective. It is primarily aimed at the single-user Macintosh client machine owned and used by a security conscious user. Its methods can be equally applied to a multi-user machine; however there are numerous additional security risks presented the moment a Mac OS X machine is made multi-user.

1.1 - Background

Apple's MacOS has taken a dramatic change from its predecessors ("MacOS Classic"), introducing numerous parts of FreeBSD, NeXT and the Mach (Darwin) kernel into the MacOS environment.

"Keep others out - With Mac OS X, you may never need to worry about security again."[1]

A default install of Mac OS X is one of the more secure Unix operating systems from a network-security point of view, with no network services open by default. However, there are still numerous drawbacks to its local and network security which can be addressed by the administrator of the machine.

1.2 - Similar papers

It should be noted that there are already a number other papers that already cover the topic of securing Mac OS X. This paper has tried to stand out by:

- Not letting security paranoia result in recommendations with little or no security benefit but a potential inconvenience to the user.
- Including security recommendations and pointing out vulnerabilities that others have not considered or mentioned.
- Simplifying and reducing the amount of background and semi-relevant information.

1.3 - Structure

There are two major sections to this paper: Local Security and Network Security. The first section, Local Security, covers security and hardening methods of the operating system from the perspective of a local user. The user may be either sitting at the machine's console or remotely logged in using a protocol such as Secure Shell (SSH) or Apple Remote Desktop (ARD).

The second section, Network Security, covers security of Mac OS X from a remote or network perspective. It covers services, network protocols and hardening methods that may affect the machine's vulnerability to an external attack.

1.4 - Intended audience

This document is intended for any user or administrator of Mac OS X who is conscious of the security of their machine/s. The audience is expected to have a basic to intermediate knowledge of Mac OS X.

[1] "Top 10 Reasons to Upgrade", Apple web-site, http://www.apple.com.au/Mac OS X/upgrade/reasons.html

They should be able to:

- Navigate around Mac OS X's GUI
- Know how to open and use a local shell (e.g. Terminal.app or xterm)
- Have some basic Unix knowledge. E.g. understand the difference between a normal user and the root user and how to run commands as root within Mac OS X.

1.5 - root usage

By default, the root user account within Mac OS X has its password disabled. Throughout this paper, you are required to run a command "as root". The method of doing this is left up to the reader, but possibilities (in order of considered strength) include:

- `sudo <command>` as a normal admin user.
- `sudo /bin/bash` as a normal admin user and then running the commands.
- Enabling the root account password, using `su` to start a shell as root and then running the commands.

1.6 - Feedback

This paper is a work in progress. Feedback, updates and corrections to the author are welcome and encouraged.

1.7 - Legal

1.8 - Revision control

20/09/04 – Initial version started.

04/11/04 – Initial version finished.

05/11/04 – Minor re-ordering of sub-sections.

07/11/04 – Cleaned up style formatting.

08/11/04 – Re-wording of random parts.

09/11/04 – Added abstract and executive summary.

09/11/04 – Minor formatting.

10/11/04 – Included checking ipfw logging output

11/11/04 – Included notes on finding TCP/UDP ports of services for firewalling

15/11/04 – Numerous spelling corrections. Changed `rm –P` to `srm`.

30/11/04 – Replaced images with original Mac OS X theme and colours.

2 - Local security

The following section covers numerous methods to harden security within Mac OS X from a local user perspective:

- With local physical access to the machine via its console
- With interactive local access to the machine via methods such as Secure Shell (SSH) or Apple Remote Desktop (ARD).

2.1 - The Login Window

The following includes instructions to enable and lock down the GUI login window. By default, Mac OS X automatically logs in rather than forcing the user to authenticate at a login window.

2.1.1 - Enabling and locking down the Login Window

To enable the GUI login window, disable password hints, access to shutdown/restart controls and automatic login you can edit the file `/Library/Preferences/com.apple.loginwindow.plist` as root or use the System Preferences Accounts pane as follows:

```
•    Apple menu -> System Preferences -> Accounts -> Login options ->
  "Display Login Windows as" -> "Name and Password"
•    Uncheck "Automatically log in as:"
•    Check "Hide the Sleep, Restart and Shut Down buttons"
•    Uncheck "Enable fast users switching" if not used
```

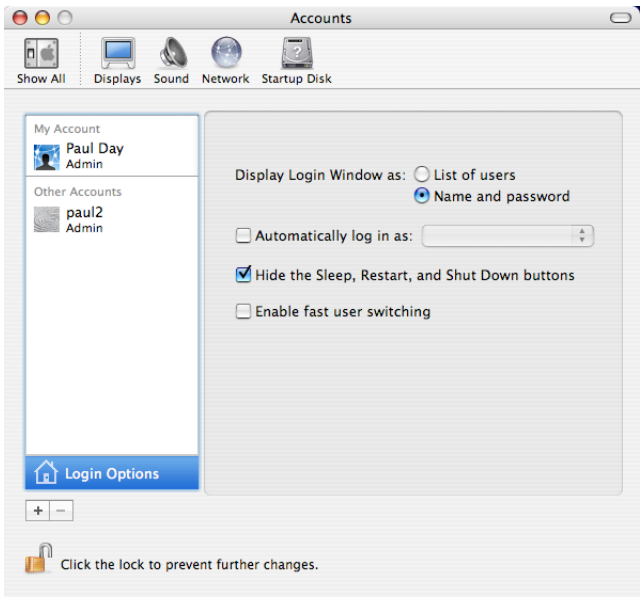

Fig. 1 - Securing Login Window options

Fast user switching is handy on a multi-user machine, however on a single-user machine where it is never used, it is an unneccsary risk (eg, An Apple Remote Desktop root compromise used Fast user switching).

To disable automatic login on a global basis:

- Apple menu -> System Preferences -> Security
- Check “Disable automatic login”

Fig. 2 - Disabling automatic login

To enable a text message to be displayed as part of the login window, you will need to edit the file `/Library/Preferences/com.apple.loginwindow.plist` as root. The file may look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>DisableConsoleAccess</key>
  <true/>
  <key>LoginwindowText</key>
  <string>Authorized users only.</string>
```

Note the `<string>` line below the key `LoginwindowText`. Insert the text you would like to appear in the Login Window here and finish it with the `</string>`.

2.1.2 - Changing passwords

It is good security practice to regularly change your password, especially as the login window does not presently make of `mlock()` or encrypted swap and a user with physical/root access to the machine could potentially get your login password from the swap files.

- Apple Menu -> System Preferences -> Accounts
- Select your username -> Select the Password field
- If asked, type in your current password -> Type in a new password -> verify the new password

2.2 - Screensaver

Mac OS X comes with a built-in screen-saver that includes password locking. This should be enabled to stop someone from using your computer when you step away from it.

To enable the screen-saver:

- Apple menu -> System Preferences -> Desktop & Screensaver -> Screen Saver -> (Select a screen-saver)
- Change "Start screen saver" to 3 minutes

To require a password to exit the screen saver:

- Apple -> System Preferences -> Security
- Check "Require password to wake this computer from sleep or screen saver"

Fig. 3 - Enabling password locking within screen saver

You may also wish to enable an active-corner to disable the screensaver for times you don't want it to come on after inactivity (e.g. while watching a movie) and, more importantly, to instantly load the screensaver:

- Apple menu -> System Preferences -> Desktop & Screensaver -> Screen Saver -> Hot Corners
- Choose a corner, e.g. bottom right -> Disable Screen Saver
- Choose a corner, e.g. top right -> Start Screen Saver

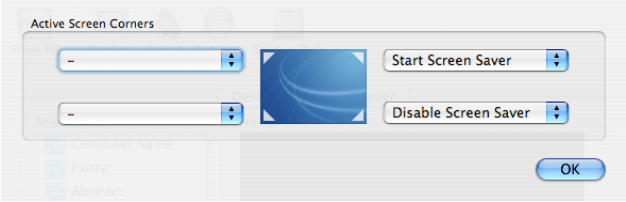

Fig. 4 - Enabling a screen saver corner

2.3 - Keychain

Mac OS X includes a utility for caching commonly used passwords. It should be noted that there is always a risk with caching a password on disk in any form, regardless of the software used.

Keychain stores its passwords on disk in an encrypted form and it is difficult for a non-root user to sniff a password between applications. However, similar to the Login Window, it is possible to get hold of a user's Keychain password with root or physical access to a machine. The best practice is to remember your passwords without storing them.

There are a number of steps you can take to minimise your risk when using Keychain Access. To enable Keychain automatic locking:

```
•    Applications -> Utilities -> Keychain Access -> Edit -> Change
  settings for Keychain "login"
•    Check "Lock after"
•    Change "minutes of inactivity" to 5 minutes
•    Check "Lock when sleeping"
•    Save
```

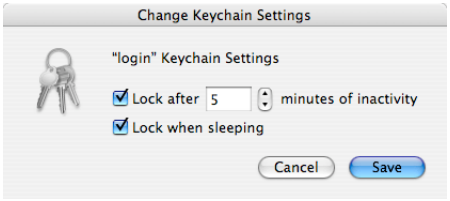

Fig. 5 - Configure Keychain Access security settings

By default, Mac OS X makes your Keychain password the same as your login password. It is good practice to keep each password different:

```
•    Edit -> Change Password for Keychain "login"
•    Type in your current user's login password
•    Type in a new different password twice
•    OK
```

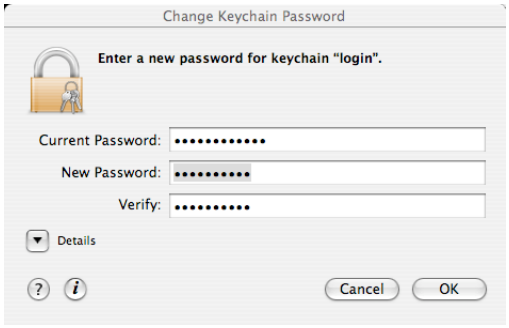

Fig. 6 - Changing your Keychain password

2.4 - Patching

As is generally the case, you should keep your Mac OS X machine regularly patched with the latest software updates, which often include security fixes.

2.4.1 - Apple Software Update

Mac OS X includes an automatic software update tool to patch the majority of Apple applications. Software Update often includes important security updates which should be applied to your machine. The tool automatically checks what updates are available and, with major upgrades, can download patches rather than full installations, to minimize the amount downloaded.

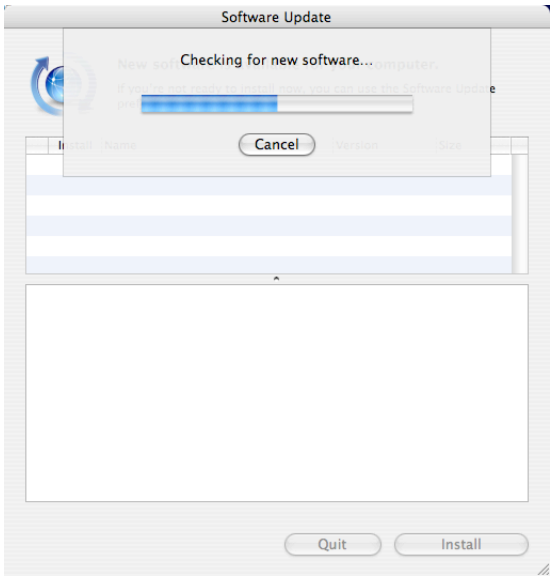

Fig. 7 - Software Update

It is best to configure Software Update to automatically check for updates on a frequent basis:

- Apple Menu -> System Preferences -> Software Updates
- Check "Check for updates"
- Choose "Daily" from drop-down menu.

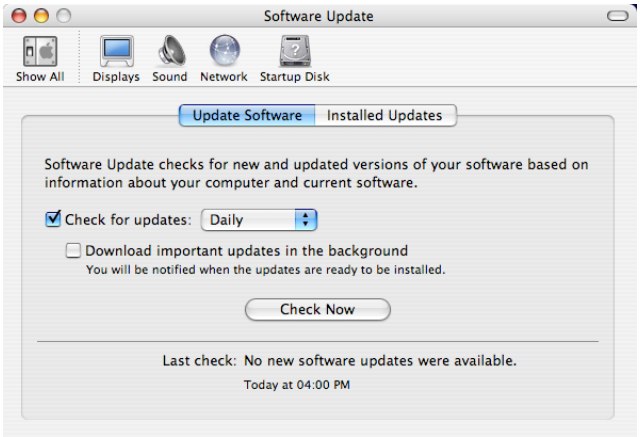

Fig. 8 - Software Update, automatically check for udpates

Your machine will now check with Apple for software updates once a day and notify you when there are new ones ready for download.

Software update can also be run from the command line as root with:

```
/usr/sbin/softwareupdate –ia
```

and scheduled to run with:

```
/usr/sbin/softwareupdate –schedule on
```

2.4.2 - Software update for Fink and Darwin Ports

If you are using the Fink or Darwin Ports packaging systems, you may also wish to have the following in root's daily crontab or in `/etc/daily`:

To update the Fink packaging system:

```
/sw/bin/fink -y selfupdate
/sw/bin/fink -y selfupdate-cvs
/sw/bin/fink -y update-all
/sw/bin/fink -y scanpackages
/sw/bin/fink -y index
/sw/bin/fink -y cleanup
/sw/bin/apt-get -y update
/sw/bin/apt-get -y install fink
/sw/bin/apt-get -y upgrade
/sw/bin/apt-get -y dist-upgrade
/sw/bin/apt-get -y clean
/sw/bin/apt-get -y autoclean
/sw/bin/apt-get -y check
```

To update Darwin Ports:

```
# Change this location to whatever your ports build directory is
cd /opt/darwinports/dports/
/opt/local/bin/port clean
/opt/local/bin/portindex
#Again, change this to be your ports build directory
cd /opt/darwinports/base
cvs -z3 update -dP
./configure
make clean && nice make
make install
# Port upgrade isn't implemented in Darwin Ports yet…
# But here it is in case it gets put in one day
/opt/local/bin/port upgrade
```

Note that Darwin Ports currently does not have a method to actually update already installed packages. The final line is the most important but is not yet implemented. It is included for future reference.

2.4.3 - Other updates

Many other major software packages include their own automatic software update utilities. These may be separate utilities such as Microsoft's AutoUpdate:

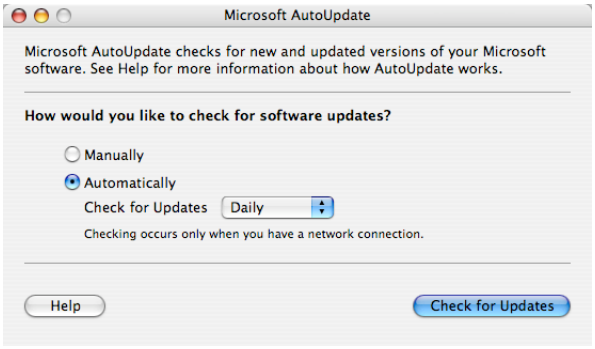

Fig. 9 - Microsoft AutoUpdate

Other packages, such as OmniGraffle, include automatic updating from within the software package itself:

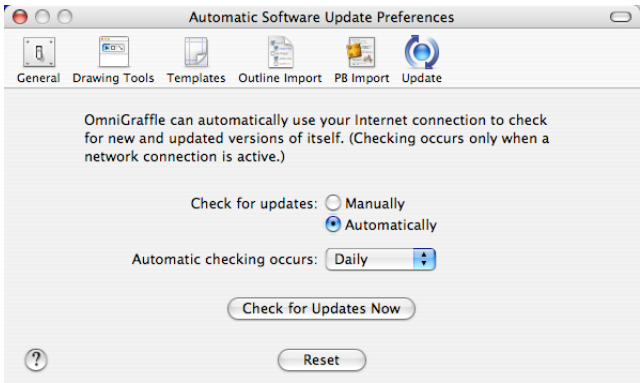

Fig. 10 - OmniGraffle automatic software update

You are encouraged to use these tools where-ever possible, however specifics are beyond the scope of this paper.

2.5 - File encryption

There are number of major ways of encrypting files within Mac OS X. By far, the most secure method is to use GnuPG; however Apple's FileVault and disk images are much more convenient.

2.5.1 - FileVault and encrypted volumes

Apple's FileVault is an implementation of its AES-encrypted volume images that automatically mount as your home directory as you login and decrypt/encrypt data on the fly. Encrypting data on your hard-drive is nothing new but MacOS 10.3 is the first Unix to integrate decryption and mounting seamlessly into the system. From the point-of-view of the user and applications, there is no encryption taking place, beyond a slight performance hit.

To enable FileVault:

- Apple menu -> System Preferences -> Security
- Turn on FileVault

Fig. 11 - Enabling FileVault

Depending on the amount of data in your home directory, it may take a while to convert it into a FileVault. It should be noted here that after encrypting your home directory, it is *not* securely deleted. It is simply unlinked and hence could be recovered.[2]

You may also wish to set a master password for the computer. The master password should be different to your login (and hence FileVault) password and can be used to decrypt your FileVault in the case of password loss.

From a security point of view, keep in mind that due to a lack of `mlock()` in FileVault, an attacker with physical or root access can gain your FileVault password and access to your encrypted files.

2.5.2 - Encrypted AES disk image

Apple's encrypted disk images don't offer the seamless mounting of FileVault, but do still encrypt on the fly as you write to them. To create an encrypted disk image:

- Applications -> Utilities -> Disk Utility
- New Image
- Save as -> Choose a name for the file system and image file name
- Where -> Choose a location to save the image file
- Size -> Choose a maximum size to allow the image to grow to
- Encryption -> Choose AES-128
- Format -> Sparse Disk Image
- Create -> Enter and Verify password
- Check or uncheck "Remember password (add to Keychain)

[2] "Mac OS X Security Issue: FileVault Leaves Unencrypted Data Behind", November 2003, http://www.securemac.com/macosx-filevault-advisory.php

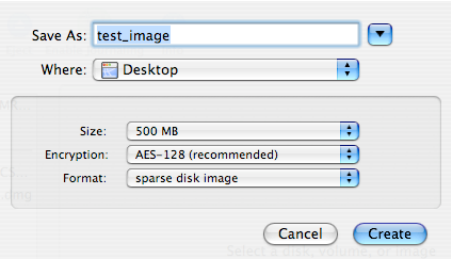

Fig. 12 - Creating an encrypted sparse image

It is no less secure to save a disk image's password in the Keychain as Apple's SecurityAgent (the program that takes the password from the user) suffers from the same vulnerability as Keychain itself.

Once you have created the disk image, you can mount it by double-clicking on it in Finder. It will then mount as `/Volumes/<image file system name>` and an icon will appear on your desktop.

2.5.3 - Openssl encrypted files

Another alternative is using openssl and a password to encrypt a file. Openssl does not employ asymmetric keys (i.e. a private and public key) and allows you to just assign a single password to the encrypted file. However, openssl under Mac OS X may suffer a similar vulnerability to FileVault.

To encrypt a file using openssl and the (128bit) blowfish encryption algorithm:

```
openssl bf –salt -in <plain file> -out <encrypted file>
```

Then securely remove the original file:

```
srm –fm <input file>
```

Finally, decrypt the file back:

```
openssl bf -d -in <encrypted file> -out <plain file>
```

A script to encrypt an entire directory could be:

```
#!/bin/sh
#
# Script to encrypt a dir and securely remove it.

if [ $# -lt 1 ] ; then
  echo "Usage: $0 dir_to_encrypt"
  exit 1
fi

file=`echo $1 | sed s/"\/"//g | sed s/"\."//g`
dir=$1

echo -n "Checking if $dir actually exists... "
if [ -d $dir ] ; then
  echo "Yes."
else
  echo "No. Exiting."
  exit 1
fi
```

```
echo -n "Checking to make sure $file.tar.gz.bf doesn't already exist... "
if [ -e $file.tar.gz.bf ] ; then
  # exists
  echo "Yes. Exiting."
  exit 1
else
  # doesn't exist
  echo "No."
fi

echo -n "Checking to make sure tempfile doesn't already exist... "
if [ -e temp.tar.gz ] ; then
  echo "Yes. Exiting. You need to remove temp.tar.gz."
  exit 1
else
  echo "No."
fi

echo "Tarring up directory..."
tar -zcvf temp.tar.gz $dir
echo "Done."
echo "Encrypting directory..."
openssl bf —salt -in temp.tar.gz -out $file.tar.gz.bf
echo "Done."
echo
echo "Here is what the encrypted archive looks like:"
ls -l $file.tar.gz.bf
echo
echo "Is it safe to securely remove $dir? (y)/n"
read remove
if [ x$remove = xn ] || [ x$remove = xN ]; then
  echo "Ok, exiting without removing it."
  srm -fm temp.tar.gz
  exit 0
else
  echo "Ok, removing $dir securely and exiting..."
  srm -rfm $dir
  srm -fm temp.tar.gz
  echo "Done"
fi

exit
```

Finally, a matching script to decrypt the archive back to a directory in the current working directory:

```
#!/bin/sh
#
# Script to decrypt a tar.gz.bf archive

if [ $# -lt 1 ] ; then
  echo "Usage: $0 archive_to_decrypt"
  exit 1
fi
```

```
file=$1
dir=`echo $1 | cut -d "." -f 1`

echo -n "Checking if $file actually exists... "
if [ -f $file ] ; then
  echo "Yes."
else
  echo "No. Exiting."
  exit 1
fi

echo -n "Checking to make sure $dir doesn't already exist... "
if [ -f $dir ] ; then
  # exists
  echo "Yes. Exiting."
  exit 1
else
  # doesn't exist
  echo "No."
fi

echo -n "Checking to make sure tempfile doesn't already exist... "
if [ -e temp.tar.gz ] ; then
  echo "Yes. Exiting. You need to remove temp.tar.gz."
  exit 1
else
  echo "No."
fi

echo "Decrypting..."
openssl bf -salt -d -in $file -out temp.tar.gz
echo "Untarring..."
tar -zxvf temp.tar.gz
echo "Cleaning up..."
rm temp.tar.gz
echo "All done."
echo

exit
```

2.5.4 - GnuPG encrypted files

Gnu Privacy Guard (an open source version of PGP) allows you to encrypt a file using a public key. You would then be able to decrypt the file at a later date using the private key and the key's passphrase.

Unlike FileVault, GnuPG makes use of `mlock()` and hence doesn't suffer from the same vulnerability. However, it has had a number of its own security concerns.

This section assumes you have already managed to install GnuPG and have created yourself a public/private key-pair. Numerous resources to help you can be found on the web. To then encrypt a file, you would use:

```
gpg –r <your key's name> --encrypt-files <filename>
```

This will create the file `filename.gpg`.

You should securely remove the original plain-text with:

```
srm –fm <filename>
```

Apple's `srm` is included with OS 10.3 (some users may prefer using the GNU fileutils `rm`). Similar to the GNU utility `shred`, `srm` over-writes the file 7 times with random data before unlinking it from the file-system.

To then decrypt the encrypted file:

```
gpg –r <your key's name> --decrypt-files <filename.gpg > filename
```

gpg can also be used with just a symmetric cipher and a single password by using the `-c` switch.

The two scripts in the section above cover en/decrypting entire directories and could be easily modified to use gpg instead of openssl.

2.6 - Configuring Open Firmware password

Configuring an Open Firmware (OF) password on your Mac will disable any boot keys when your machine is booting. This means a user with physical access to the machine is unable to boot the machine into target-disk mode, from CD-ROM or into single-user mode.

The simplest way to set an OF password is to use Apple's utility, which can be found at <http://www.apple.com/downloads/Mac OS X/apple/openfirmwarepassword.html>.

The utility asks for your user password so that it can run `sudo nvram` to set the OF password and then asks for a password to set as the OpenFirmware password:

Fig. 13 - Setting an OpenFirmware password

To set the password yourself directly from OpenFirmware:

```
<power-button>
option-apple-o-f
password
<enter your password>
setenv security-mode command
reset-all
```

You may wish to remove the OpenFirmware password when you are unable to boot the machine properly and need to re-install, back data up using target mode or boot using single-user.

To do this, remove it directly from OpenFirmware:

```
<power-button>
option-apple-o-f
<enter password>
setenv security-mode=none
nvramrc
reset-all
```

In an emergency, the OpenFirmware password can also be removed by changing the amount of RAM and then resetting the PRAM three times (press and hold `option-apple-p-r` while powering up until you hear the machine reboot three times). This is obviously also a potential security risk and for this reason, your machine should be physically secured.

You should also be aware that anyone with root/sudo access to the machine can easily get the OpenFirmware password. Like Sun's OpenBoot, OpenFirmware chooses not to hash the password before placing into non-volatile memory. The hex code of the ASCII password can be revealed with, as root:

```
nvram security-password
```

You can then convert back to ASCII to get the current OpenFirmware password.

2.7 - Disabling FireWire direct memory access

By default, the FireWire protocol gives the FireWire device access to the host's physical memory. This could potentially be used to suck the entire memory contents out of the machine (including your passwords and current working data). Alternatively, an attacker could determine where in memory the screensaver is and insert some random bytes to crash the screensaver, gaining access to the machine.[3]

An undocumented side-affect of enabling an Open Firmware password (see section above) is that it indiretly disables physical memory access for FireWire devices through the IOFireWireFamily kernel driver.

Disabling FireWire DMA appears to have little affect on the performance of FireWire.

2.8 - Disabling single-user logins

A default installation, without an OpenBoot password (or with a subverted OpenBoot password), can be booted into a single-user shell by holding down the "S" key during power-up (or `boot disk –s` from within OpenBoot). This could be used by an attacker with physical access to read your data, add extra accounts or change your passwords.

The following section introduces a method of ensuring a user must enter a password before being presented with a root-user shell as part of a single-user login.

As root:

```
vi /etc/ttys
:1,$s/secure/insecure/g
:wq
```

[3] "FireWire Physical DMA Security – All your memory is belong to the guy with the FireWire cable", Matt Johnston, http://matt.ucc.asn.au/apple/

To generate a password for root to use when logging into a single-user booted system we use openssl:

```
openssl passwd –salt <xy> <password>
```

Replace `<xy>` with two random letters to act as salt for the hashing and `<password>` with the password you want to use for the single-user login. This is completely separate from the local root password, which, if it exists, is stored in the NetInfo database by default.

Now copy the hash that was returned by `openssl` into your paste buffer, open the file `/etc/master.passwd` in `vi` (or your favourite editor) and replace the asterisk (`*`) next to "`root:`" with the hash so the file looks something like:

```
##
nobody:*:-2:-2::0:0:Unprivileged User:/var/empty:/usr/bin/false
root:8d4Gfm/Dhzw6Q:0:0::0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1::0:0:System Services:/var/root:/usr/bin/false
```

Write the file to disk (with `:wq`) and exit `vi`. You will now be asked for the password when booting into single-user.

2.9 - Disable Safari auto-open

Safari, Apple's web-browser, includes a feature where it will automatically launch a number of different file types with their associated application. This could potentially pose a risk with the user unwittingly opening a file without realising it.

To disable the feature:

- Safari -> Preferences... -> General
- Uncheck "Open 'safe' files after downloading

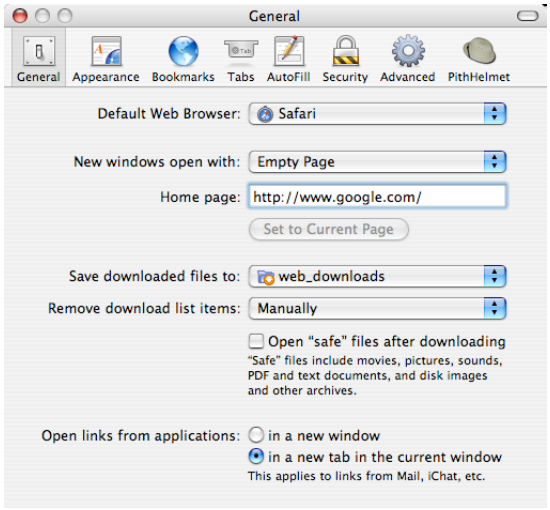

Fig. 14 - Disabling Safari auto-open

2.10 - Removing other local users

There are other vulnerabilities within Apple's Mac OS X 10.3.7 that have not yet been publicly disclosed and hence won't be discussed in this paper. However, it should be noted (although probably obvious) that to ensure the security of your Mac OS X machine, you should not allow any other local users access to your machine, whether by Fast User Switching or SSH.

2.10.1 - Removing normal local users

The cleanest and easiest way to remove extra users is by using the Accounts System Preferences pane:

- Apple menu -> System Preferences -> Accounts
- Select the other account
- Click the minus ("-") button -> Delete Immediately

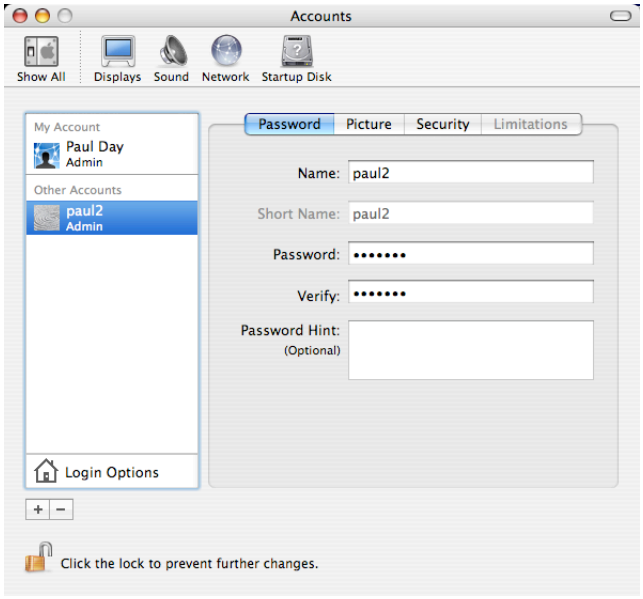

Fig. 15 - Using the Accounts preferences pane to remove extra users

2.10.2 - Checking system user accounts

You may also wish to ensure that no other accounts (not shown in the Accounts preferences pane) have been added by an application installation and left with insecure/default passwords. These could be exploited by an attacker allowing them login to your machine.

To do this you need to make changes within the NetInfo database, either via the GUI or the command line. To remove passwords on extra system accounts using the GUI:

- Applications -> Utilities -> NetInfo Manager -> Domain -> Open -> / -> OK -> / -> users
- Choose a system user -> Ensure it has no "passwd" entry
- If it does have a password entry, click the lock in the bottom left -> authenticate -> select the "passwd" line" -> Delete
- Close the window -> Save -> Update this copy

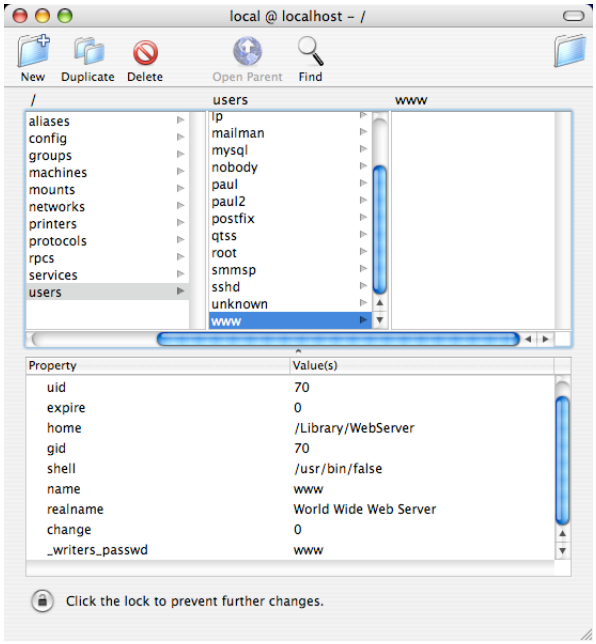

Fig. 16 - Checking for active users in NetInfo Manager

2.11 - Fix file permissions

Over time, permissions and ownership of numerous files may become insecure. This is generally caused by installation of packages put together by non-security-savvy software developers.

To try to correct this situation, it is a good idea to regularly use Apple's Disk Utility to fix file permissions. This can be done by:

- `Applications -> Utilities -> Disk Utility`
- `Select your / disk-partition`
- `First Aid -> Repair Disk Permissions`

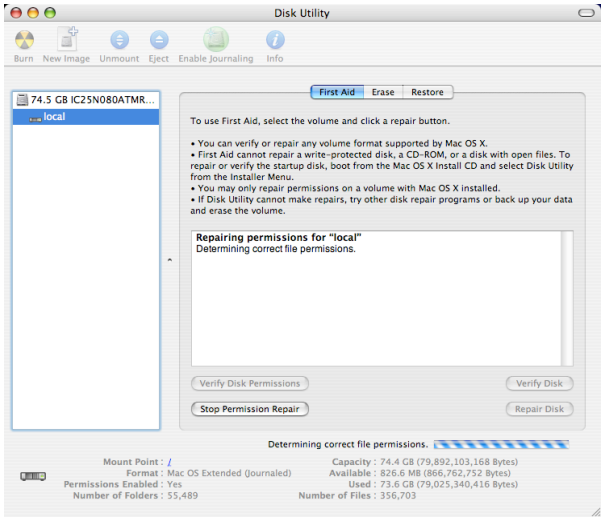

Fig. 17 - Repairing file permissions

It can also be done from the command line as root:

```
/usr/sbin/diskutil repairPermissions /
```

The output may look like:

```
Started verify/repair permissions on disk disk0s3 local
Determining correct file permissions.
We are using special permissions for the file or directory
./System/Library/Filesystems/cd9660.fs/cd9660.util.  New permissions are
33261
Permissions differ on ./private/var/log/install.log, should be -rw-r--r--
, they are -rw-r-----
Owner and group corrected on ./private/var/log/install.log
Permissions corrected on ./private/var/log/install.log
Permissions differ on ./private/var/log/wtmp, should be -rw-r--r-- , they
are -rw-r-----
Owner and group corrected on ./private/var/log/wtmp
Permissions corrected on ./private/var/log/wtmp
The privileges have been verified or repaired on the selected volume
Verify/repair finished permissions on disk disk0s3 local
```

You may choose to add this to root's or the system cron files, e.g. `/etc/weekly.local`.

diskutil is unable to automatically correct all insecure/incorrect permissions for you. To list all files with potentially insecure or strange permissions, run the following commands as root and examine (or redirect) the output:

To list all setuid/gid (binaries that run with a user or group ID of someone other than the user running then, commonly root) files:

```
find / -type f \( -perm -4000 -o -perm -2000 \) \-exec ls -al {} \;
2>/dev/null
```

To list all world writable files:

```
find / -type f \( -perm -2 \) \-exec ls -al {} \; 2>/dev/null
```

To list all world writable directories:

```
find / -type d \( -perm -2 \) \-exec ls -ald {} \; 2>/dev/null
```

To list all un-owned files:

```
find / -nouser -o -nogroup \-exec ls -al {} \; 2>/dev/null
```

Based on the output of these commands, you may choose to change or remove permissions to some files manually. Make sure you are fully aware of the purpose of a file before fiddling with its permissions. Random permission changes may result in an unusable system!

2.12 - Removing Classic

Some users may have chosen to install Mac OS Classic support. Classic provides Mac OS 9 emulation support within Mac OS X, which allows a user to seamlessly run an old Mac OS application on their new Mac OS X machine.

If you're not actually using any Classic applications, it is best to disable and remove Classic support entirely. Run the following commands as root:

```
rm –rf /System/Library/PreferencePanes/Classic.prefPane/
rm –rf '/System/Library/Classic/'
rm –rf '/System/Library/CoreServices/Classic Startup.app/'
rm –rf '/System/Library/UserTemplate/English.lproj/Desktop/ Desktop (Mac
OS 9)/'
rm –rf '/System Folder/'
rm –rf '/Mac OS 9 Files/'
rm –rf '/Applications (Mac OS 9)'
```

2.13 - Securing Bluetooth

Bluetooth is a radio (2.4GHz) data technology that allows a user to wirelessly connect numerous personal devices to allow communication between them. Bluetooth achieves what is sometimes referred to as a Personal Area Network (PAN), allowing you to, for example, have your mobile phone, hands-free kit, PDA and computer all communicating wirelessly.

Unfortunately, Bluetooth has numerous security drawbacks. This section discusses a number of methods to help lock down Bluetooth on your Mac OS X machine. The methods can also be applied to your other, non-Mac OS X, Bluetooth devices (eg, PDA, mobile phone).

2.13.1 - Turn it off

If you're not actively using the Bluetooth connection, you should disable it:

- Apple menu -> System Preferences -> Bluetooth -> Settings
- "Turn Bluetooth Off"

Fig. 18 - Disabling Bluetooth

2.13.2 - Put the device in hidden/invisible mode

Your devices only need to be in "visible" or "discoverable" mode when pairing them with your other Bluetooth devices. Once you have paired devices, you should disable visibility. Paired devices are still able to communicate even when not in discoverable mode.

To make your Mac invisible:

```
•    Apple menu -> System Preferences -> Bluetooth -> Settings
•    Uncheck “Discoverable”
```

Note that invisible/non-discoverable mode does not make your device entirely invisible. It simply makes it harder to find.

2.13.3 - Turn on authentication

Once Bluetooth authentication is on, devices generally need to then use a common password to pair with another device, although there are vulnerabilities in some vendor’s implementation. To turn on password authentication:

```
•    Apple menu -> System Preferences -> Bluetooth -> Settings
•    Check “Require Authentication”
```

2.13.4 - Turn on encryption

Turning on Bluetooth encryption means that the majority of data transmitted between Bluetooth devices is encrypted with a common key. This makes it difficult for a third party to sniff the data or use recorded data in “replay attacks”. To turn on Bluetooth encryption:

```
•    Apple menu -> System Preferences -> Bluetooth -> Settings
•    Check “Require Authentication” -> check “Use Encryption”
```

2.13.5 - Do not allow auto-acceptance of files

It is best to always be asked for confirmation when accepting a file, stopping a dangerous file or Trojan to be automatically uploaded. To do this:

```
•    Apple menu -> System Preferences -> Bluetooth -> File Exchange
•    “When receiving items:” -> Choose “Prompt for each file”
•    “When PIM items are accepted” and “When other items are accepted:” -
  > Choose “Ask”
```

If you never use Bluetooth to push files from another device to your Mac, set it to automatically “`Refuse all`”.

2.13.6 - Disable file shares

If you do not actively share files from the Mac to your other Bluetooth devices, disable all sharing (read-only and read/write) of files:

```
•    Apple menu -> System Preferences -> Bluetooth -> File Exchange
•    Uncheck “Allow other devices to browse files on this computer”
```

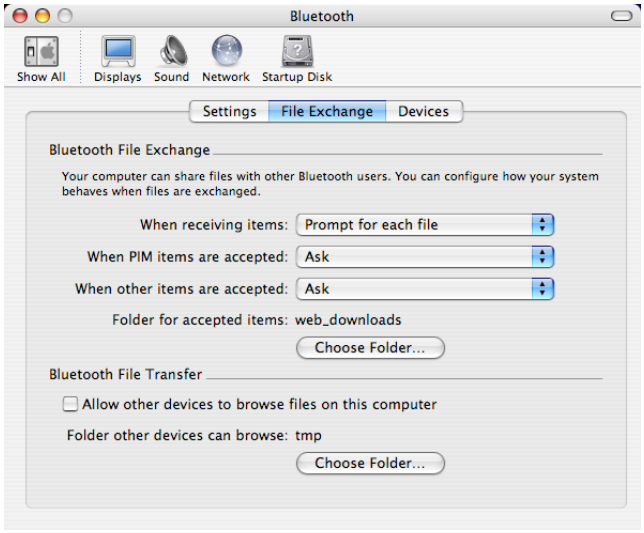

Fig. 19 - Disabling Bluetooth file sharing

2.13.7 - Do not pair with unknown devices

To alleviate the chances of an attacker pairing with your machine, do not pair with an unknown device or allow physical access to your machine to any un-trusted party.

3 - Network security

The following section describes methods of securing Mac OS X from an external, or network, perspective.

3.1 - Disabling services

By default, Mac OS X does not come with any network services enabled. However, some services may have been enabled unwittingly or by installing extra software. This section describes methods of ensuring unknown services are disabled.

3.1.1 - Sharing

Apple's Sharing preference pane is a front-end to xinetd and SystemStarter. It is used to enable and disable a number of common Internet services such as SSH ("Remote Login") and the Apache web-server ("Personal Web Sharing").

By default, Mac OS X 10.3 comes with all the Sharing network services turned off. However, some users may have enabled services unnecessarily.

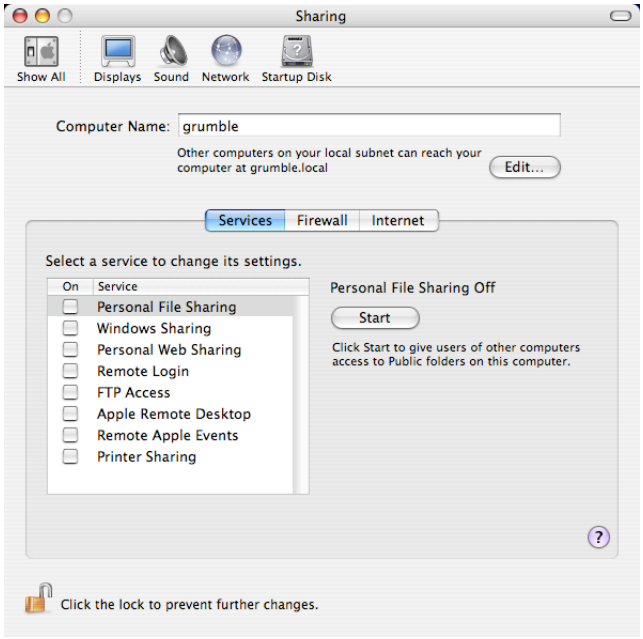

Fig. 20 - The Sharing preferences pane

To disable all services:

- Apple menu -> System Preferences -> Sharing
- Uncheck any checked service

A very basic description of each service can be read by selecting the service and reading the description provided below the Start/Stop button.

The following table shows the Apple service name, normal Internet service name, and software associated with providing the service:

Apple Service	*Internet Service*	*Software*
Personal File Sharing	AFP(overTCP)	AppleFileServer
Windows Sharing	SMB/CIFS	Samba
Personal Web Sharing	HTTP	Apache
Remote Login	SSH	OpenSSH
FTP access	FTP	tnftpd
Apple Remote Desktop	ARD	ARD Helper
Remote Apple Events	EPPC	AEServer
Printer Sharing	LPR/printer	CUPS

Fig. 21 - Table showing Apple Sharing Services

If you must have remote access to your Mac, SSH ("Remote Login") is considered to be one of the more secure methods. SSH can also be used for file transfer by using SCP (Secure Copy) and SFTP (Secure FTP). You can also use it for securely tunnelling other services, for example ARD or VNC. See below for instructions on restricting to particular IPs (either through `xinetd` or ipfw) and securing the default sshd settings.

3.1.2 - inetd

Mac OS X uses the xinetd Internet Super Server for providing a number of IP-based services. Some are enabled/disabled through the Sharing preferences pane while many others (including what are commonly referred to as "useless Unix services") aren't. A list of all services it can provide (from a default installation) can be found in `/etc/xinetd/`.

A listing of any services that have been enabled (either through the Sharing preferences pane or otherwise) can be found by:

```
grep disable /etc/xinetd.d/* | grep no
```

Any services that are not required should be disabled. This can be done by editting the file revealed by the command above and changing the line "disable = no" to "disable = yes". For example, your ssh file may look like:

```
service ssh
{
 disable = yes
 socket_type = stream
 wait = no
 user = root
 server = /usr/libexec/sshd-keygen-wrapper
 server_args = -i
 groups = yes
 flags = REUSE IPv6
 session_create = yes
}
```

Once all unnecessary services have been disabled, you can restart xinetd with:

```
kill –HUP `cat /var/run/xinetd.pid`
```

If you have disabled every service and want to kill off xinetd entirely:

```
kill `cat /var/run/xinetd.pid`
```

If you're choosing to leave a service enabled, you can either restrict what IPs can connect to it within xinetd, or within the ipfw firewall software (see section below). If you decide to restrict it within xinetd, you have the choice of either "allow some, deny rest" or "deny some, allow rest".

As the final line (i.e. above the closing "}") within the xinetd configuration file for the service you're restricting, add in your specifications. To "allow some, deny the rest":

```
only_from = <ip or subnet>, <ip or subnet>, <ip or subnet>
```

Or to "deny some, allow the rest":

```
no_access = <ip or subnet>, <ip or subnet>
```

Insecure services can also be tunnelled with encryption using SSH. Doing so, you leave the service firewalled to the outside world and tunnel a connection into the machine using SSH. OpenSSH itself also has specific user access controls on top of xinetd's and a firewall. See the section below for specifics on securely using SSH.

3.1.3 - OSX hostconfig Services

Mac OS X uses a service start-up system called `SystemStarter`, which replaces the `init` scripts most people would be familiar with from Unix System V variants. It does include a number of features not available in `init`, such as including dependencies in the service, rather than relying on manual ordering within a certain run-level.

A number of `SystemStarter` scripts source the `/etc/hostconfig` file to see if they should start or not. This file contains variables we can set to quickly enable/disable services at boot time.

The following table lists items you may find in `/etc/hostconfig` and a short description of what they're used for:

Service	*Description*
AFPSERVER	Apple File Serving, over TCP for "Personal File Sharing"
AUTHSERVER	Apple NetInfo Authentication service
AUTOMOUNT	Automatic mounting of NFS mount-points (not to be confused with amd)
CUPS	Local printing services
IPFORWARDING	IP routing for other clients
IPV6	IP version 6 protocol support
MAILSERVER	The postfix SMTP mail server
NETINFOSERVER	Bind to a NetInfo server for directory and authentication access
NFSLOCKS	Network File System file locking support
NISDOMAIN	Bind to a NIS domain server for authentication
RPCSERVER	Remote Procedure Call support for numerous Unix services, such as NFS
TIMESYNC	Run NTPd to maintain constant time synchronisation
QTSSERVER	Apple QuickTime Streaming Server modules

WEBSERVER	The Apache web-server for “Personal Web Sharing”
SMBSERVER	Windows file sharing using Samba
DNSSERVER	BIND DNS server
COREDUMPS	Writes a core dump to disk in the case of a kernel panic
VPNSERVER	Apple’s VPN service daemon (LT2P and PPTP)
CRASHREPORTER	Apple’s crash logging service
XGRIDSERVER	Act as a server for Apple’s grid computing software, xgrid
XGRIDAGENT	Act as a client for Apple’s grid computing software, xgrid
ARDAGENT	Apple Remote Desktop server

Fig. 22 - Table showing hostconfig entries and descriptions

Suggested services to enable include CUPS (with “`-YES-`“) to allow printing and NETINFOSERVER (with “`=-AUTOMATIC-`“), which will load netinfod on a stand-alone machine for authentication.

You can enable ntpd for consistent time synchronisation for meaningful logs if you wish. If you choose to disable it, you may wish to add the ntpdate command to /etc/daily or root’s crontab.

```
/usr/sbin/ntpdate -p 8 –u time.asia.apple.com
```

Change “`time.asia.apple.com`” to a local NTP server closer to your location.

3.1.4 - Other OSX Services

Finally, some SystemStarter and mach_init.d scripts don’t actually refer to an entry in /etc/hostconfig to see if they should be run or not. These scripts require manual examination.

SystermStarter and mach_init store their scripts in three locations: `/Library/StartupItems/`, `/System/Library/StartupItems` and `/etc/mach_init.d`.

An example service that starts from StartupItems without examining a /etc/hostconfig entry is the NFS server, nfsiod, starting from `/System/Library/StartupItems/NFS/NFS`. To de-activate it, as root you would edit the script and comment out the line that starts nfsiod:

```
# nfsiod is the NFS asynchronous block I/O daemon, which implements
# NFS read-ahead and write-behind caching on NFS clients.
#nfsiod -n 4
```

Apple’s auto-mount daemon (ADM – not to be confused with the NFS automount service) is used for automatically mounting CDs and image files. It can be disabled in `/System/Libraries/StartupItems/AMD/AMD`. It also checks `/etc/hostconfig` for a “AMDSERVER:=-NO-“, which can be inserted manually (it isn’t included in `/etc/hostconfig` by default).

A default system is unlikely to have any further items that aren’t controlled by `/etc/hostconfig`. However, third-party applications you have installed may. You may wish to examine the contents of each `/System/Library/StartupItems/*/*` and /etc/mac_init.d/* file to determine what services start automatically.

Finally, you can check for any services left running by using, as root:

```
/usr/sbin/lsof | grep LISTEN
```

3.2 - Disabling directory access methods

By default, Mac OS X comes with a number of directory access methods enabled, which could be open to exploitation (e.g. the LDAPv3 service accepts an LDAP server from DHCP by default, which could be faked by a rogue DHCP server on the LAN).

For a stand-alone Mac OS X client, the majority of (or potentially all) services are not required. The following is a table of each of the Directory Access methods and a description of its use:

Directory Access method	*Use*
Active Directory	Windows 2000 domain file sharing and authentication
AppleTalk	Apples legacy protocol for discovering file and print services
BSD Flat File and NIS	`/etc` flat files and Unix Network Information Service (NIS) or Yellow Pages (yp) directory and authentication
LDAPv3	LDAP directory access and authentication
NetInfo	Apple's directory access and authentication
Rendezvous	Apple multicast protocol for file, print, chat, music and other network services
SLP	Service Location Protocol – open standard file and print server discovery
SMB	Windows workgroup file and print sharing/serving

Fig. 23 - Table showing Directory Access methods and their use

To disable services you don't require:

- `Applications -> Utilities -> Directory Access`
- `Uncheck unrequired services`

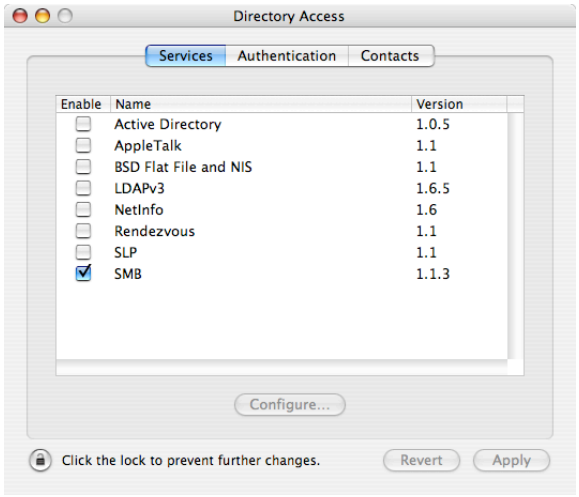

Fig. 24 - Configuring Directory Access

If you need to use LDAP for directory services (such as an enterprise LDAP email address book), ensure you have disabled the DHCP-supplied LDAP Server option:

- `Applications -> Utilities -> Directory Access -> LDAPv3 -> Configure`
- `Uncheck "Use DHCP-supplied LDAP Server"`

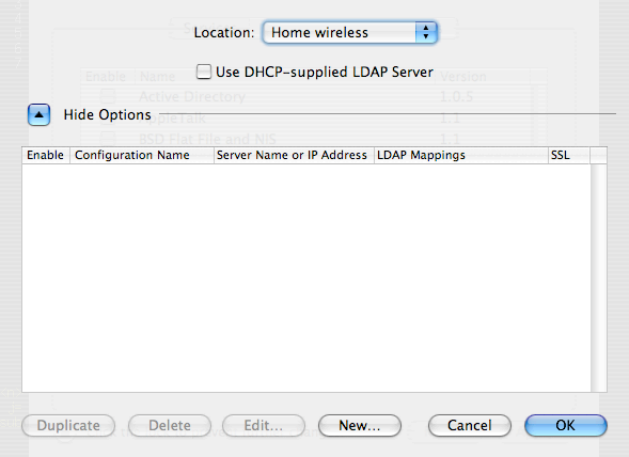

Fig. 25 - Disabling DHCP-supplied LDAP Server

3.3 - Configuring a firewall

By default, Mac OS X does not come with its built-in firewalling software, `ipfw`, enabled. The following section shows how best to enable a firewall on your machine.

3.3.1 - Mac OS X's built-in firewall configuration

Mac OS X includes a method for enabling a default set of firewall rules within the Sharing preferences pane:

- `Apple menu -> System Preferences -> Sharing -> Firewall -> Start`

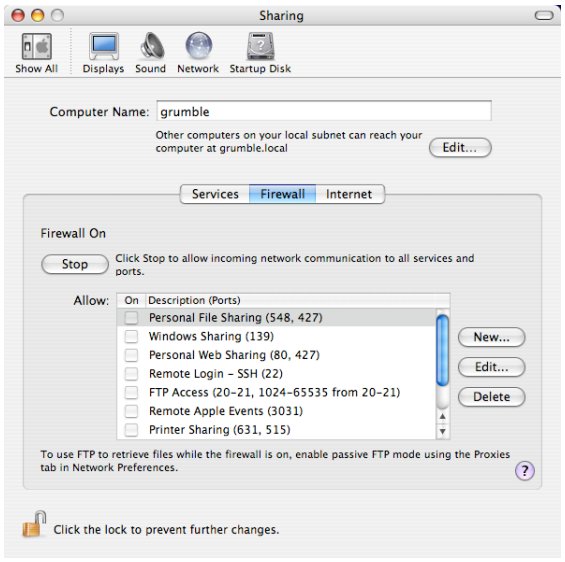

Fig. 26 - Enabling ipfw through System Preferences

By default, the firewall Sharing install isn't is relatively mediocre from a security point of view, but much better than no firewall at all. The following is a list of the rules it adds:

```
02000 allow ip from any to any via lo*
02010 deny ip from 127.0.0.0/8 to any in
02020 deny ip from any to 127.0.0.0/8 in
02030 deny ip from 224.0.0.0/3 to any in
02040 deny tcp from any to 224.0.0.0/3 in
02050 allow tcp from any to any out
02060 allow tcp from any to any established
12190 deny tcp from any to any
65535 allow ip from any to any
```

If you have enabled services, they will automatically be allowed through the firewall from 0/0 (everyone). If you have installed a third-party service, you may need to manually add it firewall:

```
•    New -> Port Name -> Other
•    Port Number, Range or Series: -> Type in the port number/s or range
  of ports the application needs inbound access for
•    Description: -> Type in the name of the service
```

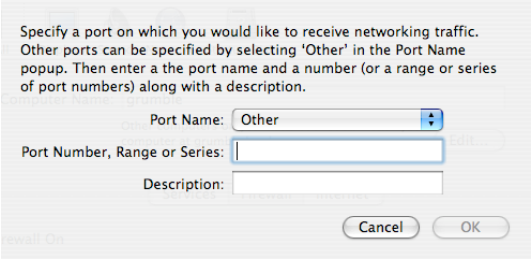

Fig. 27 - Adding an extra service to the firewall

A number of manually-installed services are already listed in the "New" window under the "Port Name" menu.

3.3.2 - Third party applications

There are a number of GUI applications (commercial, shareware and freeware) that can also help you administer the `ipfw` firewall. Discussing each of these is beyond the scope of this article. You will find a number of them by searching your favourite search engine or Mac software web-site.

3.3.3 - Manual firewall configuration

The following section discusses designing and implementing a manual firewall script using `ipfw`.

As root, create the a `SystemStarter` directory and open its parameter's list in your favourite editor:

```
mkdir /Library/StartupItems/firewall
vi /Library/StartupItems/firewall/StartupParameters.plist
```

Insert the following into StartupParameters.plist:

```
{
 Description = "firewall";
 OrderPreference = "None";
 Provides = ("firewall");
 Requires = ("Network");
 Messages =
 {
        start = "Starting firewall";
        stop = "Stopping firewall";
 };
}
```

Next, edit `/System/Library/StartupItems/IPServices/StartupParameters.plist` and insert the following between "Provides" and "Uses":

```
Requires = ("firewall");
```

So that it reads:

```
{
 Description = "Internet services";
 Provides = ("Super Server", "Config Server");
 Requires =("firewall");
 Uses = ("mDNSResponder", "Portmap", "NetworkExtensions");
 OrderPreference = "None";
}
```

This creates a dependency and ensures the firewall has been configured before any network services you've left enabled are loaded. This ensures none of the services are loaded with no protection between them and the outside world.

Finally, open up `/Library/StartupItems/firewall/firewall` in your editor and, at a minimum, insert the following rule-set. You may wish to add extra rules in the appropriate section from the example rules below this section.

```
#!/bin/sh

## Declare variables
# Path to firewalling software
FW="/sbin/ipfw"

## Flush any existing rules from the firewall
$FW -q flush

## Outgoing

# Drop big-brother-like MS VPC7 license checking going out
$FW add deny udp from any to any 21790
# Drop big-brother-like MS Office license checking going out
$FW add deny udp from any to any 2222
# Allow pretty much anything else out
$FW add allow all from any to any out

## Incoming
```

```
# Allow all from/to local loopback interface
$FW add allow all from any to any via lo0
# Then deny anything pretending to come from 127 on other ifs
$FW add deny log all from 127.0.0.0/8 to any in

# Allow relevant outgoing connections back in
# Allow half open TCP back in (although not active ftp)
$FW add allow tcp from any to any established in
# Allow related UDP back in
# DNS - UDP/53
$FW add allow udp from any 53 to any 1024-65535 in
# NTP - UDP/123
$FW add allow udp from any 123 to any 123 in
$FW add allow udp from any 123 to any 1024-65535 in
# DHCP - UDP/67
# DHCP request to server back in to client
$FW add allow udp from any 67 to any 1024-65535 in
# DHCP offer from server in to client
$FW add allow udp from any to any 68 in
# Allow the neccesary ICMP in
# (echo reply, dest unreachable, ttl exceeded, IP header bad)
$FW add allow icmp from any to any icmptypes 0,3,11,12

###
### Insert your custom rules here
###

# Reject IDENT/AUTH with an ICMP reply
$FW add reject tcp from any to any 113 in

# Deny (drop without ICMP) the rest and log to /var/log/system.log
$FW add deny log all from any to any

exit
```

Some rules you may wish to insert could include the following:

```
# Windows/SMB/Samba client access
$FW add allow udp from any 137-139 to any in
$FW add allow udp from any 445 to any in
$FW add allow tcp from any 137-139 to any in
$FW add allow tcp from any 445 to any in

# PPTP VPN client access
# (replace <ip> with your VPN server's IP)
$FW add allow 47 from <ip> to any in

# H.323 client access (NetMeeting and similar)
$FW add allow udp from 0/0 to 0/0 1720 in
$FW add allow tcp from 0/0 to 0/0 1720 in
$FW add allow tcp from 0/0 to 0/0 30000-30010 in
$FW add allow udp from 0/0 to 0/0 5000-5099 in

# XWindows client (server really) in an XNest running in display :1
# (replace <ip> with the Unix box's IP)
$FW add allow tcp from <ip> to any 6001 in
```

```
# XDMCP client (server really) in an XNest running in display :2
# (replace <ip> with the Unix box's IP)
$FW add allow tcp from <ip> to any 6002 in
$FW add allow udp from <ip> 177 to any in

# SSH server
$FW add allow tcp from 0/0 to any 22 in
```

To determine what TCP or UDP port a service uses (so that you can let incoming requests through your firewall), you can check the `/etc/services` file:

```
grep –i <service name> /etc/services
```

3.3.4 - Monitoring ipfw

The final rule in the above script tells ipfw to log any packets hitting the final deny rule before silently dropping them. As root, you can see which packets are being dropped with a command like:

```
/usr/bin/tail –f /var/log/system.log | grep ipfw
```

3.4 - Kernel tweaking

The following section describes a number of kernel variables that should be set to ensure the most secure network settings. Insert the following into `/etc/sysctl.conf` to ensure they're at their most secure:

```
# Verbose firewall logging
net.inet.ip.fw.verbose=1
net.inet.ip.fw.verbose_limit=65535
# ICMP limit
net.inet.icmp.icmplim=1024
# Stop redirects
net.inet.icmp.drop_redirect=1
net.inet.icmp.log_redirect=1
net.inet.ip.redirect=0
# Stop source routing
net.inet.ip.sourceroute=0
net.inet.ip.accept_sourceroute=0
# Stop broadcast ECHO response
net.inet.icmp.bmcastecho=0
# Stop other broadcast probes
net.inet.icmp.maskrepl=0
# TCP delayed ack off
net.inet.tcp.delayed_ack=0
# Turn off forwarding/routing
net.inet.ip.forwarding=0
# Turn on strong/randomized TCP sequencing
net.inet.tcp.strict_rfc1948=1
```

They can also be manually entered at the command line (or in another script) at any time with the following syntax as root:

```
/usr/sbin/sysctl –w <variable>=<setting>
```

3.5 - Securing SSH

SSH (Secure Shell), is provided under Mac OS X using the open-source package OpenSSH. It can be used for a secure remote interactive shell (SSH), secure file transfer (SFTP), secure copy (scp), secure X-windows forwarding (X11Forwarding) and encrypted tunnelling of other IP services.

General SSHd changes

SSHd is highly configurable and can be further locked down from it default settings. Its server configuration file can be found under Mac OS X as `/etc/sshd_config` and the following changes from the default configuration are recommended:

```
#Protocol 2,1
(to)
Protocol 2

#PermitRootLogin yes
(to)
PermitRootLogin no

Subsystem sftp /usr/libexec/sftp-server”
(to)
#Subsystem sftp /usr/libexec/sftp-server
```

3.5.1 - Using SSH keys for authentication

It is considered more secure to login with an SSH key pair than a password. A machine that has already been hacked may have a trojanned sshd binary or authentication services which may be able to give a copy of your password to the attacker. If you have the same password on multiple machines (which is obviously not recommended) they may then login to those other machines using your credentials.

On the other hand, logging in with an SSH key does not allow an attacker to gain your password, even if you are using the same SSH key (with same passphrase) to login to other machines. To disable password authentication:

```
#PasswordAuthentication yes -> PasswordAuthentication no
```

To generate an SSH key pair on your external machine (assuming it runs OpenSSH):

```
user@host:~$ ssh-keygen -b 4096 -t dsa -C "Key for user@host Nov 2004"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/user/.ssh/id_dsa.
Your public key has been saved in /Users/user/.ssh/id_dsa.pub.
The key fingerprint is:
f3:99:d7:05:be:7f:41:42:64:97:b1:e7:d1:41:c9:08 Key for user@host Nov
2004
```

DSA is considerably faster than RSA for key generation and signing, however some argue that DSS has some potential security flaws in its signing process on machines with low random number entropy.

Ensure you add a pass-phrase to your key to protect it if the remote machine is compromised.

Now put `~/.ssh/id_dsa.pub` from the remote machine into `~/.ssh/authorized_keys` on your Mac. Your key will now be automatically used instead of a password for SSH, SCP and SFTP remote access to your machine.

3.5.2 - Forwarding X11 through SSH

Finally, if you have X11 programs that you want to export back to a remote machine, it is recommended that you use SSH's in-built X11 Forwarding in `/etc/sshd_config`:

```
#X11Forwarding no
(to)
X11Forwarding yes
```

From the client machine, you setup the SSH tunnel by typing:

```
ssh –X –l username <remote Mac>
```

3.5.3 - Tunnelling other IP services through SSH

SSH can also be used to tunnel an otherwise insecure protocol through it.

For example, you may wish to use a VNC server runnin on the Mac OS X machine. VNC by itself is not encrypted and it's password is sent plain-text over the network. A somewhat more secure solution to this problem is to leave the SSH port firewalled, tunnel a VNC connection through to the machine and connect to the VNC port on it's loop-back interface.

For example, to make a tunnel through to the remote Mac's TCP port 5900 (commonly VNC), you would do:

```
ssh –N –L 5900:127.0.0.1:5900 <remote Mac>
```

This command binds SSH to port 5900 on the localhost and tunnels it, via SSH, to port 5900 on the remote Mac. You would now point your VNC client to 127.0.0.1 (ie, the localhost's loopback interface) on port 5900 and it will securely connect to the VNC server on your remote Mac.

3.5.4 - Restarting sshd after config changes

Because Mac OS X spawns sshd from `xinetd` rather than as a stand-alone server, there is no need to restart anything. Changes you make to `sshd_config` are read in on the next connection to that service.

4 - Conclusions

With the move from Mac OS Classic's roots to a Unix-based operating system, Apple's Mac OS has undergone massive changes.

While it is one of the more secure Unix operating environments by default, there are a number of methods the administrator of the machine can make use of to harden the environment further.

This document has outlined a number of these methods to secure Mac OS X from a local and network perspective.

5 - References

"A Corsaire White Paper: Securing Mac OS X", Stephen de Vries, Corsaire, 22 June 2004

"Apple Mac OS X v10.3.x "Panther" – Security Configuration Guide", Systems and network Attack Centre (SNAC), National Security Agency, 2004

"BlueBug", Martin Herfurt of Herfurt Salzburg Research Forschungsgesellschaft mbH, Austria, http://agentsmith.salzurgresearch.at/agentsmith_projects_bluebug.html

"Bluesnarfing @ CeBIT 2004 - Detecting and Attacking enabled Cellphones at the Hannover Fairground", Martin Herfurt of Salzburg Research Forschungsgesellschaft mbH, Austria, http://agentsmith.salzburgresearch.at/Downloads/BlueSnarf_CeBIT2004.pdf

"Bluetooth – The Official Bluetooth Membership Site", Bluetooth Special Interest Group, https://www.bluetooth.org/

"DD'S Ultimate Guide to Mac OS Security", http://homepage.mac.com/macbuddy/SecurityGuide.html

"Directory Access Help Pages", Apple Corp., 2004

"Jay Beale's Unix Security Site". Jay Beale, http://www.bastille-linux.org/jay/

"Locking Down mac OS X", Jay Beale, JJB Security Consulting, LLC and GWU Cyber Security Policy & Research Institute, Delivered at Black Hat USA, 2003

"MacSecurity.org", http://www.macsecurity.org/

Personal correspondence, Matt Johnston, October and November 2004

"Scripts for OS X & Darwin", JBCorp, http://users.ez-net.com/~jasonb/secureit.html

"SecureMac.com", http://www.securemac.com/

"Serious flaws in Bluetooth security lead to disclosure of personal data ", Adam Laurie of A.L. Digital Ltd. , http://www.thebunker.net/release-bluestumbler.htm

"War Nibbling: Bluetooth Insecurity", Ollie Whitehouse of @stake, October 2003, http://www.atstake.com/research/reports/acrobat/atstake_war_nibbling.pdf

6 - List of figures

The following lists all figures and their page locations included in this paper.