Global Information Assurance Certification Paper

Mac OS X 10.1.4: Security Analysis and Recommendations
Daniel Deal
June 4, 2002
GSEC Practical Version 1.3

Summary

Mac OS X's core is based on BSD UNIX and, therefore, Mac OS X inherits the UNIX legacy and its security strengths and weaknesses. Apple deserves credit for making their product fairly secure out-of-the-box. All services served by the Internet superserver are disabled by default. To my knowledge, Mac OS X is the first commercial version of UNIX that ships with important third party security tools like TCP wrappers, OpenSSH, and the packet-filtering IP firewall. Mac OS X also provides the ability to notify users when system updates are available. Apple's latest OS lags behind other BSD distributions, though, with regards to some security measures. The operating system lacks any method to hide password hashes from unprivileged users, has insignificant password strength requirements, and lacks the ability to use a password hash algorithm other than DES. Some programs unnecessarily have set-UID and set-GID bits set and this also poses potential problems. This paper is an introduction to the security implications of Apple's latest offering (Mac OS X 10.1.4 at the time of this writing), providing particular focus on NetInfo, Mac OS X's directory system, and is intended to be a starting point for your own research.

Introduction

Mac OS X has broad appeal to both end-users and administrators alike. It melds the familiar features and robustness of UNIX with the user-friendliness expected from a Macintosh OS into a cohesive package. Administrators in heterogeneous environments that include UNIX and Linux systems will welcome Mac OS X to the stable of systems they support. They will appreciate its versatility, built-in security mechanisms, and remote administration capabilities. End-users will value the stable environment provided by Mac OS X's protected memory and pre-emptive multitasking kernel. Apple's latest offering also has the distinction of being the only UNIX flavor that has a native version of Microsoft Office. This feature alone will help Mac OS X to gain a more widespread presence than any other flavor of UNIX.

Because of its broad appeal and its potential to become the most widespread BSD distribution on the desktop, it is imperative that administrators understand the risks Mac OS X introduces to their networks. As security-conscious administrators know, a network is only as secure as its weakest link. Apple's new operating system presents security concerns both common to most UNIX-like operating systems and peculiar to Mac OS X. This paper addresses security concerns regarding choice of filesystem, physical access, NetInfo

directories, password authentication, the internet superserver and other UNIX daemons, and root privilege mechanisms. I will also address strengths such as Mac OS X's built-in firewall, its semi-automated software update feature, its support of popular commercial anti-virus software, and its ability to run powerful UNIX security and security auditing tools.

I have devoted a considerable portion of my research to NetInfo, Mac OS X's directory system. This topic held much interest for me because I was wholly unfamiliar with the system. I found many sources that described NetInfo's primary vulnerability: Unprivileged users and processes can extract sensitive information from a NetInfo directory. I read of a possible solution that involved moving sensitive information to the traditional, protected locations used by other UNIX distributions. I did not, however, find resources that described the effectiveness of this solution or the problems that may arise by altering Apple's intended NetInfo configuration scheme. This lack of information led me to perform my own experiments and I present my results in detail below (see **NetInfo and Disclosure of Sensitive Information**).

The earliest version of the Classic environment supported by Mac OS X is Mac OS 9.1. For the sake of brevity I will use the term "Mac OS 9.x" to refer to Mac OS 9.1 and Mac OS 9.2.x, and I will use "Mac OS X" and "X" interchangeably. I will also use the convention that GUI tools, like *NetInfo Manager*, will be italicized. I will precede command-line examples with a ">" prompt to indicate that these are commands that should be entered at a shell prompt. Additionally, I will use the term "administrative user" to refer to any user belonging to the group "admin," but will use the term "administrators" to refer to the target audience of this paper—those people responsible for maintenance and security of systems on their networks. The two groups may overlap, but I make the distinction here for clarity.

Filesystem Considerations

Mac OS X supports installation on disk partitions formatted using UFS or HFS+ filesystems. There are advantages to installing Mac OS X on a UFS volume. These security advantages outweigh the minor pitfalls associated with using the UFS filesystem.

UFS is case-sensitive like most UNIX filesystems whereas HFS+ is case-preserving, but case-insensitive. The case-insensitive nature of HFS+ creates vulnerabilities. With regard to security, programmers invariably make assumptions. Conclusions drawn from these assumptions become invalid when the assumptions no longer hold true. A well-known illustration of this point with respect to Mac OS X is the use of Apache to serve files on HFS+ volumes. Apache programmers assumed served filesystems would be case-sensitive and this assumption created vulnerabilities when Apple shipped Mac

OS X, with Apache, on HFS+ volumes. A fix has been available for some time, but Apache is certainly not the only UNIX program that has made this assumption. Additional vulnerabilities associated with using UNIX-derived tools on Mac OS X can be avoided by limiting their use to UFS filesystems.

Both filesystems support UNIX metadata (owner and mode bits) necessary for filesystem discretionary access control (DAC) mechanisms, which are common to all flavors of UNIX. On a UFS filesystem, this metadata is stored in a file's inode. On a HFS+ filesystem, this metadata is stored with the file data because HFS+ does not use inodes. Although HFS+ supports DAC metadata, Mac OS 9.x neither creates this metadata nor respects DAC metadata created by Mac OS X. A disadvantage of installing X on a HFS+ volume is that booting into Mac OS 9.x will circumvent DAC mechanisms and allow a user unrestricted access to your Mac OS X volume (see **Physical Access Concerns**, below). Mac OS 9.x cannot, however, mount or read UFS volumes. This inability to use UFS volumes may be inconvenient, but putting a Mac OS X installation on a UFS volume reduces the vulnerabilities posed by physical access to the computer.

There are several caveats regarding the installation of Mac OS X on a UFS volume. If X is installed on a UFS volume, the system volume will be named "/" instead of the default "Mac OS X" and any customization of the volume name will be lost upon restart. Having the root of the filesystem named "/" is at worst an annoyance and at best a familiar feature to UNIX users. Perhaps the only genuine problem will persist only as long as Classic applications continue to be used: The type/creator code scheme used to associate documents with applications in Classic will not function if Mac OS X is installed on a UFS volume. If a user of Mac OS X relies on Classic applications, the convenience of launching these applications with a simple double-click on a document in Finder may outweigh the benefit of improved security that the UFS filesystem offers. Other issues with using Mac OS X on UFS seem trivial. These issues include an "Open Enclosing Folder" bug in Sherlock and an issue with the Classic environment not starting without some manual modifications.

Airport, Apple's name for 802.11b wireless Ethernet, presents its own security risks: ease of network sniffing and decryption of traffic by attackers even if WEP is used. As of Mac OS X 10.1, however, Airport now functions when X is installed on a UFS volume. Airport functionality is the only problem regarding the use of UFS that Apple has addressed to date, but the outstanding UFS issues have minimal impact upon a Mac OS X system. Overall, UFS filesystems presents fewer vulnerabilities than HFS+ filesystems. Unless the use of type/creator codes is imperative, my recommendation would be to ensure that Mac OS X is installed on a UFS volume.

Physical Access Concerns

Physical access to a Macintosh running Mac OS X presents a number of security concerns. Just as with other flavors of UNIX, physical access to the machine provides the ability to bypass existing security mechanisms by booting from a device other than the default boot device (e.g. installation CDs or alternate boot disks, either internal or external). Booting from installation CDs, especially, poses greater risks for Mac OS X systems than it does for other commercial UNIX flavors because the installation CDs are easier to obtain. Every new Macintosh ships with installation CDs, whereas, in my own experience, Sun SPARCstations and SGI IRIX workstations most often do not ship with installation CDs. I cannot speak for IBM AIX or HP-UX systems, and it is true that both Solaris can easily be downloaded from Sun's website and any distribution of Linux can be downloaded from numerous mirror sites, but Mac OS X installation CDs may be more prolific due to likely higher-volume sales of Macintoshes than other UNIX systems. Due to the ease of obtaining and booting installation CDs or booting from alternate boot volumes and bypassing filesystem permissions, it is doubly important that administrators of Mac OS X systems understand the methods available to boot alternate startup volumes and what measures can be taken to prevent unauthorized use of these features.

Administrative users can boot from alternate locations by selecting an alternate System folder in the *Startup Disk* preference pane in *System Preferences*, by holding down the *option* key during startup, or by holding down the *c* key to boot from an installation CD. If the system is booted into Mac OS 9.x, filesystem permissions on HFS+ volumes can be circumvented, allowing the equivalent of root-level access to those volumes. Booting from alternate Mac OS X installation locations circumvents filesystem permissions for both HFS+ *and* UFS volumes. If an attacker has installed X on an external drive and can boot from it, he can authenticate against his own root or administrative user password hash on the external drive instead of the hashes stored in the default boot device. Installing X on a UFS volume obviously imparts no resistance to this method of attack.

Apple has also provided a method by which a user may reset any user password on a Mac OS X system. This is accomplished by booting from a Mac OS X CD and selecting "Reset Password" from the *Installer* menu. Apple considers this a feature. It will certainly be useful in a home setting where an administrative user may not understand the importance of remembering passwords, but it presents a risk of which any administrator should be aware.

"Target Disk Mode" also enables booting from an alternate volume. Mac OS X systems that have built-in FireWire ports can be started up in Target Disk Mode by holding down the *t* key upon startup. Connecting another Macintosh via FireWire cable to the system booted in Target Disk Mode will allow the mounting of its volumes. If the host computer is running Mac OS 9.x, it will be able to mount HFS+ volumes on the target computer. If the host computer is running Mac OS X, it will be able to mount UFS *and* HFS+ volumes. Either way,

the host computer will potentially gain root-level access to any volumes it can mount.

Another method of booting a Mac OS X system is single user mode. One may enter single user mode by simply holding down the *command-s* key sequence during system startup. The risk here is that single user mode requires no authentication by default and imparts root-level access to the system.

The most apparent method to eliminate these risks associated with physical access to a Mac OS X system is to change the "security-mode" variable in the system's Open Firmware. This setting is supported by Apple Open Firmware 4.1.7 and later. Supported values for this setting are "none" (the default), "command," or "full." The effects of these values of the "security-mode" variable, at the Open Firmware prompt, are described clearly by CodeSamurai in a SecureMac.com article:

> The "command" mode just restricts the commands that may be executed to "go and "boot." Additionally, under the "command" mode, the "boot" command may not have any arguments—that is, it will only boot the device specified in the boot device [sic] variable; no other command may be entered or any settings changed unless the password is supplied. Moreover, this password protection feature also applies to booting up with the option key held down (which allows you to choose from available bootable volumes…). Finally, in "full" mode, the machine is completely prohibited from booting until the password is entered (21).

Apple provides a GUI utility called, appropriately enough, "Open Firmware Password" to set the Open Firmware security-mode variable to "command" and create an Open Firmware password. Once these settings are enabled and a password is set, (in addition to the Open Firmware command restrictions outlined above) keys that affect normal startup are disabled. An Apple Knowledge Base document provides details:

> When turned off, Open Firmware Password Protection:
> - blocks the ability to use the "C" key to start up from a CD-ROM disc.
> - blocks the ability to use the "N" key to start up from a NetBoot server.
> - blocks the ability to use the "T" key to start up in Target Disk Mode (on computers that offer this feature).
> - blocks the ability to start up in Verbose mode by pressing the Command-V key combination during startup.
> - block [sic] the ability to start up a system in Single-user mode by depressing the Command-S key combination

during startup.
- blocks a reset of Parameter RAM (PRAM) by pressing the Command-Option-P-R key combination during startup.
- requires the password to use the Startup Manager, accessed by pressing the Option key during startup…
- requires the password to enter commands after starting up in Open Firmware, which is done by depressing the Command-Option-O-F key combination during startup. (11)

To enable these keys again the Open Firmware Password application must be used to reset the security-mode variable to "none." The password can be reset and changed 1) by any user of the admin group, 2) by starting up the computer from a Mac OS 9.x System Folder, or 3) if one has access to the internal hardware of the Macintosh. If the first method poses a risk, then administrators should verify that all users belonging to the admin group require such privilege and should consider using the sudo utility to allow finer-grained control of administrative privileges than the admin group scheme allows (see **Authorized Root Privilege Mechanisms**, below). The Open Firmware password itself will prevent all but one method (the *Startup Disk* preference pane) of booting Mac OS 9.x, so method two should pose no risk. If there is a threat associated with the vulnerability of physical access to the internal hardware, an administrator should lock the case of the Macintosh.

It is important to note that Apple neither supports nor endorses the use of these Open Firmware security measures on versions of Mac OS X earlier than 10.1 or when used with third-party software utilities. Improperly changing Open Firmware settings may cause damage that only Apple can repair and these repairs may not be covered by Apple's warranty. Good examples of potential harm are reports of permanent Open Firmware corruption if the Open Firmware password is not disabled before performing a firmware update.

The msec group has released a utility called *FWsucker* that will extract and decrypt the Open Firmware password. It is available at http://www.msec.net/software/FWSucker.sit. This program comes with little documentation and I have found that it worked only if my Macintosh was booted into Mac OS 9.2.2. It would not work while Mac OS X was booted. This program should pose little risk because unprivileged users will not be able to boot into Mac OS 9.x if Open Firmware is password-protected. If the Open Firmware password is set, the only way to boot into Mac OS 9.x without knowing the firmware password is to select a Mac OS 9.x system folder in the *Startup Disk* preference pane. This action can only be performed by users of the admin group. Note that it is trivial procedure, then, for any administrative user to gain the Open Firmware password.

Leaving a system unattended while logged-in as a user with administrator privileges or with an open shell that has administrator or root privileges is

against recommended practices on any flavor of UNIX. All users should password-protect their screen saver and activate it when they step away from a Mac OS X system. This will prevent passersby from tampering with the system. One may enable this effect by clicking the "Use my account password" in the "Activation" tab of the *Screen Saver* panel in *System Preferences*. One should also select an appropriately short delay for screen saver activation using the slider here and create a hot-corner for immediate activation of the screen saver in the "Hot Corners" tab.

An out-of-the-box Mac OS X install, once activated by the creation of the first administrative user, may be setup to automatically login that user upon system startup. This behavior should be disabled by unchecking the "Automatically log in" box in the "Login Window" tab of the *Login* preference pane in *System Preferences*. A final precaution that should be taken is to prevent Mac OS X from revealing valid usernames in the login window. One may do this by clicking the "Name and password entry fields" radio button, under the "Display Login Window as:" heading on the same tab.

To lessen the risk associated with physical access to a Mac OS X computer, administrators should make several changes to a default installation. They should create an Open Firmware password. This measure disables most methods of booting from alternate boot devices. They should carefully limit the privilege of belonging to the admin group to restrict the use of the *Startup Disk* preferences pane to boot from alternate locations. Administrators should disable automatic login and disable the display of usernames in the login window. They should physically lock the cases of Macintoshes. Additionally, all users should use a password-protected screen saver. The sum of these measures is a more physically secure Macintosh.

NetInfo and Disclosure of Sensitive Information

NetInfo is intended to function as a hierarchical, distributed directory system and it performs this function well. It was not designed, however, to be secure. NetInfo provides information to NetInfo domain clients in a fashion similar to Sun's NIS system. It also presents similar vulnerabilities. As with NIS, an unprivileged user can obtain any directory information from the NetInfo domain to which a Mac OS X system is bound (either from a NetInfo server or from the local database). Administrators can configure NetInfo tools to consult traditional UNIX "flat files" rather than NetInfo directories when C library functions make directory queries. This presents the possibility to restrict viewing of sensitive data by using filesystem DAC mechanisms. This section will explore this alternative, present a procedure for implementing it, and explain the consequences of altering Apple's default configuration. I will also describe how and when Mac OS X backs up NetInfo, touch upon the existing "shadow property" security mechanism of NetInfo, and present changes to restrict access

to NetInfo's command-line utilities.

NetInfo shares a weakness with NIS that permits the disclosure of sensitive information to unprivileged users and processes. Any user in an NIS domain can issue "ypcat passwd" and obtain user account information and password hashes. Likewise, an unprivileged Mac OS X user can issue "nidump passwd /" to obtain the same information from a NetInfo domain. If a Mac OS X system is not bound to a remote NetInfo server, the above command will be equivalent to "nidump passwd ." and will dump the local NetInfo user information in passwd file format.

Under the right conditions, this method can also be used if a malicious user on a system not belonging to a domain can guess the name of a NetInfo domain and the corresponding name or address of a domain server. By default, the NetInfo "trusted_networks" property exists, but has no value. This has the effect of denying access to the system's served NetInfo directories to all but the localhost. If the property does have valid values, then any system on a trusted network could coax a NetInfo server to reveal any portion of a served NetInfo directory. The bottom line is that a default install of Mac OS X is not vulnerable to remote exploitation of NetInfo's trustfulness, but administrators should be aware of this weakness of NetInfo as a distributed directory system.

How can the local problem of NetInfo's revelation of password hashes and other sensitive information be circumvented? Though Mac OS X has a shadow password file, /etc/master.passwd, its comments tell us it is only referenced if the system is started in single user mode. It turns out that an administrator can configure the NetInfo query daemon, lookupd, to search traditional flat configuration files in the /etc directory before (or instead of) the NetInfo database.

The lookupd daemon answers queries from C library routines, like getpwuid(), that on most UNIX systems would be answered by searching configuration files in the /etc directory. On a default Mac OS X install, lookupd consults the local.nidb NetInfo directory database in /var/db/netinfo/. Unless explicitly configured to do so, lookupd will never consult files in the /etc directory.

The lookupd daemon uses search "agents" to answer queries. Once it finds an answer to a query it caches the result to speed up subsequent queries. Search agents include the CacheAgent (which searches the lookupd cache), the NIAgent (which searches NetInfo databases), the YPAgent (which searches NIS/YP domain servers), the LDAP Agent (which searches LDAP servers), the DNSAgent (which queries DNS servers), the FFAgent (which searches flat files), and the NILAgent (which always returns a negative response in order to stop a search). We are concerned with the flat file agent, FFAgent. The default order is to search using the CacheAgent and then the NIAgent. We want lookupd to use the FFAgent before it searches using the NIAgent. The FFAgent will consult

many well-known configuration files in the /etc directory (14):

/etc/master.passwd	Users
/etc/group	Groups
/etc/hosts	Computer names and addresses
/etc/networks	Network names and addresses
/etc/services	TCP/IP service ports and protocols
/etc/protocols	IP protocol names and numbers
/etc/rpcs	ONC RPC servers
/etc/fstab	NFS mounts
/etc/printcap	Printers
/etc/bootparams	Bootparams settings
/etc/bootp	Bootp settings
/etc/aliases	E-mail aliases and distribution lists
/etc/netgroup	Netgroups

There are several steps required to alter the order of the search agents consulted by lookupd. I will outline here the procedure necessary to configure lookupd to search for user information and password hashes in the secure (mode 0600) /etc/master.passwd file before consulting the local NetInfo database. You must be the root user to complete this procedure, but I encourage you to read this entire document before attempting to alter the lookupd search order—there are surprises ahead.

We should backup our NetInfo database before proceeding. We should perform our own backup now, but note that the /etc/crontab file tells the cron daemon to run /etc/daily every day at 3:!5 AM. This daily script backs up NetInfo directory databases by performing a nidump of all domain databases found in /var/db/netinfo and writes them to /var/backups/<*domaintag*>.nidump. An unmodified Mac OS X install will have only the local database, local.nidb. Note that this dump is world-readable by default. It is in our best interest to apply the principle of least privilege here and alter the /etc/daily file to prevent unprivileged users or processes from examining this text file. I have included the relevant section of my /etc/daily script below:

```
if [ -d /var/db/netinfo ]; then
   echo ""
   echo "Backing up NetInfo data"
   cd /var/db/netinfo
   for domain in *.nidb; do
      domain=$(basename $domain .nidb)
      nidump -r / -t localhost/$domain > $bak/$domain.nidump;
      chmod 400 $bak/$domain.nidump
```

The last line I have added; the rest is the default. I encourage you to make a

similar change.

We can follow Apple's lead and use the same method to backup our local NetInfo database using nidump:
>nidump –r / -t localhost/local > /var/backups/local.nidump
>chmod 400 /var/backups/local.nidump
Now that we have a backup of our NetInfo directory database, we modify it without fear of damaging our only copy. If mistakes are made, we can restore from our backup using niload:
>niload –r / . < /var/backups/local.nidump

The first step of the procedure to change the lookupd search order is extraction of password hashes from NetInfo.
1. Use the nidump utility to dump the NetInfo user information in passwd file format:
>nidump passwd .

The preferred method of altering the /etc/master.passwd file is to use vipw. The vipw program performs file locking to ensure consistency of the /etc/master.passwd file. It will also verify that the file conforms to the correct passwd file format and will then run pwd_mkdb, which creates the db-format password files /etc/pwd.db and /etc/spwd.db. Based on my limited testing (moving the master.passwd file), the FFAgent does not consult the secure spwd.db database for password hashes, as would be the case on other BSD distributions.

2. If you have few users, you can run vipw and cut and paste the hashes into the /etc/master.passwd file. If you have many users, you may want to issue the command "nidump passwd . > /etc/master.passwd" in single user mode, then "chmod 600 /etc/master.passwd."

3. We must create the necessary /locations/lookupd directory in our local NetInfo database:
>nicl . –create /locations/lookupd

Now we need to populate the directory with the needed subdirectories, properties, and values. The sample below can be cut and pasted into a text file. This is a modified version of the sample from Apple Knowledge Base article 106499 (10). Here the FFAgent precedes the NIAgent for user lookups. A note regarding lookupd and logging: lookupd by default does not perform any logging. The third line in the file below will enable logging, but by default lookupd will only log LOG_NOTICE or higher priority messages. To change this behavior, the next line is used to set the log message priority threshold for lookupd (to LOG_INFO). My limited testing has shown that lookupd does not log much, regardless of the syslog priority threshold you declare here.

{

```
"LookupOrder" = ("CacheAgent", "NIAgent");
"name" = ("lookupd");
“LogFile” = (“/var/log/lookupd.log”);
“LogPriority” = (“LOG_INFO”);
"MaxThreads" = ("12");
      CHILDREN = (
            {
            "name" = ("users");
            "LookupOrder" = ("CacheAgent", "FFAgent", "NIAgent");
            }
      )
}
```

4. Put the above text into an appropriately named file, lookupd.config, for example, and use the niload utility to modify the NetInfo database:
>niload –r /locations/lookupd . < lookupd.config

5. Send lookupd the SIGHUP signal so that it will reread its configuration.
>kill –HUP `cat /var/run/lookupd.pid`

Sending lookupd the SIGHUP signal can sometimes cause it to behave strangely. In my testing, at least once lookupd no longer referenced the FFAgent though this agent was first in the lookup order. Restarting Mac OS X resolved this problem.

6. Now your system will look for user information in /etc/master.passwd, which only the root user can read. We should now remove, at the very least, the unprivileged user password hashes from our NetInfo database. Again if you have many users, you will want to write a script to do this. We can use the niutil command to lock user accounts if lookupd consults the NIAgent:
>niutil –createprop . /users/<username> passwd “*”

Now the NetInfo utilities cannot reveal user password hashes. But this arrangement of lookupd creates several problems. The lookupd configuration does not alter the behavior of other user management programs. For example, the Mac OS X passwd command and the graphical user administration tool, the *Users* preference pane in *System Preferences*, update only the NetInfo directory database. User changes made using these tools will require administrators to create crontabs that update /etc/master.passwd at regular intervals to maintain this lookupd search arrangement. The administrator would also need to replicate or move sensitive data from other NetInfo directories to their flat file counterparts (NetInfo /machines to /etc/hosts, for example).

Another problem with this lookupd arrangement exists. When authenticating to the *NetInfo Manager* application in order to make changes, the application only attempts to authenticate against the user’s hash in the “passwd” property of the

/users/<*username*> NetInfo directory. Removing or locking the "passwd" property of administrative users will prevent them from using this application unless the root account has been enabled. Removing all password hashes from the NetInfo database will effectively disable the ability of this application to make changes, which some administrators may wish to do.

Though the chpass and vipw commands can be used to update or modify /etc/master.passwd, Mac OS X includes no adduser command to better automate user account creation. The adduser command would typically add an entry to /etc/master.passwd, create a new home directory, and populate it with files from /etc/skel. The Mac OS X equivalent of /etc/skel is /System/Library/UserTemplate/English.lproj. A tarred version of the skeleton files and directories can be found at /System/Library/PrivateFrameworks/Admin.framework/Versions/A/Resources/English.lproj/user.gnutar. I have not found any reference to Mac OS X-specific versions of adduser or passwd that update the flat files in /etc. I would presume that source for the passwd command from another BSD distribution may be compiled and work correctly with flat files (or better yet Argonne National Laboratory's anlpasswd or Matt Bishop's passwd+ could be used). There exist, however, several adduser shell scripts available on the Internet that will allow Mac OS X administrators to add users to a local NetInfo database. The Linux/UNIX, searchable, software-update tracking site, http://freshmeat.net, contains a link to adduser and deluser scripts by Dino Amato, at http://www.brownnut.com/adduser_OSX.htm. Aaron Faby also provides a similar set of scripts at http://www.aaronfaby.com/netinfo-scripts.1.0.tar.gz.

The simplest route for an administrator who wishes to maintain flat files may be to use these third-party NetInfo tools and the *User* preference pane to modify the system's configuration and have cron update his flat files regularly. Perhaps the dearth of information that I have found on the Internet regarding the above lookupd changes and their consequences is a sign that Mac OS X administrators and developers are throwing their hands up over the issue of NetInfo security and simply decided to live with it and its deficiencies rather than deal with the inconvenience of moving data from NetInfo and maintaining flat files.

Other issues with NetInfo regard the differences between its Mac OS X and Mac OS X Server implementations. Mac OS X Server uses valueless NetInfo properties called "_shadow_<*propertyname*>" that are intended to hide the value of "*propertyname"* from unprivileged NetInfo requests for such directory information. The Apple Knowledge Base only explicitly describes this shadowing effect for Mac OS X Server 1.x, but the *User* preference pane of Mac OS X, nevertheless, creates "_shadow_passwd" properties for new users. This makes sense because Mac OS X and Mac OS X Server must certainly share much source code. On Mac OS X, though, this shadow property does not prevent unprivileged requests from retrieving the value of the "passwd" property

(the users' password hash). It would seem that Apple has removed this shadowing behavior in Mac OS X, but did not remove its Mac OS X Server vestiges. Does Mac OS X Server's NetInfo property shadowing work well? From what I have read, it can be easily circumvented, and a false sense of security is often worse than none at all. The removal of this shadow property functionality is no great loss to Mac OS X administrators.

So is the lookupd FFAgent the answer to hiding user password hashes and other sensitive directory information? Unfortunately, it is not. After reading about the "_shadow_passwd" functionality in Mac OS X Server, I wanted to verify that Mac OS X ignored this property. Apple describes the intended Mac OS X Server behavior of the "_shadow_passwd" property in Knowledge Base article 60112:

> When an executable without special privileges uses BSD-level APIs such as getpwent, getpwnam, and getpwuid, an asterisk ("*") is returned for the passwd field rather than the actual protected password. (13)

My goal was to determine the Mac OS X behavior of unprivileged getpwuid() calls when lookupd uses the NIAgent and the FFAgent. I expected if the NIAgent were used that getpwuid() would return a valid hash, regardless of the existence of the "_shadow_passwd" property for a user. I also expected that if the FFAgent were used that getpwuid() would return an asterisk instead of a valid hash because unprivileged calls would consult the passwd file rather than the master.passwd file, which they would not be able to read.

To test my hypothesis, I configured lookupd to search using only the NIAgent. I wrote and compiled this simple bit of code:

```
#include <stdio.h>
#include <sys/types.h>
#include <pwd.h>

int main (int argc, char* argv) {

struct passwd *p;

        p = getpwuid(0);
        printf("root hash is %s\n",p->pw_passwd);
        p = getpwuid(505);
        printf("test hash is %s\n",p->pw_passwd);
}
```

Note that this code was compiled by an unprivileged user, was mode 0755, and the owner and owner-group of the executable were the unprivileged user and group staff, respectively. The results will not surprise the reader because, as stated above, Mac OS X does not appear to support "_shadow_<*propertyname*>" properties in NetInfo. The root user has no

"_shadow_passwd" property. I expected that this program would reveal this hash and it did. The user with uid 505 was created with the *User* preference pane and did have the "_shadow_passwd" property, but, as expected, this user's hash was revealed, too.

I then configured lookupd to search for user information using only the FFAgent and ran the above test program, again as an unprivileged user. The results were surprising. The program printed the valid hashes for both users as found in /etc/master.passwd. I have tested this on two systems because I was skeptical. I encourage Mac OS X administrators to verify these results for themselves. My findings are that any Mac OS X user can determine all user password hashes regardless of whether the hashes exist in a NetInfo directory or the mode 0600 shadow file, /etc/master.passwd.

NetInfo Conclusions and Recommendations

Where does this leave the frustrated administrator? We can lessen our risk by denying access to the NetInfo utilities and our NetInfo database(s) to unprivileged users. This should be done regardless of whether an administrator decides to use flat files or NetInfo. The shell script that I have included below is adapted from one found at http://sleight.port5.com/dl/ni.patch (45).

```
#!/bin/sh
# Change permissions for NI CLI utilities
chmod go-rwx /usr/bin/nicl
chmod go-rwx /usr/bin/nireport
chmod go-rwx /usr/bin/niutil
chmod go-rwx /usr/bin/nigrep
chmod go-rwx /usr/bin/nifind
chmod go-rwx /usr/bin/nidump
chmod go-rwx /usr/bin/niload
# Change permissions for Aqua NetInfo Manager
chmod  o-rwx \
/Applications/Utilities/NetInfo\ Manager.app/Contents/MacOS/NetInfo\ Manager
# Change permissions for NetInfo Databases
chmod go-rwx /var/db/netinfo/local.nidb
chmod go-rwx `find /var/db/netinfo -name '*.nidb'`
# Change permissions for NetInfo Backup Directory
chmod go-rwx /var/backups/
```

These changes will prevent unprivileged users from using the NetInfo tools, but they will not prevent them from copying their own versions of the tools from another Macintosh or writing C programs that use the getpw*() system calls. I encourage administrators, nonetheless, to make these changes.

NetInfo was designed as a directory system, not a secure authentication system, and was created before much emphasis was placed on computer security. Its behavior reflects these roots. NetInfo and its associated tools will, on a default installation on X, allow any user to obtain sensitive system configuration and user data. Altering the search order of *lookupd* may allow UNIX administrators to more easily maintain consistent configurations across their UNIX platforms, but at this time it does not impart any additional security to Mac OS X. If and when Apple fixes the behavior of lookupd and query functions like getpwuid() and its ilk, this alternative may prove more secure than using NetInfo for password authentication. For now, the only useful measure administrators can take is to restrict access to the NetInfo command-line utilities using filesystem DAC mechanisms.

Password Concerns and Strength Testing

Perhaps of more concern than the disclosure of password hashes is whether or not user passwords are strong and resistant to dictionary attacks. The Mac OS X passwd command seems to perform only one password strength check; it will reject passwords less than five characters long. The *User* preference pane is even worse; it only requires four-character long passwords. This leaves something to be desired. Additionally, I have found through my own testing that the *User* preference pane and the passwd command do not consult /etc/passwd.conf. Attempting to configure passwd to use Blowfish encryption with /etc/passwd.conf does not result in password hashes beginning with "$2," as would be expected of Blowfish hashes, but results in DES hashes.

Though the company targets its products for home consumers who may have no interest in strong password security, Apple should provide strength-checking facilities that are easy to enable. The password creation and change programs should allow an administrator to configure the minimum password length, enable password aging, select a stronger hashing algorithm than DES, and include a way for an external tool or filter to check the strength of passwords—perhaps by making the password tools pluggable authentication module-aware.

The lack of any method to force the creation of strong passwords requires administrators to use another proactive password-checking method: password cracking. This should only be performed with authorization, preferably written authorization, from someone with the authority to confer it upon an administrator. Many tools exist to perform password cracking. A fairly complete listing of these tools can be found at http://packetstorm.decepticons.org/Crackers/, though many of the tools listed there are not related to UNIX password cracking. The tools I describe below all have, in addition to dictionary modes, a brute-force mode that will exhaustively

search the entire keyspace of possible passwords. Ultimately, though, the effectiveness of any cracking program is directly related to the size and appropriateness of dictionaries supplied to it.

One of the most well known tools is the venerable Crack, written by Alec Muffett. A brief tutorial exists for Crack to aid in its compilation on Mac OS X (47). Some changes to the Makefile and one source file are required and this tutorial provides some hints to help get Crack compiled. I found that compiling Eric Young's libdes library, included with the Crack distribution, with optimization options "-O4 -fomit-frame-pointer -funroll-loops" rather than the tutorial's suggestion of "= -O" increased the performance of the libdes library's crypt() function from about 31,000 crypt()s per second to about 41,400 crypt()s per second on my Power Macintosh G4/450. Crack supports whatever crypt algorithm you wish to use and its manual.txt file contains information on using a library other than libdes if needed. This could come in handy if or when Mac OS X adopts the MD5 or Blowfish algorithms for password hashing.

John the Ripper, a password cracker known for its speed, is also available for Mac OS X, albeit in development form. The latest development build of John the Ripper, version 1.6.31-dev, includes "macosx-ppc-cc" as a build target. In order to use the development version, you will need to copy the run/*.chr charset files from a version 1.6 distribution to the run directory of the development build. A default build is able to perform about 47,000 combinations per second on my system. If John is compiled with "-O4 -fomit-frame- pointer -funroll-loops" rather than the default optimization level "-O3," his performance jumps to about 73,000 combinations per second on my Mac. John's FAQ describes this metric as combinations of login and password per second, not crypt()s per second, but running "./john –test" reveals that this metric is just a bit less than actual crypt()s per second on my system. John will also crack MD5 and Blowfish hashes without any intervention by the user.

Several Mac OS Classic API crackers are also available. I have found many references to a program called *Meltino*. I have seen mention of versions 1.32 and 2.01 but links to the creator's website are no longer valid and version 2.01 appears to have been released around Jun 22, 1999 (27). I am led to believe it is not being maintained.

Another Mac OS password cracker that uses the Carbon API, and can be considered a native Mac OS X GUI application, is *Macintosh Hacker's Workshop* (MHW). MHW contains four modules: Generator, Gecos-Reader, Wordlist-Cleaner, and Cracker. The Generator module allows the creation of sequential wordlists using user-selected keyspaces. The Gecos-Reader module extracts Gecos-field words from password files to add to wordlists (something I think the application should do without request). The Wordlist-Cleaner module removes duplicate words from dictionaries (handy if you've merged many dictionaries). Finally, the Cracker module does the real work. On

my system, MHW performed about 36,000 “passwords per second” with only Finder running. Presumably this metric is very close to crypt()s per second.

Though MHW is limited to cracking DES hashes and is not as fast as John the Ripper, it will be worthwhile for Mac OS X administrators to keep their eyes on this tool because, being the only Mac-centric cracking tool of the group, it may be the most likely to eventually include AltiVec-enabled crypt algorithms. Judging by the speed improvement the PPC74xx CPU’s distributed.net clients have received from Dan Oetting’s AltiVec-enabled code additions, this could mean a vast improvement in the speed of MHW. Hopefully such AltiVec-enabled crypt code would eventually work its way into the other tools as well.

Administrators should perform their own password-strength auditing, for three reasons: Mac OS X has no built-in mechanisms to ensure strong password creation, it provides no method to obscure password hashes from unprivileged users, and it uses only the relatively computationally-unintensive (compared to the Blowfish or MD5 algorithms) DES password-hash algorithm. Two well-known password cracking tools from the UNIX realm are available for Mac OS X. Both Crack and John the Ripper are capable tools for password-strength auditing. The Macintosh Hacker’s Workshop, a native Mac OS X GUI application, may also prove useful. Though John the Ripper is the fastest tool of this group, its speed can be significantly improved by changing compiler optimization flags. Regardless of their relative speeds, one must supply them with good dictionaries to effectively use any of these tools. Below are links to the cracking tools I have mentioned, as well as links to some dictionary files that will prove useful.

Crack 5.0a by Alec Muffett:
ftp://ftp.cert.dfn.de/pub/tools/password/Crack/Crack_5.0a.tar.gz
John the Ripper:
http://www.openwall.com/john/
Meltino 2.01:
http://freaky.staticusers.net/cracking/Meltino/Meltino2.01_PPC.sit.bin
Macintosh Hacker’s Workshop 1.1 and its documentation:
http://grungie.code511.com/MHW_1.1_Release.sit.bin
http://grungie.code511.com/MHW_11_Doc/index.html
Team2600’s 350,000+-word dictionary, Dict2000
http://www.team2600.com/software/downloads/Dic2000.sit
Other dictionary links:
http://www.outpost9.com/files/WordLists.html
ftp://ftp.ox.ac.uk/pub/wordlists

Superserver and Daemon issues

Though Mac OS X retains the BSD startup scripts /etc/rc and /etc/rc.common,

many daemons are started from scripts below the /System/Library/StartupItems/directory, including all the daemons mentioned below. I encourage administrators to inspect these startup scripts, as well as /etc/hostconfig, which controls the execution of some of these scripts.

The internet super-server, inetd, is started by the IPServices script when a Mac OS X system boots. A default install of X has all services that inetd may potentially launch disabled (commented out) in /etc/inetd.conf. The only inetd-launched daemon that may be enabled via Mac OS X GUI tools is ftpd, which may be enabled in the *Sharing* preference pane by checking the "Allow ftp access" checkbox on the "File & Web" tab. The inetd.conf file wraps all daemon requests that can be wrapped (RPC daemons using TCP cannot be wrapped) using Wietse Venema's TCP wrappers (/usr/libexec/tcpd). Nothing, though, is effectively wrapped unless one creates tcpd's /etc/hosts.allow and /etc/hosts.deny configuration files. The best practice here is to deny access to everyone to whom it is not explicitly allowed. One may accomplish this by issuing "echo ALL:ALL > /etc/hosts.deny" and then explicitly adding authorized daemon and users/hosts combinations to /etc/hosts.allow as described in the hosts_access(5) man page. Alternatively, xinetd, the secure replacement for inetd, may be obtained from http://www.xinetd.org and administrators unfamiliar with its compilation and use may find a MacSecurity.org tutorial useful (23).

Mac OS X includes an implementation of the secure replacement for telnetd and the insecure "r*" utilities (i.e., rsh, rlogin, rexec), OpenSSH. The current version distributed by Apple via the *Software Update* preference pane is 3.1p1. The OpenSSH daemon, sshd, is started by the SSH script upon system startup if the "Allow Remote Login" checkbox is checked on the "Application" tab in the *Sharing* preference pane. This version of OpenSSH has not been compiled to use Wietse Venema's libwrap library.

At the date of this writing, the latest available version of OpenSSH, found at http://www.openssh.org, is 3.2.3p1 and administrators that wish to do so may compile and install this version using guidelines found at Stepwise.com (1), though the instructions found there are for an earlier version of OpenSSH. I encourage administrators who want a libwrap-capable sshd server to compile and install the latest version themselves.

The portmap daemon, which is needed to answer remote procedure calls, is launched at startup by the Portmap script if any NFS exports have been defined. Consulting the portmap man page, we find that this is a secure version which can be wrapped by adding access rules to /etc/hosts.allow as described in the hosts_access (5) man page. The portmap man page notes, however, that access rules must describe hosts by IP address only.

Sendmail is well known for having vulnerabilities throughout its long history, but administrators may find it useful because some other programs, like cron, mail

log and error messages to the root user. The sendmail daemon, as it ships configured with Mac OS X, is broken. Jeremy Mate has written an article (39) describing a few ways to fix the directory permission problems that cause this. He also includes an outline of how to upgrade the relatively old version, 8.10.2, that ships with X to the latest version from http://www.sendmail.org, which is currently 8.12.3. An administrator would be wise to add firewall rules (see **Firewalls**, below) to drop TCP and UDP packets destined to port 25 that do not originate from localhost.

The cron daemon, which uses crontab files to schedule jobs, is started at system startup by the Cron script. By default the only crontab that exists on a Mac OS X system is /etc/crontab and this file schedules the execution of the /etc/daily, /etc/weekly, and /etc/monthly scripts:

```
-rw-r--r--  1 root  wheel   725 Sep  2  2001 /etc/crontab
-rw-r--r--  1 root  wheel  3690 May 18 12:32 /etc/daily
-rw-r--r--  1 root  wheel  1684 Sep  2  2001 /etc/weekly
-rw-r--r--  1 root  wheel  1101 Sep  2  2001 /etc/monthly
```

There is no reason that unprivileged users should need to read these files and administrators should remove read permissions for others:
>chmod o-r /etc/crontab /etc/daily /etc/weekly /etc/monthly
Administrators should note that without the existence of the /var/cron.allow or /var/cron.deny files any local user may create crontab files in /var/cron/crontabs using the crontab command. Administrators would be well served to limit the privilege of crontab creation by creating either of these files.

A measure Apple recommends is to enable ntpd, the network time synchronization daemon. Accurate time will aid in intrusion forensics if the need arises. One can enable network time synchronization by checking the “Use a network time server” checkbox on the “Network Time” tab of the *Date & Time* preference panel and selecting an appropriate NTP server from the presented dropdown menu.

Authorized Root Privilege Mechanisms

Mac OS X ships with the root account disabled, though there are several ways to enable it (28). Users of the admin group may perform tasks requiring root privileges by authenticating themselves to Aqua tools, like the *Software Update* preference pane, using Mac OS X’s Security framework. Mac OS X also includes the sudo command-line utility that allows users to execute commands as the superuser, root, or any other user. Only users authorized in /etc/sudoers may use sudo, and this configuration file allows fine-grained control of which programs may be run by which users. By default, users of the admin group may

execute any program with root privileges using sudo.

As Darwin, Mac OS X's core, is based on BSD, X follows the BSD convention that only users belonging to group wheel can use the su command to become root user if the root account has been enabled. When the *User* preference pane is used to create a new user, that user is placed into the groups wheel and admin if the "Allow user to administer this computer" checkbox is checked. Administrators should verify that only authorized users are members of these two groups.

In addition, many programs owned by user root and group wheel have the set-UID bit (SUID) or set-GID bit (SGID) turned on, granting root or group wheel privileges to any user who executes these programs. Preston Norvell (41) does an excellent job of describing many of these SUID and SGID programs and explaining which can safely be removed, made unprivileged (by removing set-UID or set-GID bits using chmod), and which should be left alone. It would be wise to take inventory of these programs and occasionally verify that no new ones appear when searched for using find.

Firewalls

Mac OS X includes a packet-filtering firewall called IP firewall, which is included in many BSD distributions. This firewall allows incoming traffic from any host to any port by default. One can configure it by using the ipfw command-line utility or by using GUI tools like sunburst sunShield (freeware) or Brian Hill's Brickhouse (shareware), or glucose Impasse (shareware). For the intrepid, tutorials exist for setting up IP firewall rulesets using the ipfw ruleset language (17, 22, 35). Regardless of how IP firewall rulesets are created, administrators should familiarize themselves with the ruleset language and the ipfw man page to aid in troubleshooting rules. For administrators seeking something more or something different than IP firewall, three commercial, alternative firewall packages also exist: Pliris Firewalk X 2, Intego Net barrier, and Norton Personal Firewall.

Sunburst sunshield:
http://homepage.mac.com/opalliere/Menu3.html
Brian Hill's Brickhouse:
http://personalpages.tds.net/~brian_hill/brickhouse.html
glucose Impasse:
http://www.glu.com/products/impasse/index.html
Pliris Firewalk X 2:
http://www.pliris-soft.com/products/firewalkx/firewalkx.html
Intego Net barrier:
http://www.intego.com/netbarrier/home.html
Norton Personal Firewall:

http://www.symantec.com/sabu/nis/npf_mac/index.html

System Software Updates

An important part of system administration is keeping systems up-to-date with bug fixes and security updates. Apple has provided a means to notify Mac OS X users of system updates. The *Software Update* preference pane can modify this behavior and download and install software updates. The default configuration is to check weekly for updates. There is no reason not to change this behavior to daily checking in this preference pane. Administrators will want to review the list of updates before installing them because many are localization updates for non-English languages and may be unnecessary.

Although Apple may not provide the latest versions of some of its included open-source programs, like OpenSSH, as quickly as they are released, many of these programs can be downloaded from their respective homes and hand-installed by administrators. Administrators may wish to wait for Apple to include any updated third-party tools into official Mac OS X updates because Apple no doubt performs internal compatibility testing on official updates before distributing them. Information on security updates included in Apple's software updates can be found at http://www.apple.com/support/security/security_updates.html.

Virus prevention

Macs in general are less likely to contract viruses than Wintel systems because the overwhelming thrust of malicious programmers' efforts are targeted at Microsoft Windows on Intel hardware. Administrators should expend a minimum of effort, nevertheless, to mitigate the risks of malicious software by installing commercial anti-virus software. Three commercial anti-virus packages exist for Mac OS X: Symantec's Norton AntiVirus for Macintosh 8.0, Sophos Anti-Virus for Macintosh, and MacAfee Virex X 7.0.

Symantec Norton AntiVirus:
http://www.symantec.com/nav/nav_mac/index.html
Sophos Anti-Virus:
http://www.sophos.com/products/software/antivirus/savmac.html
MacAfee Virex X:
http://www.mcafeeb2b.com/products/virex/default.asp

UNIX Security Tools

Perhaps Mac OS X's greatest strength is its ability to run so many security tools with which many UNIX administrators are already familiar. There are far too

many to list here, but I will include, briefly, some that may be more useful than others. The only notable ones I looked into that will have problems compiling on Mac OS X are tripwire, the popular file integrity monitor, and the latest beta (2.0b1) of PortSentry, the portscan detector.

The fink project (http://www.sourceforge.net/projects/fink) aims to bring many open source UNIX tools to Mac OS X. It uses the Debian Linux package management system. Binary packages available via fink that may interest administrators include, among others, the packet capture library libpcap, the network sniffers ethereal and ettercap, the portscanner and OS fingerprinting tool nmap, and the network Swiss army knife netcat. Before installing fink, you may wish to link /sw/include to /usr/local/include and /sw/lib to /usr/local/lib. Fink only writes to /sw so that it will not trample on anything Apple distributes now or in the future. Apple does not distribute many of the header files or libraries needed to compile some tools. Fink is a helpful tool in obtaining these various libraries and headers. You will want to either alter your include and library search paths somehow or simply link, move, or copy the libraries and headers of packages fink installs to the locations where makefiles will look for them.

The popular intrusion detection system Snort is also available, but it needs libpcap header files that fink does not install. You can download the libpcap tarball from http://www.tcpdump.org/release/libpcap-0.6.2.tar.gz. If you have installed libpcap using fink, you only need to move the libpcap headers to /usr/local/include and compile Snort:

```
>wget http://www.snort.org/dl/snort-1.8.6.tar.gz
>tar zxf snort-1.8.6.tar.gz
>cd snort-1.8.6
>./configure --with-libpcap-libraries=/sw/lib
>make ; make install
```

This will put snort in /usr/local/bin and its man page in /usr/local/man. Besides the snort man page, documentation and setup guides can be found at http://www.snort.org/documentation.html.

Nessus, a network-scanning vulnerability detector, is one of my favorite tools and with a little coaxing you can install it on Mac OS X. Nessus requires the GTK and openssl libraries and headers be installed. Apple ships X with openssl (it is required by OpenSSH), but not its headers. I used fink to get these two packages, among others, and then copied all libraries and headers to /usr/local/lib and /usr/local/include, respectively. I compiled the latest stable version, 1.2.1, by simply issuing "./configure ; make ; make install" in the untarred Nessus directories in the appropriate order (nessus-libraries, libnasl, nessus-core, and then nessus-plugins).

Tools written in Perl should also have little problem running on Mac OS X. Mac OS X ships with Perl, but administrators will likely have to download additional

Perl modules to get some Perl tools running. Good examples of useful Perl auditing tools are swatch, a logfile watcher, and whisker, a CGI vulnerability scanner.

Many other security tools are available for Mac OS X. Anything that will compile on other BSD distributions stands a chance of compiling under Mac OS X. Below I include links to the developers of the tools mentioned above and a few others. This list is not intended to be complete, but merely a starting point for your own research.

Fink, UNIX software for your Mac
http://www.sourceforge.net/projects/fink
tcpdump and libpcap
http://www.tcpdump.org/
Snort, an intrusion detection system
http://www.snort.org
Nessus, a vulnerability scanner
http://www.nessus.org
Ethereal, a network sniffer and traffic analyzer
http://www.ethereal.com/
ettercap, a network sniffer for switched LANs
http://ettercap.sourceforge.net/
Whisker, a CGI vulnerability scanner
http://www.wiretrip.net/rfp/p/doc.asp/i2/d21.htm
Psionic PortSentry
Note: PortSentry 1.1 includes an "osx" build target that works without error, but 2.0b1 will not build on Mac OS X/Darwin
http://www.psionic.com/products/
Osiris, a file integrity checker
http://osiris.shmoo.com/
swatch, a logfile watcher
http://www.oit.ucsb.edu/~eta/swatch/

Useful Security sites

Computer security is not a destination but a journey. Administrators must make themselves as aware as possible of both existing and arising issues. The entirety of this paper was researched using the Internet and experimentation on my Macintosh. In addition to the many links I have already provided above and those provided by Patrick Harris (30), and updated by Roland E. Miller, III (40), I would encourage Mac OS X administrators to add the security sites below to their cache of Internet security resources.

CERT Coordination Center	http://www.cert.org
Packet Storm	http://www.packetstormsecurity.org

Security Focus	http://www.securityfocus.com/
Rain Forest Puppy:	http://www.wiretrip.net/rfp/
#RootPrompt.org	http://rootprompt.org/
Security Geeks	http://securitygeeks.shmoo.com/

Conclusions

Mac OS X is fairly secure out-of-the-box. Apple has disabled all inetd-launched daemons and it is the province of administrators to enable any needed services. Apple has conveniently configured inetd.conf to wrap daemon requests with TCP wrappers, but administrators must create hosts.allow and hosts.deny files. Mac OS X's portmap daemon makes libwrap calls and uses these configuration files. As of Mac OS X 10.0.1, Apple includes OpenSSH and the *Sharing* preference panel enables sshd rather than telnetd. Apple ships Mac OS X with the root account disabled and provides sudo for more fine-grained privilege control than the su/group wheel scheme. Apple includes IP firewall, the packet-filtering firewall used by other BSD distributions. The *Software Update* preference pane provides a simple and schedulable method to check for system software updates. Apple provides a utility to set an Open Firmware password—a modification that prevents most methods of circumventing filesystem protection mechanisms.

Apple has some work to do, however, to make Mac OS X more secure. They should provide a mechanism to hide password hashes and other sensitive data from unprivileged processes and users. Mac OS X needs a mechanism to enforce strong password creation. They should adopt more computationally-intensive password hash algorithms like MD5 and Blowfish. Apple should also re-evaluate which included programs require root privileges through the use of the SUID and SGID mechanisms.

The responsibility for implementing some security measures must fall upon system administrators. They should determine whether the type/creator code document-to- application association scheme is needed and, if not, reinstall Mac OS X on a UFS volume. They should set an Open Firmware password. Administrators should disable both automatic login and the display of usernames in the login window. They should lock the cases of Macintoshes. They should get authorization to perform password-strength auditing and schedule regular password audits. To restrict access to network daemons, administrators should both create the TCP wrapper configuration files hosts.allow and hosts.deny and configure firewall rules for IP firewall. *Software Update* should be scheduled to check for system updates daily instead of the default weekly interval. A final measure that administrators may take is to install open-source security tools (e.g. intrusion detection systems like Snort or PortSentry). UNIX administrators given charge of Mac OS X systems will appreciate that many popular open-source security and security-auditing tools

are available for Mac OS X.

The introduction of Mac OS X version 10.0 seemed rushed and more driven by the need to get the product out the door, before it was feature complete. Mac OS X 10.1 delivers the feature-complete version that should have been released as version 10.0. I am confident that Apple's developers will now shift their focus more toward refinement and, hence, the improvement of Mac OS X's security. Apple has said that Jaguar, the next version of Mac OS X, will include BSD updates, IPSec integration, and a VPN client. There may be other updates that are not "buzzwordthy," and we can hope that Jaguar will bring X closer to par with its BSD Brethren in terms of security.

References

1. Anguish, Scott. "Building OpenSSH 3.0.2 on Mac OS X 10.1.1." 9 Mar. 2002. URL: http://www.stepwise.com/Articles/Workbench/2001-12-17.01.html (5 June 2002).

2. Anguish, Scott. "Mac OS X 10.1 Local Security Exploit." 20 Oct. 2001. URL: http://www.stepwise.com/Articles/Admin/2001-10-15.01.html (5 June 2002).

3. Apple Computer, Inc. "A look inside Jaguar." URL: http://www.apple.com/macosx/newversion (3 June 2002).

4. Apple Computer, Inc. "An Introduction to Mac OS X Security." URL: http://developer.apple.com/internet/macosx/securityintro.html (5 June 2002).

5. Apple Computer, Inc. "Macintosh: How to Use FireWire Target Disk Mode." 20 May 2002. URL: http://docs.info.apple.com/article.html?artnum=58583 (5 June 2002).

6. Apple Computer, Inc. "Mac OS X 10.0: AirPort Does Not Work From UFS Partition." 21 Mar. 2002. URL: http://docs.info.apple.com/article.html?artnum=106252 (5 June 2002).

7. Apple Computer, Inc. "Mac OS X 10.0: Choosing UFS or Mac OS Extended (HFS Plus) Formatting." 9 May 2002. URL: http://docs.info.apple.com/article.html?artnum=25316 (5 June 2002).

8. Apple Computer, Inc. "Mac OS X 10.0: Classic Does Not Work From a UFS Disk on First Use." 1 Jun. 2001. URL: http://docs.info.apple.com/article.html?artnum=106277 (5 June 2002).

9. Apple Computer, Inc. "Mac OS X 10.0: Startup Volume Is Named '/' Instead of

'Mac OS X.'" 21 Mar. 2001. URL:
http://docs.info.apple.com/article.html?artnum=106191 (5 June 2002).

10. Apple Computer, Inc. “Mac OS X 10.1: Binding Local NetInfo Database to an NIS Domain.” 2 Nov. 2001. URL:
http://docs.info.apple.com/article.html?artnum=106499 (5 June 2002).

11. Apple Computer, Inc. “Mac OS X 10.1: How to Set up Open Firmware Password Protection.” 22 Feb. 2002. URL:
http://docs.info.apple.com/article.html?artnum=106482 (5 June 2002).

12. Apple Computer, Inc. "Mac OS X: How to Change or Reset a User's Password." 25 Mar. 2002. URL:
http://docs.info.apple.com/article.html?artnum=106156 (5 June 2002).

13. Apple Computer, Inc. “Mac OS X Server 1.x: About Security With Mac OS X Server.” 13 July 2001. URL:
http://docs.info.apple.com/article.html?artnum=60112 (5 June 2002).

14. Apple Computer, Inc. “Mac OS X Server 1.x: Lookupd Release Notes.” 28 Nov. 2001. URL: http://docs.info.apple.com/article.html?artnum=24902 (5 June 2002).

15. Apple Computer, Inc. "Mac OS X: Sherlock Cannot Open Enclosing Folder on UFS or NFS Volume." 13 Feb. 2002. URL:
http://docs.info.apple.com/article.html?artnum=106739 (5 June 2002).

16. Apple Computer, Inc. “Open Firmware Password 1.0.2: Information and Download.” 22 Feb 2002. URL:
http://docs.info.apple.com/article.html?artnum=120095 (5 June 2002).

17. Arentz, Stefan. “Building your own personal firewall.” 9 Oct. 2000. URL:
http://wopr.norad.org/articles/firewall/ (5 June 2002).

18. Arentz, Stefan. “Mac OS X – Apache & Case Insensitive Filesystems.” 10 June 2001. URL: http://www.shmoo.com/mail/bugtraq/jun01/msg00098.shtml (5 June 2002).

19. Bertram McGrath. “Securing FreeBSD Under Mac OS X.” 30 Sept. 2001. URL: http://rr.sans.org/mac/freebsd.php (5 June 2002).

20. blb. “Missing libpcap headers.” 14 Sept. 2001. URL:
http://www.macosxhints.com/article.php?story=2001091304413237#comments (5 June 2002).

21. CodeSamurai. “Open Firmware Password Protection.” URL:

http://www.securemac.com/openfirmwarepasswordprotection.php (5 June 2002).

22. Cote, Daniel. “Setting up firewall rules on Mac OS X 10.1.” 8 Jan. 2002. URL: http://www3.sympatico.ca/dccote/firewall.html (5 June 2002).

23. Curator and The Shmoo Group. “An Unofficial Xinetd Tutorial.” URL: http://www.macsecurity.org/resources/xinetd/tutorial.shtml (5 June 2002).

24. Dino Amato, “adduser_OSX for MacOSX v1.1.” 30 June 2001. URL: http://www.brownnut.com/adduser_OSX.htm (5 June 2002).

25. Faby, Aaron. “Apple NetInfo Network Management System.” 18 Oct. 2000. URL: http://www.aaronfaby.com/NetInfo.rtf (5 June 2002).

26. Freaks’ Macintosh Archive. “Cracking [k] modified.” URL: http://freaky.staticusers.net/crack.shtml (5 June 2002).

27. Freaks’ Macintosh Archive. “Newest Files, Updated and News.” URL: http://freaky.staticusers.net/prevupdate-3.shtml (5 June 2002).

28. Griffiths, Rob. “Enabling the root password (three ways).” 24 Mar. 2001. URL: http://www.macosxhints.com/article.php?story=20010324095804436 (5 June 2002).

29. Grungie. “Macintosh Hacker’s Workshop.” URL: http://grungie.code511.com/MHW_11_Doc/index.html (5 June 2002).

30. Harris, Patrick. “Macintosh Internet Security Basics.” 15 Sept. 2000. URL: http://rr.sans.org/mac/mac_sec.php (5 June 2002).

31. Hill, Brian R. “Authentication and Authorization using the Security Framework.” 28 May 2001. URL: http://www.stepwise.com/Articles/Technical/2001-03-26.01.html (5 June 2002).

32. Jan Verhoog, Gert. “Setting up a network of ‘thin’ Mac OS X clients.” Revision: 1.1.1.1. 12 Feb. 2002. URL: http://www.phil.uu.nl/~gjv/comp/thinosx/thinosx.html#ChangeThePermissionsOfTheNITools (5 June 2002).

33. Kershaw, Michael. "Linux 802.11b and wireless (in)security." 4 Mar. 2002. URL: http://www.linuxsecurity.com/feature_stories/wireless-kismet.html (5 June 2002).

34. Lavigne, Dru. “Adding a User to FreeBSD – Part Two.” 10 Jan. 2001. URL:

http://www.onlamp.com/pub/a/bsd/2001/01/10/FreeBSD_Basics.html (5 June 2002).

35. Lavigne, Dru. “BSD Firewalls: IPFW Rulesets.” 10 May 2001. URL: http://www.onlamp.com/pub/a/bsd/2001/05/09/FreeBSD_Basics.html (5 June 2002).

36. Mahoney, Bob. “Welcoming MacOS X: An Example of Practical Threat Assessment in a University Environment.” 18 June 2001. URL: http://rr.sans.org/mac/macosx.php (5 June 2002).

37. Majka, Mark. “Re: nidump and passwd.” 20 June 2000. URL: http://www.darwinfo.org/devlist.php3?number=521 (5 June 2002).

38. Malayter, Ryan. “[RC5] MacOS G4 AltiVec client in Beta.” 30 Nov. 1999. URL: http://lists.distributed.net/hypermail/rc5.Nov1999/0128.html (5 June 2002).

39. Mate, Jeremy. “Sendmail on Mac OS X.” URL: http://www.sial.org/sendmail/macosx/ (5 June 2002).

40. Miller III, Roland E. "Mac OS X 10.0 Security Essentials." 21 Aug. 2001. URL: http://rr.sans.org/mac/OSX_sec.php (5 June 2002)

41. Norvell, Preston. “Improving the Security of a Default Install of Mac OS X (v10.1).” 5 Mar. 2002. URL: http://rr.sans.org/mac/default_install.php (5 June 2002)

42. Sanchez, Wilfredo. “The Challenges of Integrating Unix and the Mac OS Environments.” Version. 1.10. 5 May 2002. URL: http://www.mit.edu/people/wsanchez/papers/USENIX_2000/. (3 June 2002).

43. Sato, Hiroyuki. “_shadow_xxx does not work NetInfo on MacOS10.1.4.” 8 May 2002. URL: http://www.darwinfo.org/devlist.php3?number=16858 (5 June 2002).

44. Security Focus Online. “Apple Mac OS X nidump Password File Disclosure Vulnerability.” 4 Sept. 2001. URL: http://online.securityfocus.com/bid/2953/info/ (5 June 2002).

45. sleight@shellyeah.org. “ni.patch.” URL: http://sleight.port5.com/dl/ni.patch (5 June 2002).

46. SolarfluX. “Changing the Default Password Encryption Algorithm.” 21 Mar. 2002. URL: http://bsdvault.net/sections.php?op=viewarticle&artid=89 (5 June. 2002).

47. Stratton, Jerry. “Compiling Crack on OS X.” URL: http://cerebus.sandiego.edu/~jerry/UNIXTips/Crack.html (5 June 2002).

Last Updated: July 23rd, 2017

Upcoming Training

SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
Community SANS Trenton SEC401	Trenton, NJ	Aug 21, 2017 - Aug 26, 2017	Community SANS
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor