CORSAIRE

The natural choice for information security solutions

A Corsaire White Paper:

Securing Mac OS X

Author	Stephen de Vries
Document Reference	Securing Mac OS X 10.4 Tiger v1.0.doc
Document Revision	1.0 Released
Date	19 August 2005

Table of Contents

1. Introduction

Mac OS X 10.4 Tiger provides one of the most secure default installations of any desktop operating system. The operating system also includes numerous features and additional tools to help users manage the security of their data. The install follows the accepted best practice of disabling all network services unless explicitly enabled and the default security settings should suit the needs of most users in a workstation setting.

While the default installation provides a relatively secure system it may not always meet organisational security requirements. This guide is aimed at users in environments requiring stronger security controls in an operating system, making full use of the protection features offered by Mac OS X 10.4 It may also be of use to system administrators wishing to enforce an organisation wide desktop security policy.

This guide is an updated version of the guide for Securing Mac OS X (10.3) Panther and covers the new security features offered by Tiger as well as incorporating additional security guidelines that were omitted in the original guide.

2. Using this Guide

This guide covers the security features of Mac OS X 10.4.2 as a multi-user networked system. Most of the console and network based security features are common between Mac OS X and Mac OS X Server, however this guide does not cover Server's additional user, directory and network based security features.

The reader should be familiar with using the UNIX command line and editing plain text configuration files. Most of the operations will require administrator access and it is recommended that each file be backed up before editing it.

In Tiger, some property list (plist) files used to store system and application preferences use a binary format by default. They can be converted to text format using the plutil command or operated on using the Property List Editor application. For example, to edit the file /Library/Preferences/com.apple.loginwindow.plist the following command can be used from the terminal:

```
sudo open /Library/Preferences/com.apple.loginwindow.plist
```

Settings are added using key and value pairs, where the key is case sensitive. When saving the file, the "Save as..." function should be used, since the Save function results in a permission error.

While every effort has been made to test the settings specified in this guide, no guarantee can be made as to their effectiveness or suitability for individual systems.

Any changes to your system are made at your own risk.

2.1 The Common Criteria

The Common Criteria (CC) is an international set of security standards for operating systems that provides a reliable view of the security of those systems. Mac OS X 10.4 has been certified under the

Common Criteria, and Apple has released both an administrative guide[1] and a toolkit[2] in order to bring the default system to a compliant state. While this guide covers many of the same areas as the Common Criteria it was not designed to replace the CC administrative guide, and should not be used as such.

If CC compliance is desired, it is recommended that the administrative guide and toolkit provided by Apple first be used to achieve compliance. This guide should then be consulted for information on additional security controls not covered by the Common Criteria.

3. Summary of Security Hardening

This hardening guide includes the following areas:

1. *General Security Settings* – General security settings and preferences
2. *Patching & Maintenance* – Strategies to perform regular checks for security updates and patches to mitigate risks in the operating system and software in a timely manner.
3. *Physical Access Controls* – Steps to make the Mac OS X host resilient to an attacker with physical console access.
4. *Keychain* – Securing the central authentication repository, or keychain, to reduce the risk of unauthorised access.
5. *File system* – Applying access controls to the file system, including the use of ACLs.
6. *Accountability* – Using logging, process accounting and auditing to maintain accountability.
7. *Data Encryption* – Use of disk-based encryption to prevent unauthorised access to sensitive data, and to provide organisational escrow to that data.
8. *Antivirus Solutions* – Solutions to mitigate the risk of viruses or other malware affecting Mac OS X hosts.
9. *Controlling Administrative Access* – An explanation of administrative privilege under Mac OS X and how best to secure access.
10. *Firewall* – Details of the provided firewall and how to use its full functionality.
11. *Network Services* – Information about the available services and how to deploy them securely.
12. *File Sharing* – How to share files securely.
13. *Intrusion Detection System (IDS)* – Available host based, and network based IDS solutions.

4. Security Hardening Guidelines

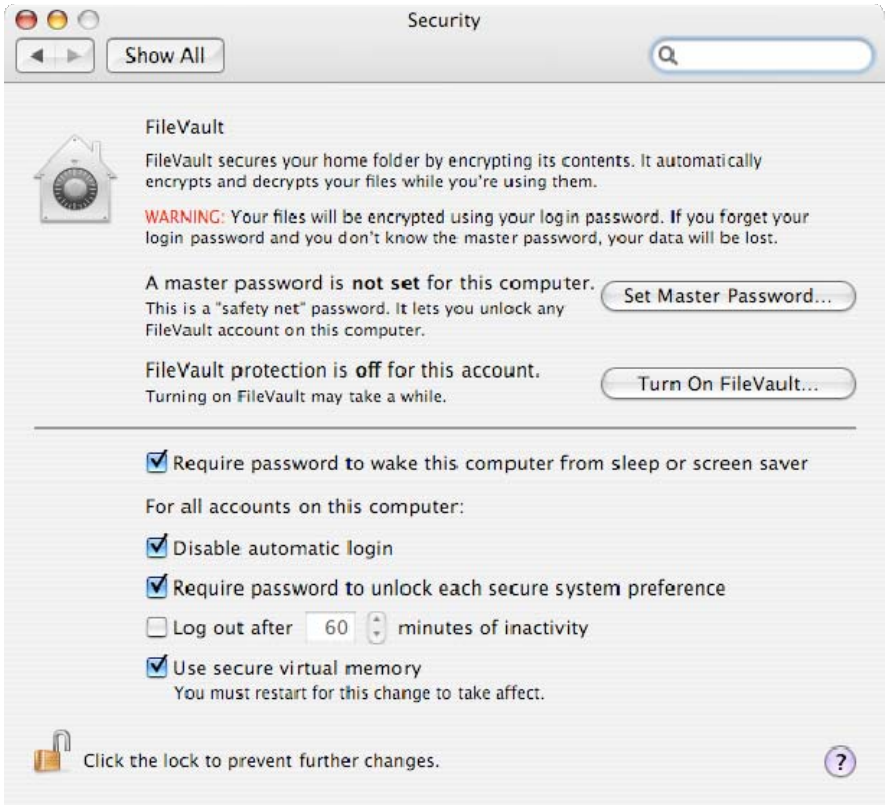
4.1 General Security Settings

4.1.1 Security Preferences

A number of system wide security preferences can be accessed from the *Preferences -> Security pane*.

[1] http://images.apple.com/support/security/commoncriteria/CC_AdminGuide.pdf

[2] http://www.apple.com/downloads/macosx/apple/commoncriteriatoolsfor104.html



A password should be required to wake the computer from sleep or from the screensaver.

Automatic login should be disabled for all users of the system; ensuring that each user must enter their username and password before being granted access.

It is possible to log the user out after a period of inactivity, but in most environments the same security benefits of this feature can be appreciated by using the screensaver password.

Tiger provides "secure virtual memory" which is effectively an encrypted swap space. This ensures that data written to the swap space cannot inadvertently be read. This feature corrects a vulnerability present in previous incarnations of Mac OS X that could allow an attacker, with local access to the system, to access keychain passwords – including the password used for FileVault. It is strongly recommended that this feature be enabled.

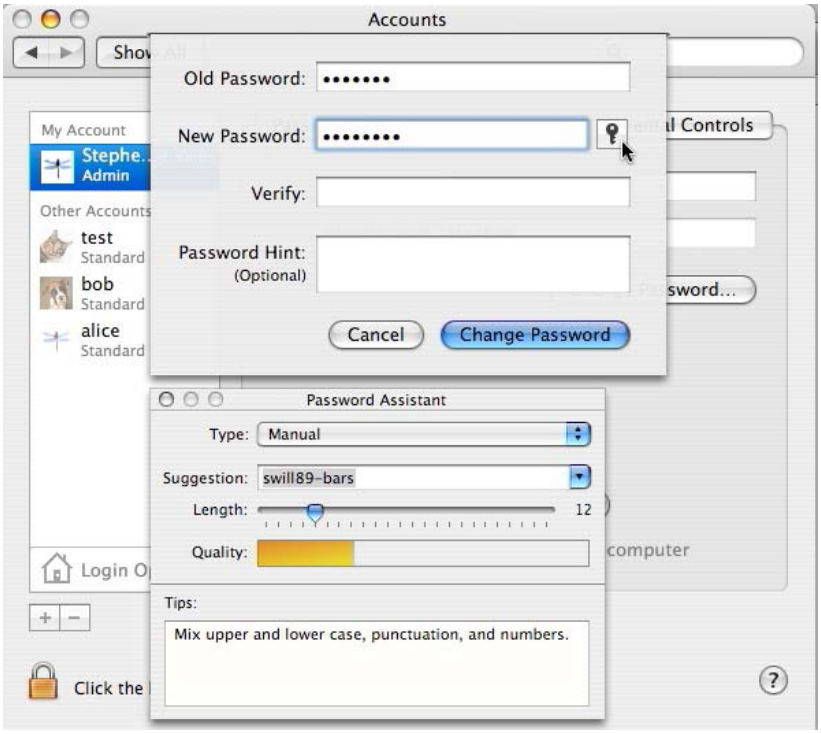
The use of FileVault will be covered in more detail in section 4.7.1.

4.1.2 Passwords

Strong password choice is a fundamental part of a secure system. On a multi-user and/or networked system, it is even more important that users adhere to organisational standards for password choice and password management.

4.1.2.1 Password Assistant

The password assistant is a user-friendly application that assists users in choosing good quality passwords. It can be accessed by clicking the key button that is present on all password choice dialogs. For example, when changing the account password:



The assistant provides feedback to the user as to the quality of the chosen password in the form of a colour-coded bar and also provides tips on improving the password. The assistant can also suggest strong "memorable" passwords for users. It is strongly recommended that the use of this tool be encouraged in user induction and training programs.

4.1.2.2 System wide password policy

Tiger provides the *pwpolicy* command line tool to allow administrators to set user and global password policies, including the ability to specify:

- Password strength in the form of length and character set. It is not possible to determine whether a password is based on a dictionary word or not, nor can the use of special characters or mixed case be enforced.
- Password expiration
- Password reuse
- Maximum number of failed authentication attempts

In the absence of an organisational password policy, the following policy is recommended:

- Users cannot reuse the last 12 passwords

- Passwords must be at least 8 characters in length
- Passwords must contain at least 1 alphabetic and 1 numeric character
- Passwords must be of mixed case and contain at least 1 special character – *this cannot be enforced through pwpolicy.*
- After 5 failed authentication attempts the account is locked out

This can partially be implemented using the following command:

```
sudo pwpolicy -a adminusername -setglobalpolicy "usingHistory=10
minChars=8 requiresAlpha=1 requiresNumeric=1 maxFailedLoginAttempts=5"
```

where *adminusername* is the name of the current administrative user. For more information on the policy options supported by pwpolicy, see the pwpolicy man page.

4.1.3 Disable Core Dumps

When certain errors are encountered, the operating system could create a "core dump", which is a dump of the contents of memory to a file. While this is useful for debugging purposes it is considered a security risk since sensitive information held in memory could be exposed. To prevent the kernel from creating core dumps it is necessary to create (as the super-user) the text file /etc/sysctl.conf so that it contains the following line:

```
kern.coredump=0
```

The change will take effect when the system reboots.

4.2 Patching & Maintenance

4.2.1 Software Update GUI tool

Mac OS X uses the Software Update tool to download and install system and application patches. It is recommended that it be configured to check for new updates daily and can also be configured to perform downloads in the background and notify the user when the update is ready for installation. The preferences for Software Update can be found under *Preferences -> Software Update.*

4.2.2 Command Line

Software updates can also be listed and applied through the command line tool: /usr/sbin/softwareupdate. This makes it possible to install updates in shell scripts, or to invoke it remotely through SSH. For example the following command could be executed to automatically install all required updates and log the output to a file:

```
sudo /usr/sbin/softwareupdate -i -r 2>&1 >> /Library/Logs/auto-softwareupdate.log
```

Caution should be used when adding software update to the crontab, as some required updates need the system to be manually rebooted before taking effect.

4.3 Physical Access Controls

In environments where attackers could gain physical access to the system, it is important that additional security mechanisms are in place to protect the system from unauthorised access. Should an attacker gain physical access to a system, they could boot an alternate operating system and read data stored on the hard drive, or enable a firmware password that will render the system inoperable. The best solution is to control physical access to systems by putting them under lock and key but sometimes this isn't possible; especially for mobile users or desktop users in shared environments.

4.3.1 Open Firmware security

Open Firmware is the BIOS used by Apple systems, and is used to provide low-level control to some parts of the hardware. Open Firmware uses a command line driven interface more similar to that used by Sun Microsystems than the graphical BIOS used by x86 PCs. For the purposes of securing the system, two operations need to be performed in Open Firmware: Setting a password, and changing the security level. These features are only available in Open Firmware version 4.1.7 or later (see: http://docs.info.apple.com/article.html?artnum=106482).

4.3.1.1 Caveats

Open Firmware security can be subverted in a number of ways and therefore does *not* provide complete protection. It does, however, provide more protection than the default settings and will make it more difficult for an attacker to gain access to data.

4.3.1.2 Accessing Open Firmware (OF)

The changes to the firmware described below are made directly from the Open Firmware command line. Apple has released a graphical tool that sets the firmware password, but it does not allow granular control of the security mode[3]. To access the OF command line, the system should be rebooted and Command-Option-O-F held down while the system boots. A screen that similar to the following should be presented:

```
Apple PowerMac,4 4.4.9f1 BootRom build on 11/13/02 at 13:41:09
Copyright 1994-2002 Apple Computer, Inc.
All Rights Reserved

Welcome to Open Firmware, the system time and date is: 02:36:52 01/15/2003
Full security mode.

To continue booting, type "mac-boot" and press return.
To shut down, type "shut-down" and press return.

ok
0>
```

4.3.1.3 Setting a firmware password

From the Open Firmware command line, type:

```
password
```

When prompted, enter and re-enter the chosen password. The password should comply with the organisational security policy.

```
0> password
Enter a new password: ********
Enter password again: ********
Password will be in place on the next boot! Ok
0>
```

[3] This tool can be found on the installation media under */Application/Utilities/Open Firmware Password.app*

Once the password is set, it is necessary to set the security mode to one of the three values: *none*, *command* or *full*, which are described in more detail below:

- **None** – This is the default setting and provides no Open Firmware security protection. Even if a password is set, it has no effect if the security mode is none. It is also possible to set another firmware password without first entering the old one.
- **Command** – This setting causes the system to prompt for a password when any changes to Open Firmware are attempted. It will also require a password when booting from any device besides the default boot device.
- **Full** – This mode requires that a password be entered before booting and before any changes are made to Open Firmware. A password will be required before *every* reboot.

Once the appropriate security mode has been selected, it can be set by typing:

```
setenv security-mode full
```

To save the changes and reboot, type:

```
reset-all
```

4.3.2 Login

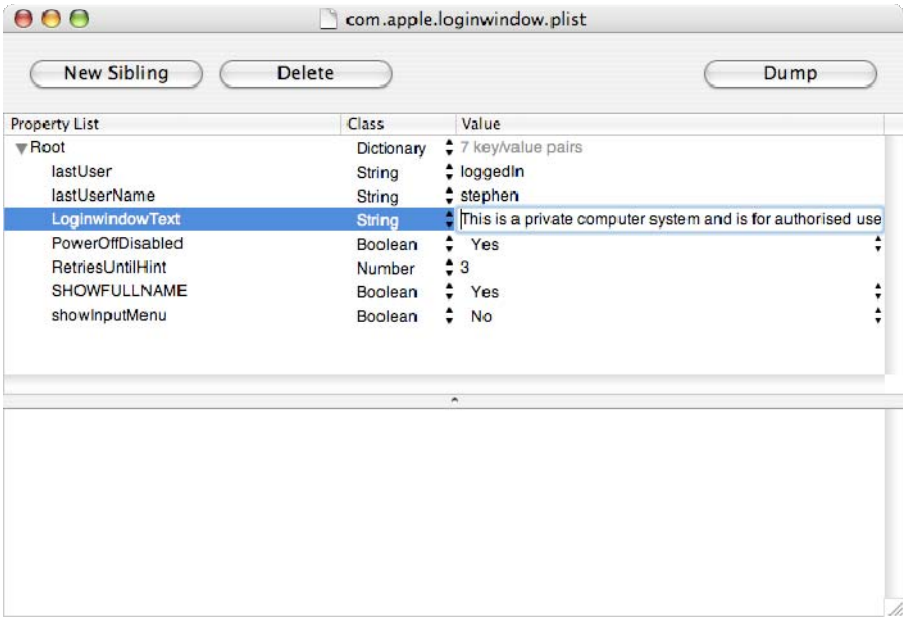
4.3.2.1 Banner

A login banner informs users accessing a system about the system's function, ownership and consequences of unauthorised access. This information should be displayed at all points of entry to the system, usually, login prompts on the desktop, shell logins and ftp access prompts. An appropriate login banner should be defined, after consultation with the organisations legal team. An example login banner could be similar to:

```
This is a private computer system and is for authorised use only.

Any or all use of this system and all files on this system may be intercepted and monitored.

Unauthorised or improper use of this system may result in disciplinary and/or legal action. By continuing to use this system you indicate your awareness of and consent to these terms and conditions of use.

LOG OFF IMMEDIATELY if you are not an authorised user of this system or do not agree to the conditions stated in this warning.
```

The first place that this banner should be displayed is at the desktop login prompt where all local users will see it. To insert a login banner in the Mac OS X login window, edit the file /Library/Preferences/com.apple.loginwindow.plist and insert the key named *LoginwindowText* with the value of the login banner that should be displayed:

```
sudo open /Library/Preferences/com.apple.loginwindow.plist
```



The login window with the new text will be displayed after a reboot.

Note that this message will not be displayed when switching users using the fast user-switching feature.

A number of other services should use the same login banner and it will be useful at this point to create a text file containing the banner in */etc/login_banner*. Since each service that displays the banner will format it according to its own protocol, the length of each line should be less than 80 characters (which is the default for many terminal applications).

4.3.2.2 Displayed usernames

By default, Mac OS X displays a list of usernames with accompanying graphic at the console login prompt. This provides too much information for passing attackers and should be disabled, requiring users to enter their usernames and passwords.

Disable this setting from: *System Preferences -> Accounts -> Login Options -> Display Login Window as: Name and password*.

4.3.2.3 Password hints

Password hints allow users to set a hint if they have forgotten their passwords. While this is a helpful feature for some home users who don't login very often, it is typically not appropriate in a corporate environment, as it increases the risk of an attacker successfully guessing the password.

To disable password hints on the system, open the file /Library/Preferences/com.apple.loginwindow.plist as root, in the Property List Editor application:

```
sudo open /Library/Preferences/com.apple.loginwindow.plist
```

Change the RetriesUntilHint value to 0.

4.3.2.4 Restart, Sleep and Shutdown

The Sleep, Restart and Shutdown buttons are provided on the login screen. Although it is possible to prevent these buttons from being displayed in the login window, it is not currently possible to disable an unauthenticated user from accessing these functions[4].

To prevent the buttons from being displayed deselect the option from *System Preferences -> Accounts -> Login Options -> Show the Restart, Sleep and Shutdown buttons.*

4.3.3 Screensaver

A screensaver should be activated after a short period of inactivity, and should require a password to unlock the workstation. This prevents unauthorised passers-by from accessing an unattended workstation that is logged in. A ten-minute period of inactivity before the screensaver is triggered should suit most organisations. The screensaver can be enabled from *System Preferences -> Desktop & Screensaver*. To enable password protection on the screensaver *Require password to wake this computer from sleep or screensaver* should be selected from the Security pane of System Preferences.

[4] http://www.macosxhints.com/article.php?story=20050603213256255

4.4 Keychain

The Mac OS X Keychain allows users and applications to store and access authentication details in one place. It uses the familiar paradigm of a keychain to store and access private authentication credentials. Users can lock or unlock the keychain with a single password; applications can only access authentication details when the keychain is unlocked. Multiple keychains can be created to group similar authentication credentials.

By default, a keychain called "login" is used to store credentials used by most applications. The password for this keychain is the same as the login password and the Keychain is automatically unlocked when a user logs in and is locked again upon logout.

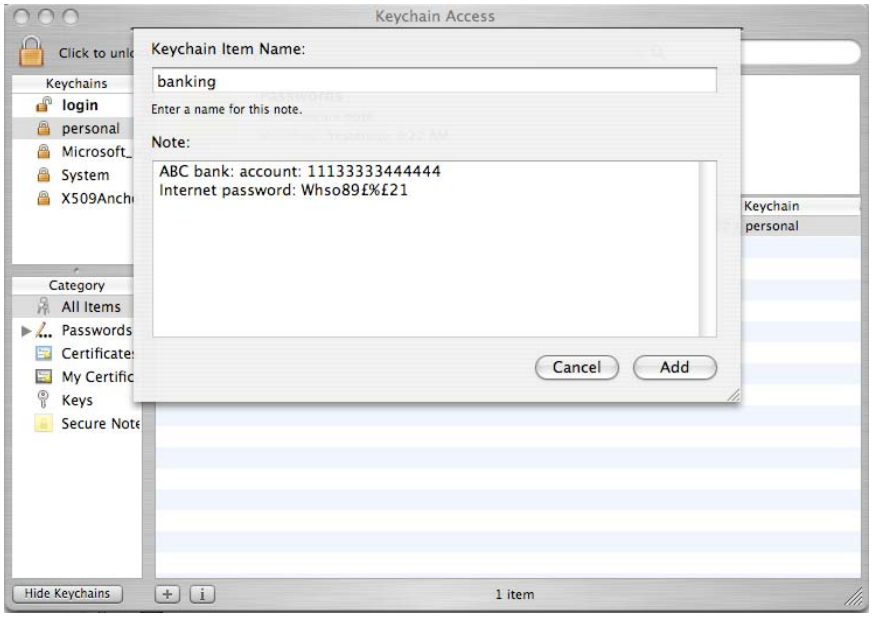
The security of the "login" keychain can be further improved by changing its password to something other than the login password. This will ensure that the keychain has to be explicitly unlocked before any items can be accessed and also prevents keychain items from being accessed if the login credentials are compromised. From the Keychain Access application in Applications -> Utilities, choose *Edit -> Change password for Keychain "login".*

A keychain should also be locked after a period of inactivity and when the system wakes from sleep. These options are accessed from the *Edit -> Change settings for Keychain "login"* menu in the Keychain Access application.

The Keychain Access application also allows individual access controls to be placed on each key in the Keychain. Where keys grant access to particularly sensitive information, it is recommended that the access control be changed to '*Ask for Keychain password*'.

Keychains can also be used to store confidential information that does not belong to a specific application, such as bank, credit card or confidential personal information in the form of a *Secure note item*. These can be added to existing keychains from the *File -> New Secure Note Item.* It is recommended that secure notes be added to a separate keychain that does not share the same password as the login keychain. This ensures that should the confidentiality of the login password be compromised, the secure note will remain secure.

To create a new keychain and add a secure note, choose File -> New Keychain, once a name and password have been chosen, select the keychain and choose *File -> New Secure Note Item:*



From the Keychain Access application's preferences there is an option to *Show status in menu bar*, this allows common keychain function to be easily accessible from the menu bar. It also allows quick access to the screen lock function.

4.5 File System

4.5.1 Default umask

The default umask on Mac OS X systems is 022; this means that by default all users are granted read access to all newly created files. It is recommended that this be changed to 027 so that only users in the same group are permitted automatic read access to new files. Organisational security standards may dictate an alternative umask. Since the umask setting must be specified as a decimal value, the octal value 027 would be 23 in decimal notation. To change the default umask, execute the following command:

```
sudo defaults write /Library/Preferences/.GlobalPreferences NSUmask 23
```

4.5.2 File system defaults

The initial account used to administer the system, as well as any accounts created prior to changing the umask, will allow all users read access to the files in their home folders. It is recommended that this be changed so that only owners and groups have read access to these files. For all users on the system, execute the following command:

```
sudo chmod –R 740 /Users/username
```

Where username is the name of the user. This operation will have to be performed every time a new user is added to the system.

This has the added effect of preventing other users from reading the contents of ~/Public and ~/Sites folders and from writing files to the ~/Public/Drop Box folder. It is recommended that the permissions on these folders be changed on a per user basis.

4.5.3 Access Control Lists

4.5.3.1 Background

One of the more significant changes to Mac OS X, from a security point of view, is the support of Access Control Lists (ACL's) on the file system. This allows for more granular control of file system resources than was previously offered through standard BSD file permissions. The ACLs implemented by Mac OS X 10.4 follow the POSIX standard and are entirely compatible with ACL's on Microsoft Windows systems.

ACL's are stored within the extended attributes of the HFS+ file system. Note that it is not necessary to reformat the file system to take advantage of the extended attributes. Before ACLs can be used, the relevant file system needs to be enabled; this can be done through the fsaclctl command. The command takes the path to the mounted file system as one of the parameters. For example, to enable ACL's on the root file system the following command can be used:

```
sudo fsaclctl –p / -e
```

To disable ACL's on the same file system, use:

```
sudo fsaclctl –p / -d
```

4.5.3.2 Roles, Permissions and Resources

Access controls define the actions (*permissions)* that a *role* can perform on a *resource.* A role is typically a user or a group of users. Permission could be an action like *read* or *write* and a resource could be a file or directory on the file system.

Standard BSD file permissions provide 3 classes of 'roles' (owner, group and other) and 3 classes of *permission* (write, read and execute) that can be used to control access to a resource. While this provides an adequate access control mechanism in some environments it can be cumbersome and unwieldy in many multi-user environments. For example, if user Bob wanted to allow user Alice to read one of his files (but did not want anyone else on the system to read the file), he would have to create a new group, put Alice in the group, and then allow read access to that group – a fairly complex operation for a relatively straightforward policy. In some cases, UNIX permissions simply do not offer the functionality needed to reflect the access control policy for a resource. For example, using UNIX permissions it is not possible for a user to permit someone to delete a file, but not change (write) it – since there is no separate "delete" permission defined. Similarly, it is not possible to allow a user to append to a file, without also allowing them to change the existing contents.

ACL's address these issues by supporting more granular permissions and by allowing these to be assigned to individual roles, instead of only groups of roles. The following tables list the supported permissions and the resources to which they can be applied:

All file system objects

Permission	Description
delete	Delete the item. Deletion may be granted by either this permission on an object or the delete_child right on the containing directory.
Readattr	Read an object's basic attributes. This is implicitly granted if the object can be looked up and not explicitly denied.
Writeattr	Write an object's basic attributes.
Readextattr	Read extended attributes.
Writeextattr	Write extended attributes.
Readsecurity	Read an object's extended security information (ACL). When not explicitly denied, all users who can list the object can read its security attributes.
Writesecurity	Write an object's security information (ownership, mode, ACL).
Chown	Change an object's ownership.

Non-directory file system objects (e.g. files)

Permission	Description
read	Open for reading.
write	Open for writing.
append	Open for writing, but in a fashion that only allows writes into areas of the file not previously written.

Execute	Execute the file as a script or program.

Directories

Permission	Description
list	List entries.
Search	Look up files by name.
add_file	Add a file.
add_subdirectory	Add a subdirectory.
delete_child	Delete a contained object.

A single ACL can be specified per resource, each ACL consists of a list of *ordered* Access Control Entries (ACEs). The semantics of an ACE follow the pattern:

Role [allow or deny] permission resource

This may be familiar to users of network based access control lists or firewall rules. Mac OS X uses the chmod command to manipulate ACLs. For example, the following command could be used to add an entry to an ACL that allows user Alice read access to a resource:

```
chmod +a "alice allow read" ./resource.txt
```

It is important to note that ACLs are acted on before the UNIX permissions and they operate on the principle that the first matching rule is the one that takes effect – in other words, the order of the list is significant. For example, consider the following file, BSD permissions and ACL:

```
-rwxrwxrwx + 1 alice  wheel    6 May  1 15:45 test.txt
 0: user:bob deny read
 1: user:bob allow read
```

The BSD permissions permit all users (including bob) read access to the file. However, entry 0 in the ACL explicitly denies read access to user bob, while the subsequent rule allows him read access. In this case bob will be denied access, since rule 0 is the first matching rule.

4.5.3.3 Inheritance

ACLs set on directories can be inherited by subdirectories and files. The following inheritance permissions are applicable to directories:

Permission	Description
file_inherit	Inherit to files.
Directory_inherit	Inherit to directories.
limit_inherit	This flag is only relevant to entries inherited by subdirectories. It causes the directory_inherit flag to be cleared in the entry that is inherited, preventing further nested subdirectories from also inheriting the entry.
only_inherit	The entry is inherited by created items but is not considered when evaluating the ACE for the given resource.

4.5.3.4 Manipulating ACLs

ACL's can be manipulated using the chmod command and the following modes:

Permission	Description
+a *entry*	Add entries to the list. Note that this mode inserts entries following the canonical form: • Local deny • Local allow • Inherited deny • Inherited allow
-a *entry*	Deletes the first matching entry.
+a# *position entry*	Inserts the entry at exactly 'position' in the list. Note that this does not follow the same canonical form as the +a mode.
=a# *position entry*	The entry at 'position' is replaced by the entry provided.

To view ACL's, Tiger provides an additional option to the ls command: -e, typically used as: ls –le.

4.5.3.5 Example

Consider the following access control requirements for a folder and its contents:

1. All users are permitted to view file names and change into directories.
2. The user 'test' is allowed to read all files in the documents directory, but not in any subdirectories.
3. User 'bob' is permitted to create new files and edit existing files, but is not permitted to create directories.
4. The group 'admin' is permitted to create directories but not to delete them or view their contents.
5. The user 'alice' (who is not an administrator) is allowed to delete directories and files.
6. Anything not explicitly allowed in these rules, is denied.

Note that, strictly speaking, this access control policy is not enforceable on any system that implements discretionary access control, since administrative users always have the ability to change permissions or disable ACL's all together, even if this right is not explicitly granted by the policy.

Since the UNIX permissions will come into effect if there is no matching ACL, for this example, the permissions on the directory will be set to 000, i.e.

```
sudo chmod 000 documents
```

This partially applies the requirement of rule 6. The umask will also play a role here, since newly created files will use the UNIX permissions defined by the mask. A umask of 027 is assumed. The chmod command will be used to manipulate the ACL and add ACEs. The +a method inserts the ACE in canonical form – this could have an adverse effect on the policy since canonical form may not express the policy exactly. Since the policy is clearly defined, the ACE will be added with explicit entry numbers to ensure that it is applied in the intended fashion.

Rule 1: *All users are permitted to view file names and change into directories.*

```
sudo chmod +a  "everyone allow list,search,directory_inherit" documents
```

The 'list' right allows users to view the files in the directory (e.g. ls documents) and the 'search' right allows users to change into the directory. The 'directory_inherit' permission ensures that this ACE is also used on any subdirectories. Note that subdirectories will only inherit the ACE if they are created after the entry was made.

Rule 2: *The user 'test' is allowed to read all files in the documents directory, but not in any subdirectories.*

```
sudo chmod +a# 1 "test allow read,file_inherit,directory_inherit,limit_inherit" documents
```

The 'read' attribute is only applicable to non-directory objects and will therefore only be applied to contained files. The 'limit_inherit' permission ensures that this ACE is only applied to the documents directory itself, and not to any subdirectories.

Rule 3: *User 'bob' is permitted to create new files and edit existing files, but is not permitted to create directories.*

```
sudo chmod +a# 2 "bob allow add_file,directory_inherit" documents
```

The 'add_file' permission is only applicable to directories, and only permits the role to add a file, not another directory.

```
sudo chmod +a# 3 "bob allow write,file_inherit,directory_inherit" documents
```

The write permission is only applicable to files, not directories. The 'file_inherit' permission is used so that the permission is inherited by files. Although bob has write permission, he does not have delete permissions so will not be able to delete files (although he can edit them and clear their contents).

Rule 4: *The group 'admin' is permitted to create directories but not to delete them or view their contents.*

```
sudo chmod +a# 4 "admin allow add_subdirectory,directory_inherit" documents
```

The add_subdirectory command neatly provides the required functionality. However, since subdirectories created by the admin group would use the UNIX permissions set by the umask, they would be writable by the owners. And since the sticky bit was not set on the documents directory, admin users would be able to delete files in the subdirectories they created. The following command explicitly denies this ability.

```
chmod +a# 0 "admin deny delete_child,directory_inherit" documents
```

Note that the rule is inserted at position 0; this follows general good practice when manipulating an ACL: Where the role and permission are specific it is preferable to place the entry towards the top of the ACL. This ensures that the entry is the first matching rule and does not get overridden by any prior 'allow' rules. In this example, the position is irrelevant, since the corresponding 'allow' rule is the UNIX permission which is always evaluated after the ACL.

Rule 5: *The user 'alice' (who is not an administrator) is allowed to delete directories and files.*

```
chmod +a# 5 "alice allow delete_child,directory_inherit" documents
```

The 'delete_child' permission means that all contained objects can be deleted, including directories and files.

4.6 Logging and Auditing

It is important that user accountability be maintained on a secure system. Accountability ensures that actions performed on the system can be traced back to users, this is maintained through: system and application logs, process accounting and auditing.

4.6.1 Logging

The syslogd daemon is used for system and application logging on Mac OS X and its behaviour is controlled by the file: /etc/syslog.conf. Since syslog is a common UNIX logging facility, existing standards within the organisation may stipulate how it should be configured.

The following changes to the default configuration in /etc/syslog.conf are recommended:

Change the line:

```
authpriv.*;remoteauth.crit                              /var/log/secure.log
```

to:

```
authpriv.*;remoteauth.err;auth.err                      /var/log/secure.log
```

This ensures that remote authentication error messages and other authentication error messages are logged to the /var/log/secure.log file.

Additionally, it is recommended that logs be stored on a remote system. These would be useful in the case of a full system compromise where the logs on the system itself become untrustworthy. Remote logging across the network is supported by the syslog facility, but since this is performed through an unencrypted and unauthenticated transport the risk exists that the log information could be compromised in transit or manipulated. Ensure that the network used to transport log information and the remote syslog server are adequately protected.

To enable remote logging of the /var/log/secure.log and /var/log/system.log files to the host 10.0.0.1, add the following lines to the syslog.conf file:

```
authpriv.*;remoteauth.err;auth.err                      @10.0.0.1
*.notice;authpriv,remoteauth,ftp,install.none;kern.debug;mail.crit      @10.0.0.1
```

4.6.2 Process Accounting

Mac OS X supports process accounting, which logs all commands executed by all users. To enable this feature create the folder /var/account

```
sudo mkdir /var/account
```

And then create the empty file /var/account/acct

```
sudo touch /var/account/acct
```

This file will be used to store all the process accounting information. To start accounting on the system issue the command:

```
sudo accton /var/account/acct
```

Process accounting will automatically be started on system start-up. Since accounting has to write to the specified file, if it cannot for any reason (e.g. the file system is full), then accounting will be stopped and will only resume when the problem has been corrected.

4.6.3 Auditing

Mac OS X supports granular auditing of system events. In order to use the auditing subsystem it is necessary to first enable auditing and then install the Common Criteria Tools. A full explanation of

the auditing subsystem is beyond the scope of this document. The reader is advised to consult the Common Criteria Admin Guide[5] for more information.

4.7 Data Encryption

Mac OS X provides built in data encryption features using the AES[6] algorithm with 128 bit keys. This allows users to encrypt data with military strength cryptographic functions.

4.7.1 FileVault

The first of the encryption features is the FileVault function, which encrypts and decrypts a user's entire home folder, protecting the data from unauthorised access. Decryption is performed in real-time as needed and appears seamless to the user.

The user's login password is used to decrypt the encrypted folder. An additional 'master password' may also be set which will be able to decrypt *all* FileVault protected folders on the system. This provides the ability to recover a user's data should they forget their password or leave the organisation, allowing the holder access to all users' encrypted data. Organisational access control policies should dictate whether this is desirable, and who should hold the master password. The ability to decrypt user data is currently a requirement of the UK's RIP Act (http://www.homeoffice.gov.uk/crimpol/crimreduc/regulation/).

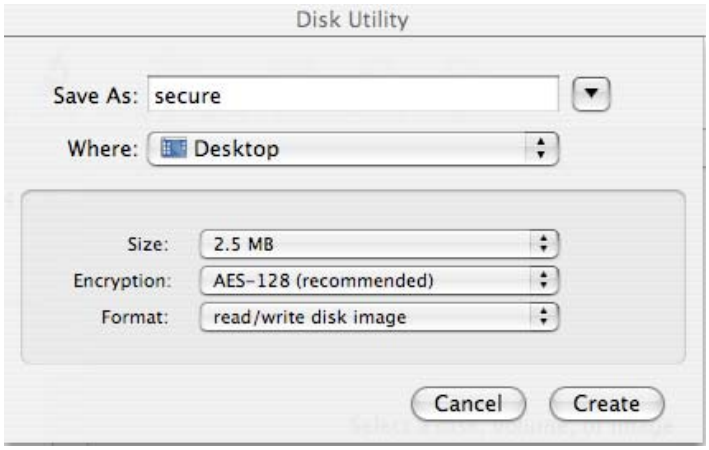
FileVault can be enabled on a per-user basis from the *Security* pane of *System Preferences*.

4.7.2 Disk Utility

The Disk Utility application (*Applications* -> *Utilities* -> *Disk Utility*) can also be used to encrypt data. When a new image is created, 'AES-128 (Recommended)' should be selected as the *encryption setting*. A password will be required to decrypt the image when mounted. This is especially useful for exchanging encrypted data, or for mobile users who wish to store their data on external drives. An option is provided to store the password for an encrypted volume in the user's Keychain, this is useful for most users and should only be ignored by users with the utmost concern for the confidentiality of their data.

[5] http://images.apple.com/support/security/commoncriteria/CC_AdminGuide.pdf

[6] http://csrc.nist.gov/CryptoToolkit/aes/



4.8 Malware

Viruses, trojans and other malware are relatively uncommon on the Mac OS X platform, and as a result currently present a far lower risk than on Windows systems. However, it should be highlighted that their relative absence does not mean that the operating system itself is immune from malware, only that the current combination of a low OS X install base together with the operating system's security features, make the Mac OS X platform more unattractive than other platforms from a virus writer's perspective.

In some organisations, security policies may mandate the use of anti-virus solutions for all desktop systems, regardless of the relative absence of viruses that specifically target Mac OS X. This could help prevent a Mac OS X system from acting as a virus transmission agent in a heterogeneous computing environment.

A number of well-known anti-virus vendors now ship versions of their products for Mac OS X, including:

- McAfee's Virex - http://www.networkassociates.com/us/products/mcafee/antivirus/desktop/virex.htm
- Symantec's Norton AntiVirus - http://www.symantec.com/nav/nav_mac/index.html
- Sophos' Anti-Virus - http://www.sophos.com/products/es/endpoint-server/sav-mac.html
- Intego's VirusBarrier - http://www.intego.com/virusbarrier/

4.9 Controlling Administrative Access

4.9.1 The root user

The default installation of OS X ships with the root user disabled, making it impossible to login or su to root. This reduces the risk of many common attacks traditionally aimed at UNIX operating systems. It is possible to enable the root user but this is strongly discouraged. The status of the root user can be checked from the *Security* menu in the *NetInfo Manager* application.

4.9.2 Administrative user

The access control mechanisms of the system may be further secured by granting administrative rights to only specific users. For each administrative user, there should be two user accounts, one to perform normal user operations, and the other to perform administrative functions. For example, if the user James is a designated administrator he should have a standard system account "james" with no special privileges and an administrative account "admin_james" with administrator rights. This provides accountability where there is more than one administrator on a system. The administrative users should be restricted from logging in to the system from network services using their administrative accounts. This further reduces the risk of the authentication credentials being compromised. To restrict remote access, the configuration of each network service will have to be altered as described in section 4.11.

4.9.3 Sudo

Since the root user is disabled, it is not possible to use the su command to obtain root privileges; instead, Mac OS X makes use of the sudo program. By default Tiger allows all administrative users access to the sudo command and it allows these users to run any program with sudo. In some circumstances, this may contravene system usage policies. In these cases, it is possible to disallow sudo access to the administrator group and instead, enable it on a per user basis.

From the terminal, edit the /etc/sudoers file by typing:

```
sudo visudo
```

Insert a hash (#) character, in front of the line:

```
%admin ALL=(ALL) ALL
```

To allow only the user 'bob' access to sudo add the line:

```
bob       ALL= (ALL) ALL
```

Make sure that at least one user has permissions to run sudo before saving the file! Access controls within the sudoers file can be specified minutely, for example, it is possible to grant the user james access to the file /usr/bin/kill, but only with the privileges of user tim. See the sudo man page for more details on tightening access controls through sudo.

4.10 Firewall

Mac OS X is derived from BSD, and as such features the IPFW firewall. By default, the firewall is disabled (as are most network services), it can be activated and configured from the Firewall tab in the *Sharing* pane of *System Preferences*. The simplistic GUI does not provide access to the full capabilities of the ipfw firewall, however. To enable a more granular control of network traffic it's first necessary to stop the firewall through the GUI and then create a new startup item, which defines the firewall's behaviour.

Create the directory */Library/StartupItems/Firewall*

Create and open the file */Library/StartupItems/Firewall/StartupParameters.plist* in a text editor and insert the following:

```
{
    Description = "Firewall";
```

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

Create and open the file */Library/StartupItems/Firewall/Firewall* in a text editor and add a firewall script that defines the firewall rule base. This is a shell script that will be launched as part of the system start-up process and should contain all the actions to be performed by the firewall, including the definition of a rule base. This should be a direct reflection of your organisation's network access policy. An example of a minimal configuration is included below:

```
#!/bin/sh

# Enable verbose logging
/usr/sbin/sysctl -w net.inet.ip.fw.verbose=1

# DEFINE VARIABLES
# Replace xxx.xxx.xxx.xxx with the external address of your system
EXT="xxx.xxx.xxx.xxx"
FW="/sbin/ipfw"

# DEFINE NETWORKS AND HOSTS
SSH_HOST="192.168.0.1"
# Replace yyy.yyy.yyy.yyy with the address of your primary DNS server
DNS1="yyy.yyy.yyy.yyy"

# START OF FIREWALL RULES
# First flush the firewall rules
$FW -q flush

# Allow all traffic from the loopback interface
$FW add allow all from any to any via lo0
```

```
# Allow the system itself to initiate any connections externally
$FW add allow all from $EXT to any

# Allow established connections to continue
$FW add allow tcp from any to any established

# Allow DNS replies back to the system
$FW add allow udp from $DNS1 53 to $EXT

# Allow certain hosts to SSH in
$FW add allow tcp from $SSH_HOST to $EXT 22

# Deny all other traffic
$FW add 65534 deny log ip from any to any
```

A more detailed example with service-based restrictions can be found at: http://www.novajo.ca/firewall.html (Although the article references Mac OS 10.2, it also applies to 10.4)

Since this file is used system wide, the ownership and permissions of the Firewall directory should be changed as follows:

```
sudo chown –R root:wheel /Library/StartupItems/Firewall
sudo chmod –R 700 /Library/StartupItems/Firewall
```

For the new firewall rules to take effect, either reboot, or manually execute the */Library/StartupItems/Firewall/Firewall* script with super user privileges:

```
sudo /Library/StartupItems/Firewall/Firewall
```

The state of the firewall (enabled or disabled) can be changed by directly changing a kernel parameter:

```
sysctl -w net.inet.ip.fw.enable=0
```

disables the firewall, while

```
sysctl -w net.inet.ip.fw.enable=1
```

enables it.

4.11 Network Services

By default all networking services are disabled, which provides less opportunities for remote attackers. Enabling network services (SSH, Personal Web Sharing, FTP etc.) allows users some form of remote access to the system and should only be permitted if there is an explicit requirement for it.

Tiger uses a new daemon management framework to handle system and daemon start up and control in the form of 'launchd'. Launchd incorporates the functionality of inetd, init, mach_init and SystemStarter and promises to simplify the management of daemons on Mac OS X. In the current version (10.4), either launchd or xinetd are used to control network services, depending on the installation method chosen.

The xinetd daemon is started by launchd. If there are existing xinetd configuration files in the /etc/xinetd.d directory, then xinetd is used to start these services. However, if the configuration files are not present, then launchd is used to start the services, as configured in the /System/Library/LaunchDaemons directory. This can result in inconsistencies between systems, since systems that were upgraded from Panther will use xinetd to control network services, while systems that were cleanly installed will use launchd to control network services.

Since launchd was designed as a general daemon management system, it does not offer as many security features for network services as xinetd does; such as IP based access control and limiting connections. If these security services are required then launchd can either be used together with TCP wrappers or xinetd can be used to handle network services.

4.11.1 Xinetd

The xinetd package used by Tiger provides a number of security features including the ability to finely control access based on source IP address and time of access. It features extensive logging and also provides some protection from denial of service attacks. The configuration of xinetd, and the degree to which access is limited will depend largely on the defined function of the server and the organisation's access control policies.

Global defaults for all services launched through xinetd are configured in the file /etc/xinetd.conf:

```
# man xinetd.conf for more information

defaults
{
        instances               = 60
        log_type                = SYSLOG daemon
        log_on_success          = HOST PID
        log_on_failure          = HOST
        cps                     = 25 30
}
includedir /etc/xinetd.d
```

The first configuration option "instances" specifies the maximum number of connections permitted for any one service. This can be used to protect the system from certain types of denial of service attacks. The value chosen will depend on the available system resources and bandwidth. Another denial of service protection mechanism is to limit the number of connections from a single host. To enable this functionality, add the following line to the configuration (just below "cps = 25 30"):

```
per_source              = 10
```

This will limit each host to only 10 connections per service.

By default, xinetd accepts connections from all hosts; to change this behaviour, add the line:

```
only_from               =
```

to the configuration options. This will *deny* access to all hosts for all services, by default. The hosts permitted to connect to specific services will have to be configured in each of those services' configuration settings.

Xinetd uses separate configuration files to define the behaviour of each service it supports. If the system was upgraded from Panther, then these files can be found in the /etc/xinetd.d directory. If a clean install was performed, then these configuration files will have to be created.

For example, to permit access from 192.168.2.2 and 10.1.1.1 to the SSH (remote login) service, edit the /etc/xinetd.d/ssh file:

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

And add the lines (just after the line 'session_create = yes'):

```
only_from               = 192.168.2.2 10.1.1.1
```

If any hosts are to be explicitly denied access, the no_access setting can be used, for example:

```
no_access               = 10.1.1.9
```

It is also possible to add networks instead of individual IP addresses to either of the configuration settings mentioned above. These can be specified in a number of formats (see the xinetd.conf man page for more details).

Any settings specified in service specific configuration files override the global settings specified in /etc/xinetd.conf. For example, to override the default maximum numbers of instances of SSH, add the line:

```
instances               = 8
```

Note:

By default, Mac OS X enables IP version 6 on all network services started through the xinetd service, but since the IP addresses specified above are in IPv4 notation, they will not work as expected. It is

necessary to disable IPv6 compatibility on all network services where access controls are specified using the IPv4 notation. This is done, by removing the string "IPv6" from the line:

```
flags     = REUSE IPv6
```

from each service's configuration file. Restart the xinetd process for the changes to the configuration files to take effect.

4.11.2 TCP Wrappers

TCP wrappers can be used together with launchd to provide IP based access control for network services. TCP wrappers are already installed on Mac OS X and the launchd configuration files need only be edited to take advantage of the features offered by TCP wrappers (tcpd). Both TCP and UDP based services can be controlled through tcpd. TCP wrappers work by accepting the initial network connection on behalf of the relevant service, logging the connection, applying access control rules and then either denying or permitting access to the service. Besides simple network based access control, tcpd is also capable of performing IDENT checks and launching shell commands in response to a connection attempt. In this paper, only the network access control features will be discussed. This differs from the way the firewall works in that tcpd will always accept the connection and then hand it over to the relevant service, or drop it. From a network perspective (say for example from a portscan), a service controlled with tcpd will always appear to be open, irrespective of whether the service is actually permitted or not.

To enable TCP wrappers on Mac OS X Tiger it is necessary to edit the appropriate launchd configuration files. These files can be found in /SystemLibrary/LaunchDaemons. For example, the default configuration file for the SSH (Remote login) daemon is /System/Library/LaunchDaemons/ssh.plist, the relevant section of the default file is:

```
<key>Program</key>
<string>/usr/libexec/sshd-keygen-wrapper</string>
<key>ProgramArguments</key>
<array>
     <string>/usr/sbin/sshd</string>
     <string>-i</string>
</array>
```

This tells launchd to run the program /usr/libexec/sshd-keygen-wrapper with the arguments /usr/sbin/sshd and –i. To enable TCP wrappers, the tcpd daemon must be called instead of /usr/libexec/sshd-keygen-wrapper and it must be passed the name of the daemon to launch and all its relevant command line arguments:

```
<key>Program</key>
<string>/usr/libexec/tcpd</string>
<key>ProgramArguments</key>
<array>
     <string>/usr/sbin/sshd</string>
     <string>-i</string>
</array>
```

Notice that in this example, tcpd calls the sshd daemon directly and does not make use of the sshd-keygen-wrapper script. It is necessary to ensure that the required ssh keys have already been created before using this method.

To enable the changes, unload and then load the configuration script:

```
sudo launchctl unload /System/Library/LaunchDaemons/ssh.plist
sudo launchctl load /System/Library/LaunchDaemons/ssh.plist
```

To control access, tcpd makes use of two files, /etc/hosts.allow and /etc/hosts.deny that specify pattern matching rules to allow or deny access respectively. The access control algorithm, works in the following manner:

1. If a matching rule is found in /etc/hosts.allow, then access is granted;
2. Otherwise, if a matching rule is found in /etc/hosts.deny then access is denied; or
3. If no matching rule is found then **access will be granted.**

The pattern matching rules take the form:

daemon_list : client_list [: shell_command]

Where daemon_list is a list of daemons, such as ftpd, sshd, smbd or wildcards, list elements are comma or space separated.

Similarly, client_list is a list of client addresses, hostnames or wildcards. Optionally, a shell_command can be specified.

Typical wildcards used to control access are:

- ALL – Matches anything, applies to both daemon names and client names; and
- LOCAL – Matches all local names, i.e. those without dots in their names;For example, to change;

For a complete list of wildcards, see the hosts_access(5) man page. A typical example configuration follows.

The ‘allow by default’ access control policy should be changed to a ‘deny by default’ policy, this can be done by editing the /etc/hosts.deny file and adding the following line:

```
ALL:    ALL
```

To specifically allow the host 10.1.1.1 access to the ftp service, add the following line to the /etc/hosts.allow file:

```
ftpd:   10.1.1.1
```

It is also possible to use the ‘EXCEPT’ operator to specify an exception to the rule. To allow all systems in the example.com domain access to SSH, except the host nossh.example.com, add the following line to /etc/hosts.allow

```
sshd: .example.com EXCEPT nossh.example.com
```

To allow the system 172.20.1.1 access to all services protected by tcpd:

```
ALL:    172.20.1.1
```

For more information on using TCP wrappers and its more advanced functionality see the tcpd man page.

4.11.3 SSH

OS X includes the OpenSSH suite of tools to provide encrypted remote shell, copy and tunnel access to a system. It is recommended that, where possible, SSH access to a system be permitted from certain hosts only. This reduces the risk of an attacker using compromised authentication details to access a system remotely or launching brute force attacks against the SSH service.

4.11.3.1 SSH Options

The default installation of OpenSSH allows both SSH version 1 and version 2 connections. Version 1 is known to suffer from security vulnerabilities, and it is strongly recommended that only version 2 be used. To disable version 1 connections, edit the file /etc/sshd_config and change the line that starts with:

```
#Protocol 2, 1
```

to:

```
Protocol 2
```

As is common with most configuration files on UNIX systems, the # character is used to specify a comment. When it is removed, the directive behind it becomes active. The settings specified in the rest of this section are appropriate for version 2 of the protocol only.

By default, a security feature called 'privilege separation' is enabled. This ensures that any buffer overflow bugs will only grant attackers access to an unprivileged user (sshd).

4.11.3.2 Authentication

OpenSSH has a number of authentication options including username and password, challenge-response, Kerberos and public key authentication. Password and public key authentication, being the most popular implementations, are discussed below.

By default OS X 10.4 permits users to authenticate with their username and password or by creating a public-private key pair and placing their public key in the ~/.authorized_keys file. Using public key authentication provides a degree of protection from brute force attacks, since users without the correct key will not be permitted to authenticate. However, when creating the key pair, it is possible for a user to specify a blank password for their private key which means that they could login to the host without entering any form of password when connecting from that particular client (or by using that private key). This could be considered a security risk in certain environments since it means that if the client host is compromised then the attacker will be able to login to the server without authentication. If public keys are used, then users should be reminded that the password used to protect the private key should comply with organisational password standards and must not be blank. To disable public key authentication, edit the /etc/sshd_config file and add the following option:

```
PubkeyAuthentication no
```

4.11.3.3 Cryptographic settings

The symmetric cipher used to encrypt the SSH session is chosen by the client from the set of ciphers made available by the server. In the default installation, these are:

- aes128-cbc
- 3des-cbc
- blowfish-cbc
- cast128-cbc

- arcfour
- aes192-cbc
- aes256-cbc
- aes128-ctr
- aes192-ctr
- aes256-ctr

The organisation wide security policy may dictate which ciphers are acceptable for remote login. To change the list of available ciphers, edit the /etc/sshd_config configuration file and change the 'Ciphers' setting to list the permitted ciphers. The ciphers should be listed on the same line and comma separated. For example, to only permit the AES algorithms with 192 or higher key length use:

```
Ciphers aes192-cbc,aes256-cbc,aes192-ctr,aes256-ctr
```

SSH also uses a message authentication code for data integrity protection. By default the following MACs are provided by the server:

- hmac-md5
- hmac-sha1
- hmac-ripemd160
- hmac-sha1-96
- hmac-md5-96

To restrict this to only approved MACs edit the 'MACs' setting in the configuration file, for example:

```
MACs hmac-sha1-96,hmac-ripemd160
```

4.11.3.4 Login banner

SSH should be configured to display a login banner before a user authenticates to the service. As discussed in section 4.3.2.1, a text file containing the login banner should be created in /etc/login_banner. To display the login banner to users, edit the /etc/sshd_config file and replace the line:

```
#Banner /some/path
```

With:

```
Banner /etc/login_banner
```

The sshd service will have to be restarted before the changes take effect. This banner will be displayed to all users attempting to access the system through SSH.

4.11.3.5 User access controls

By default, all users who have local accounts on the system are permitted to login through SSH. This is often not necessary and only provides attackers with more avenues for attack. Users with permission to SSH to the systems should be clearly defined and SSH configured to only allow access to those users. To enable this, edit /etc/sshd_config and add the following line towards the top of the file:

```
AllowUsers username1 username2
```

replacing 'username1' and 'username2' with the name of the users who should be granted access. Once the AllowUsers directive is enabled, only those users will be granted access to the system, it is not necessary to explicitly deny all other user with the DenyUsers directive – and explicitly permitting access is a better control strategy than explicitly denying it.

The AllowUsers directive can be further restricted by using the format: user@host, which will only allow access to a specific user from a specific host, for example:

```
AllowUsers glyn@192.168.5.3
```

Will allow glyn access from only the 192.168.5.3 host, and deny him access if he attempts to connect from anywhere else. For more information on controlling the access control mechanisms in SSH see the man pages for sshd_config.

Note

Five different ways of controlling access to SSH based on the source IP address have been discussed, namely:

- by creating a firewall rule;

- by controlling access through the global options of /etc/xinetd.conf – *only if the system is using xinetd to start network services, see 4.11*;

- by controlling access through the SSH specific options in /etc/xinetd.d/ssh - *only if the system is using xinetd to start network services, see 4.11*;

- by using TCP wrappers and launchd; and

- by controlling access through the ssh daemon itself.

None of these methods is inherently better than any other, but the system administration procedures within your organisation will play a large role in deciding which is used. Generally, it's better to specify host based access controls in as few places as possible to avoid unnecessary administrative headaches. Since most of the access control mechanisms mentioned above are available on all UNIX systems, the configuration of Mac OS X systems may have to conform to organisation wide system administration standards.

4.11.4 FTP

FTP, the File Transfer Protocol, is a well-established protocol for transferring data. It is a clear text protocol, which transmits both the authentication credentials and the data itself unencrypted between the client and server. For this reason it is susceptible to network sniffing attacks and its use on un-trusted network segments should be avoided if possible.

The SSH service offers file transfer mechanisms in the form of scp (secure copy) and sftp (secure file transfer) which encrypt all traffic between the client and server. However, encrypting data incurs a processing overhead which makes the SSH file transfer services slightly slower than FTP.

FTP can be enabled from the *Sharing* section of *System Preferences*.

4.11.4.1 Login banner

By default, the FTP server login banner is stored in a specific file. To display the login banner defined above, it is necessary to create a link to the correct file name:

```
sudo ln -s /etc/login_banner /etc/ftpwelcome
```

This banner will be displayed to users who attempt to connect to the ftp server.

4.11.4.2 User access controls

All users with local accounts on the system are granted ftp access by default. It is recommended that this access be further restricted so that only users with a clearly defined need for ftp are permitted access. Furthermore, the administrative user should not be permitted ftp access to the system since the clear text nature of the protocol poses too great a risk for credential theft. Other users can be restricted based on their username and/or source IP address.

To restrict access on a per user basis, create and edit the file /etc/ftpusers, an example could be:

```
tim deny
jane allow
glyn
martin@192.168.0.1 allow
martin
```

The format of the file is: 'username@host directive'. The most simple entry is a username without a directive, where FTP assumes the user is denied access. In the example above, user glyn will be denied access, along with tim; jane is permitted access. The user martin will be able to access ftp from host 192.168.0.1, but will be denied access from all other hosts.

4.11.4.3 Chroot jail

Users accessing the system through ftp are allowed to move about the entire file system as if connected locally. Read, write, create and delete access to files is limited by their user access privileges and the file permissions. To further protect the system and restrict access to potentially sensitive data, it is recommended that users are only permitted ftp access to their home folders. To enable this feature, create the file /etc/ftpchroot and insert all the usernames who will be granted ftp access, but have their access limited to their home folders. The file is a simple list of usernames, such as:

```
james
bob
design
```

4.11.5 Apache (Personal web sharing)

Personal web sharing is provided through the Apache web server, commonly used as a commercial and personal web-server throughout the Internet. Apache is a full featured, complex web server with a multitude of configurations and their associated security implications. Using Apache to share files is not recommended for most organisations, since more secure alternatives with better authentication mechanisms such as SSH are available.

This section details basic changes to the default installation to allow users to serve access-controlled web pages. A full secure web-server deployment is beyond the scope of this document.

4.11.5.1 Access control

This configuration is suitable for personal web pages where the server is to serve content from users' home directories; access to all other resources will be denied by default. Edit the /etc/httpd/httpd.conf file and find the following section:

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
```

and change it to this:

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>

<Directory /Users/*/Sites>
        <Limit GET POST>
        </Limit>
        Order deny,allow
        Deny from all
</Directory>
```

Access to the URL http://your.mac.system/ will now be denied for everyone. The second 'Directory' section explicitly denies access to all users' Sites directories, but this behaviour is still overridden by the user specific configuration files found in /etc/httpd/users. At the end of the general /etc/httpd/httpd.conf file is the line:

```
Include /private/etc/httpd/users/*.conf
```

This loads individual configuration files for each user, overriding the general access controls that have previously been specified. These user files are automatically created when a new user is added to the system. It's recommended that instead of reading the configuration files of all users, only those of users that are authorised to use web sharing will be read. Instead of using the * wildcard character to read all files, specify each users configuration file explicitly:

```
Include /private/etc/httpd/users/martin.conf
Include /private/etc/httpd/users/jane.conf
Include /private/etc/httpd/users/james.conf
```

4.11.5.2 Granting access

Once access has been denied by default, and only specific users are permitted to share their ~/Sites directories, it is now possible to edit the default configuration for each user file and secure it further. The files are merely extensions to the main configuration file, and follow the same format. A default configuration is included below:

```
<Directory "/Users/martin/Sites/">
```

```
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

By default, everyone is allowed access. If the source IP address/es of the users who will be accessing this web site are known, then these can be specified to further restrict access, for example:

```
<Directory "/Users/martin/Sites/">
    Options Indexes MultiViews
    AllowOverride None
    Order deny,allow
    Allow from 10.0.1.1
    Allow from 192.168.0.1
</Directory>
```

This has the effect of only allowing access to the http://your.mac.system/~martin URL to users from the 10.0.1.1 and 192.168.0.1 systems.

The default configuration of Apache on Mac OS X provides for unencrypted HTTP transport. However, the OpenSSL suite is included in the OS and can be enabled to provide full encryption of web server traffic. See http://developer.apple.com/internet/serverside/modssl.html for more information.

4.11.5.3 Obfuscating server version

As part of a standard HTTP exchange, the server sends a header used to identify itself. Under a default installation of Mac OS X this is returned as: “Apache/1.3.33 (Darwin)” which reveals both the exact version of Apache installed, and the underlying operating system. Attackers using remote scanning software frequently identify vulnerable system by their version numbers obtained from these banners. It is considered good security practice to remove such information from servers, particularly those accessible from public networks.

To change the HTTP server header, edit the /etc/httpd/httpd.conf file, and add the line:

```
ServerTokens prod
```

Once the server is restarted, it will return a header string of “Apache”. This may be further obfuscated through the use of the mod_header Apache component, although its use is beyond the scope of this document. See www.apache.org.

4.11.6 Bonjour

Bonjour, formerly known as Rendezvous is Apple’s implementation of the ZeroConf protocol. It uses network broadcasts to advertise system services on the local subnet, such as printers, iTunes, iChat, SSH and FTP etc.

Advertising the services available on a Mac OS X system could provide an attacker with additional information that may be useful when conducting an attack. In cases where the local subnet is not a

trusted network, such as public WiFi access points, it is recommended that the data being broadcast to the subnet be limited.

It is possible to disable support for Bonjour in each of the affected applications (e.g. iTunes, iChat, etc.) or to disable the Bonjour service as a whole. To disable Bonjour on the Mac OS X system, issue the command:

```
sudo launchctl unload -w /System/Library/LaunchDaemons/com.apple.mDNSResponder.plist
```

It should be noted that Mac OS X was not designed to operate without Bonjour and disabling the service could have a number of adverse affects on the operation of the system such as loss of printer functionality.

4.12 File Sharing

SSH provides the most secure options for sharing files, but is not always convenient for many users, however, it is possible to use SSH to create an encrypted tunnel which can be used by more familiar sharing protocols such as AFP. The file sharing services enabled should largely be dictated by the location of the Mac OS X system on the network and the sensitivity of the data transferred. As a general rule, only encrypted transfer mechanisms such as SSH and VPNs should be used over un-trusted networks. Unencrypted file sharing protocols such as FTP, Microsoft's SMB, NFS and Apple's AFP should only be used on trusted networks. When unencrypted file sharing protocols are used, the authentication credentials of users' can be compromised by network sniffing. This is an added problem when the same credentials are used for remote system access.

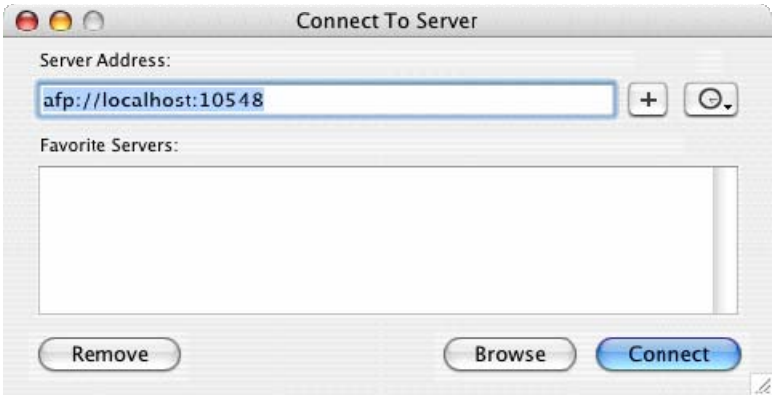
SSH provides a secure method for transferring files across un-trusted networks and also allows the creation of VPN tunnels between an SSH server and client. Apple has used this in Mac OS X Server (>10.2) to allow users to tunnel AFP over SSH by enabling "secure connections" on the server. Even without Mac OS X Server, it is still possible to manually create a tunnel for AFP over SSH.

From the client's command line enter:

```
ssh username@remote.server -L 10548:127.0.0.1:548
```

This tells the SSH client, to connect to the SSH server and map the local port 10548 to the remote port 548. This means that all connections to the local system on port 10548 will be encrypted and forwarded to the remote host on port 548 (The port used by AFP). Once entered, SSH will prompt for a password and then open a shell on the remote system. If the shell is closed, the tunnel will be closed too.

At this point AFP connections can be initiated to the remote server by pointing the server connection wizard at the local system, port 10548:



If the server is going to serve AFP only over SSH, then access to the AFP port from remote addresses can safely be denied by the firewall.

4.13 Intrusion Detection Systems (IDS)

IDS can assist administrators in identifying successful and attempted attacks on a system by monitoring file system integrity or by analysing network traffic for dangerous payloads. The most important consideration when it comes to IDS is not the technology used, but the soundness of the process in which it is used. The software only *detects* attacks and alerts the administrator, it takes no corrective action. For this reason, the real value of IDS is only seen when there are clear and defined processes for correctly managing the alerts generated by these systems. Without these processes, the resources consumed by IDS are largely wasted.

4.13.1.1 File integrity checkers

Host based intrusion detection on OS X comes mainly in the form of file integrity checking programs such as tripwire. These programs, take a snapshot of important system files and folders, sign them, and store them in a secure database. Periodic checks are then performed on the system that compares the current system files with those in the database. If any differences are noticed, in size, ownership or permissions, the designated administrator is notified.

Free software packages that provide this kind of functionality and have been updated for Tiger are:

- Tripwire – is the most mature and tested version with support for almost every OS platform (A commercial version of this program is also available, though not for Mac OS X);
- Radmind – a popular program with a native OS X graphical interface;

The following software packages have been tested on a number of UNIX platforms and may work under Mac OS X:

- Samhein – tested on FreeBSD; and
- Osiris – supports all UNIXs.

Host based IDS provide a means of alerting administrators if an attacker has managed to gain access to a system, and is actively changing files on it. As essential as it is, this form of alerting is often not quick enough to prevent costly damage to a system. Where systems are deployed in high threat environments, it is recommended that a network based IDS be deployed.

4.13.2 Network IDS

Most network IDS are based on packet capturing technology that reads network traffic, checks it for known or suspicious payloads, and alerts the administrator if anything untoward is detected. Since they analyse traffic at the network level, it is possible to be informed of an attack in real-time. There are some systems that use anomaly detection to spot malicious traffic, but by far the most popular method in current technology is to use signature based matching, similar to anti-virus software.

A well-known and free network based IDS is snort which is available on a multitude of platforms. It is distributed in source form and will require compilation and some modification before working on Mac OS X. There is a native Mac OS X package available, that's compatible with Tiger, and makes installing and managing snort easy: http://seiryu.home.comcast.net/henwen.html

References

Hardening guidelines for Mac OS X 10.3

http://www.corsaire.com/white-papers/040622-securing-mac-os-x.pdf

http://www.cisecurity.org/bench_osx.html

http://www.nsa.gov/snac/os/applemac/osx_client_final_v_1_1.pdf

Open Firmware

http://www.macdevcenter.com/pub/a/mac/2003/02/18/secure_tibook.html

http://www.msec.net/advisories/of_pwd_bypass.html

http://docs.info.apple.com/article.html?artnum=106482

Login

http://www.macworld.com/2003/08/secrets/macosxhints/

Keychain

http://www.informit.com/articles/article.asp?p=169576&seqNum=6

Firewall

http://www.cert.org/tech_tips/packet_filtering.html

ipfw manual page

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls.html

File System

chmod manual page

Network Services

http://www.xinetd.org/

xinetd manual page

launchd manual page

http://www.openna.com/documentations/articles/xinetd/part4.php

File Sharing

http://security.ucdavis.edu/MacOSXSecPaulAnnot.pdf

Acknowledgement

This Guide was written by Stephen de Vries, Principal Consultant at Corsaire.

About The Author

Stephen de Vries is a Principal Consultant in Corsaire's Security Assessment team. He has worked in IT Security since 1998, and has been programming in a commercial environment since 1997. He has spent the last four years focused on Ethical Hacking, Security Assessment and Audit at Corsaire, KPMG and Internet Security Systems. He was a contributing author and trainer on the ISS Ethical Hacking course and Technical Leader for the Automated Perimeter Scanning project.

Stephen's past roles have included that of a Security Consultant at a leading City of London financial institution and also Security Engineer at SMC Electronic Commerce. At both positions he was involved in corporate security at many levels and was responsible for consulting on the paper security policies and procedures, conducting vulnerability assessments, designing, deploying and managing the security infrastructure of the organisation.

About Corsaire

Corsaire are experts at securing information systems. Through our commitment to excellence we help organisations protect their information assets, whilst communicating more effectively. Whether they are interacting with customers, employees, business partners or shareholders, our sound advice can help our clients reduce corporate risk and achieve tangible value from their investments.

Privately founded in 1997 and with offices in the Europe and Asia Pac, Corsaire are known for our personable service delivery and an ability to combine both technical and commercial aspects into a single business solution. With over nine years experience in providing information security solutions to the UK Government's National Security Agencies, Government departments and major private and non-profit sectors, we are considered a leading specialist in the delivery of information security planning, assessment, implementation and management.

Corsaire take a holistic view to information security. We view both business and security objectives as inseparable and work in partnership with our clients to achieve a cost-effective balance between the two. Through our consultative, vendor-neutral methods we ensure that whatever solution is recommended, an organisation will never be overexposed, nor carry the burden of unnecessary technical measures.

Corsaire have one of the most respected and experienced teams of principal consultants available in the industry and have consistently brought fresh ideas and innovation to the information security arena. We take pride in being a knowledge-based organisation, but we don't just stop there. Through a culture of knowledge-share, we are also committed to improving our client's internal understanding of security principles.

It is this approach to knowledge that differentiates us from most other information security consultancies. As a mark of this, we are known globally through our active contribution to the security research community, publishing papers and advisories on a regular basis. These we share freely with our clients, providing them with immediate access to the most up-to-date information risk

management advice available, allowing them to minimise their exposure and gain an instant competitive advantage.

Whilst it is imperative for us to offer a high level of security to our clients, we believe that it is of equal bearing to provide a high level of service. At Corsaire our clients are not only protected but valued too. We work hard at building strong relationships that are founded on the cornerstones of respect and trust. With 80% of our customer base deriving from referrals we are certain that our clients value the quality, flexibility and integrity that partnering with Corsaire brings.

For more information contact us at info@corsaire.com or visit our website at http://www.corsaire.com