Improving the Security of a Default Install of Mac OS X (v10.1)

Preston Norvell
GSEC Practical (v. 1.3)

Summary

While bringing the power of Unix and the usability of Mac OS into a single operating system allows users and administrators to explore new levels of functionality and stability, Mac OS X also brings the potential of Unix-like operating systems' darker side in regards to security. Apple has done a good job in giving administrators a reasonably secure base from which to start, but more can be done. The onus is on administrators to take these further steps. While some procedures are typical of any Unix-like operating system, others are peculiar to Mac OS X. This paper will take administrators through the processes, both common and unique, of providing a more secure installation of Mac OS X.

Introduction

In March of 2001, Apple Computer finally delivered their long-promised "next generation" operating system, Mac OS X. It brought to their stable an operating system with an impressive list of features including pre-emptive multitasking, protected memory, and symmetric multiprocessing support. Adding to these industry buzzwords was the fact that it was the first commercial consumer operating system to have an Unix-like core. This BSD-based core was turned into an open-source project by Apple, which they called Darwin. Since then, Apple's Ernest Prabhakar has gone on to claim BSD as being three times more popular on the desktop than Linux (due primarily to Apple's shipping of Mac OS X standard on all their computers) (15).

Of course, with this entirely new operating system, came new security issues. Previous versions of Mac OS (9.x and earlier, now commonly referred to as the Classic OS) were not generally considered a good target of opportunity by typical hackers. The lack of an easily accessible or well-understood command line, or a standard set of networking services and capabilities conspired to keep the OS off most radar screens in this respect. With the new and powerful foundations brought by Mac OS X, came the potential for abuse by those willing to take advantage of the newfound capabilities. The well-liked Unix core that allows users to compile many BSD-compatible programs with ease also allows many exploits to be compiled or created for it with ease.

Fortunately for us, Apple has avoided many of the pitfalls of past Unix distributions (1). The chief among these is that services are turned off by default. No telnet, httpd, sendmail, or simple services on and running to be compromised when the machine is put on a network. But no system is perfect, and Mac OS X is not without its faults, and it can be made even more secure than it is as it ships. Version 10.1, which shipped in late-September 2001, is the latest major release of Mac OS X and is what I will focus on for the remainder of this paper. This version brings a number of feature and performance improvements, and in many cases it is the minimum version supported by many third-party developers. Additionally, there are two versions of Mac OS X: Mac

OS X and Mac OS X Server. Server brings additional functionality (like various Internet services and network and workstation management tools), as well as a higher price tag. This paper will base itself in Mac OS X; Mac OS X Server, which is the same OS with other services added, undoubtedly adds additional considerations those entailed here.

Among the topics I will cover are patching the operating system, and it's default applications, account management, auditing and modifying SUID and SGID programs, securing server functions, and additional considerations for future administration and maintenance of the system.

Patching the system

As with any operating system, one of the first steps in securing a newly installed machine is installing any required patches from the manufacturer. In the case of Mac OS X this means either running Software Update, or downloading the update(s) from Apple's support site (11). At the time of this writing the most current revision is 10.1.3, which requires Security Update 10-19-01 and Installer Update 1.0.

Mac OS X lacks a robust package management system that allows an administrator to query which patches are applied, or which files are affected by patch. This is done with the 'lsbom' command line utility. The .pkg package files used by Apple are Mac OS X bundles that through the Finder appear as files, but which in reality are directory objects. Packages downloaded and installed by Software Update are copied to the /Library/Receipts directory.

```
[localhost:] nephir% cd \
/Library/Receipts/MacOSXUpdate10.1.3.pkg/Contents/Resources
[localhost:] nephir% lsbom –p MUGf MacOSXUpdate10.1.3.bom
drwxrwxr-t      root    admin   .
drwxrwxr-x      root    admin   ./Applications
drwxrwxr-x      root    admin   ./Applications/Acrobat Reader 5.0
drwxrwxr-x      root    admin   ./Applications/Acrobat Reader 5.0/Contents
-rw-rw-r--      root    admin   ./Applications/Acrobat Reader
5.0/Contents/PkgInfo

…
```

Table 1 – Example use of 'lsbom'

Here we see a brief snippet of the output from 'lsbom' as used on the latest minor operating system revision. The output is intuitive to most administrators who have seen the output of the 'ls' command line utility. The permissions on the left will be applied during the install, and the username and group show the owners that will be applied.

A number of third-party developers ship their applications packaged using Apple's packaging tools. There are known problems with Apple's packaging system that can allow an improperly created package to overwrite permissions, and delete symbolic links (2). A concerned administrator can access the contents of the package as in the example above, and determine how their system is likely to be affected.

Currently, there is one additional patch that needs to be applied, the Internet Explorer 5.1 Security Update. A full list of updates available for Mac OS X v10.1, and the order in which they should be applied is available on Apple's support website (11).

Account Management

Depending on how you receive the default install of Mac OS X, it may be set to automatically log in a user upon startup. This is generally considered contrary to good security policy. To change this setting, one must open System Preferences and select the Login icon. When the panel resolves, select the Login Window tab. On this page, there's a check box labeled "Automatically log in"; deselect it. Below is a radio button group labeled "Display Login Windows as: "; administrators may want to make sure this is set to "Name and password entry fields" so usernames are not enumerated at the login screen.

The Users panel in System Preferences manages user account in Mac OS X. Using the UI, there are only two different types of users those who can administer the computer, and those who cannot. There is a check box in each user's definition that specifies whether they can administer the computer. By default the root account familiar to most Unix administrators is disabled. Those needing to execute applications as root, must instead use the 'sudo' utility. To allow finer control with groups, administrators must currently add and populate them manually using NetInfo Manager. Mac Observer has a nice article on the steps involved (7).

As an added step in account safety, it is best to set the computer's screen saver to ask a user for their authentication information when waking the screen saver. This is done through the Screen Savers panel of System Preferences. To enable this feature, select the "Use my use account password" radio button from the Activation tab.

Once patches have been installed and accounts are managed, administrators should move on to permission management, so users can only access the applications and data they need and should.

SetUID and SetGID applications

One of the main areas of concern regarding permissions on Unix systems is Set-UID (aka SUID) and Set-GID (aka SGID) programs. Mac OS X, like most systems, has a number of them, many of which do not need SUID for normal operations and several that do not need to exist at all on most systems.

Before I move on to dealing with the applications lets pause a moment to discuss what

Set-UID and Set-GID programs are. When a user logs in to a Unix or Windows NT-based system and executes a program or application it generally runs with the same permissions of the user, so whatever application is running can access and modify only the bits of the system to which the user is granted permission. Occasionally, an application needs additional privileges, such as when a user needs to mount removable media, or wants to change their password (here it needs to access the password store, which normal users cannot do for good reason). To allow applications acting on the user's behalf, the additional permissions or privileges they need they can be given Set-UID or Set-GID permissions. This allows a program to run as the user or group that owns the program (in most cases the user root). The problem with SUID and SGID programs is that sometimes they can be tricked into accessing resources that they should not be able to (via poor parameter checking, buffer overflows, etc). For this reason, it is always a good idea for an administrator to take stock of the SUID and SGID programs installed on her systems, and disable those that she does not require (5).

Table 2 contains a list of applications an administrator should consider deleting from their system or, at the least, removing the SUID bit.

```
/bin/rcp
/sbin/rdump
/sbin/rrestore
/usr/bin/rlogin
/usr/bin/rsh

/usr/sbin/sendmail

/usr/bin/chfn
/usr/bin/chpass
/usr/bin/chsh

/usr/sbin/sliplogin
```

Table 2 – Applications to consider removing

The first several applications listed in Table 2 are the infamous 'r*' utilities. They really have no business on modern systems, especially systems used by Apple's typical customer base. Sendmail is too commonly the target of exploits. It's probably a good idea to remove it, even though it is not enabled by default The 'ch*' utilities are usually used to allow a user to change various aspects of their user accounts, many administrators do not want users to have this ability, and in the case of Mac OS X they just don't work, so there's no reason to have them. 'sliplogin' is a program used in conjunction with slip-based network connections. 'sliplogin' can be used corrupt terminal lines, so it is best removed (17).

```
/usr/sbin/netstat
/sbin/ping
/usr/sbin/traceroute

/usr/bin/crontab
/usr/bin/at
/usr/bin/atq
/usr/bin/atrm
/usr/bin/batch

/sbin/dump
/sbin/restore

/sbin/route

/usr/sbin/scselect
```

Table 3 – Programs that should have their SUID bits removed

Table 3 displays a list of programs that should have their SUID bits removed.
In most cases, netstat really doesn't need to be SUID, since non-administrative users should not need to use it. In any case, it is minimally best to set the group to kmem and change it to an SGID executable. 'ping' and 'traceroute' are tools to test network functionality, and are generally needed by non-administrators. The at* commands, as well as crontab (considered a standard tool for most administrators) and batch, are used to submit jobs (applications) for scheduled operation, and are historically sources of problems so if not should have their SUID bits removed. 'dump' and 'restore' are used to interact with filesystem backups on computer; users should not be given access to this functionality. The administrator should also remove the SUID bit from 'route', as it should only be used by an administrator, and even then sparingly. 'scselect' is used in conjunction with setting network locations, ala Location Manager, and is best left alone if you're running a laptop, otherwise it doesn't need SUID.

```
/sbin/mount_nfs
/sbin/mount_smbfs
/sbin/umount
/usr/libexec/load_hdi
/usr/libexec/load_webdav
/usr/bin/smbutil

/usr/bin/passwd
/sbin/shutdown
/usr/bin/su
/usr/bin/sudo
/usr/bin/login
/usr/libexec/authopen
/usr/libexec/chkpasswd

/usr/bin/lpq
/usr/bin/lpr
/usr/bin/lprm

/usr/bin/quota

/usr/sbin/DirectoryService
```

Table 4 – Applications Best Left Alone

Table 4 represents a list of applications that are probably best left alone for one reason or another. The first six applications listed in Table 4 are all involved in mounting or unmounting filesystems. In most cases, users will want to have the ability to mount and unmount removable media at will. Unless you are treating this machine as a pure server, it is a good idea to leave these executables with their SUID bits.

'passwd', 'su', 'sudo', 'login', 'authopen', and 'chkpasswd' are all programs that interact with the authentication mechanisms on the computer; they should not be changed. Some of the printing functionality for Mac OS X is tied to the venerable 'lp*' utilities, so they should be left alone. 'quota' is used in conjunction with directory quotas and is probably ok as is. 'DirectoryService' is used to interact with various directory-based information stores, and should remain as it is.

/Applications/Utilities/Disk Utility.app/Contents/MacOS/Disk Utility
/Applications/Utilities/NetInfo Manager.app/Contents/MacOS/NetInfo Manager
/Applications/Utilities/Print Center.app/Contents/MacOS/PrintingReset
/System/Library/CoreServices/Classic Startup.app/Contents/Resources/TruBlueEnvironment
/System/Library/Filesystems/AppleShare/afpLoad
/System/Library/Filesystems/cd9660.fs/cd9660.util
/System/Library/Filesystems/hfs.fs/hfs.util
/System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/NSLCore.framework/Versions/A/Resources/NSLPlugins/slpdLoad
/System/Library/Printers/IOMs/LPRIOM.plugin/Contents/MacOS/LPRIOMHelper
/System/Library/Printers/Libraries/PrintServer/Contents/MacOS/PrintServer
/System/Library/Printers/Libraries/PrintStarter
/System/Library/Printers/Tools/jobcopyperms
/System/Library/PrivateFrameworks/Admin.framework/Versions/A/Resources/setUserPicture
/System/Library/PrivateFrameworks/Admin.framework/Versions/A/Resources/writeconfig
/System/Library/CoreServices/AuthorizationTrampoline

Table 5 – SUID Items in Bundles

There are a number of SetUID applications (many of them GUI programs) wholly peculiar to Mac OS X; Table 5 is a list of these applications. Disk Utility is Apple's media management utility; it allows an administrator to describe, partition, and format media (including hard drives, disk copy images, CDRW disc, et al.) as well as configure software RAID. NetInfo Manager allows an administrator to modify a NetInfo database, which is a NIS-like configuration and management directory that stores user information (usernames, passwords, group membership, etc), and various machine configuration parameters. These two applications (Disk Utility and Netinfo Manager) may seem like candidates for having their SetUID bits removed, but even though a normally unprivileged user may execute them, the programs themselves use Apple's security frameworks to provide additional authorization (via additional authentication). PrintingReset deletes all printer definitions on the system, but is not executable by ordinary users.

The TruBluEnvironment (aka Classic) is the compatibility environment in which Mac OS 9.x boots as a process under Mac OS X, and in which non-native, legacy applications execute. To pull off this feat of magic, it must be SetUID. While this seems scary, the engineers have put some limits on what the environment can do. As with the disk and NetInfo utilities, I am a bit leery of relying upon the application programmers (the creators of Classic) to maintain appropriate controls on processes (especially as complex as the environment surely is). But, on the other hand, Apple's Security Framework promises to allow just this sort of thing without an exceptional burden on programmers (8). 'afpLoad', 'cd9660.util', and 'hfs.util' are additional methods of mounting filesystems, and are therefore probably best left at the defaults.

'slpdLoad' starts 'slpd', the Service Location Protocol daemon which announces the availability of file and printer shares among other things, with some initial parameters (12). I have not had any problems removing the SUID bit so far, but since there is no documentation for either application, administrators may wish to be cautious about removing the permissions. 'LPRIOMHelper', 'PrintStarter', 'jobcopyperms' and 'PrintServer' appear to be a part of the printing subsystem, but they too lack documentation.

'setUserPicture' appears to do exactly what it says it does, and I was able to set my user picture (the one displayed on the login window) to /etc/master.passwd. Fortunately it didn't bother to display it, or if it did, the text was simply too shrunken to see with the naked eye. I suspect it is an OK idea to remove the SetUID bit from this, as I have had no troubles setting the picture via the System Preferences utility after doing so.

'writeconfig' is another application lacking any documentation, but according to sources, it is responsible for writing out various system configuration changes from the System Preferences (which itself is not SetUID). I would be interested in taking a further look at how this system works, and how System Preferences passes the configuration change to 'writeconfig'. Right now things seem to break pretty well if the SUID bit is removed, so it's best to leave it as is unless the administrator is adventurous.

Last, but not least amongst this parade of SetUID applications is 'AuthorizationTrampoline', which besides having by far the most fun name of any of the applications here and intuitively seeming to have something with authorization processes on the computer, lacks any other documentation.

This brings us to SetGID programs.

```
/sbin/dump
/sbin/rdump
/sbin/restore
/sbin/rrestore

/usr/bin/wall
/usr/bin/write
```

Table 6 – Set GID programs that should be modified

Table 6 lists all of the SetGID applications on a Mac OS X v10.1.3 system that should probably have their SGID bits removed ('chmod g-s applicationpath'). The first four we've discussed before, and if they haven't been wholesale deleted should be appropriately modified. The final two utilities, 'wall' and 'write' are used to send messages to users' terminal sessions (such as one would obtain with Terminal.app), and have had problems in the past (the least of which is the utter annoyance at having a terminal session interrupted by some other user on the system).

```
/usr/bin/lpc
/usr/bin/lpq
/usr/bin/lpr
/usr/bin/lprm
/Applications/Utilities/Print Center.app/Contents/MacOS/Print Center
/System/Library/Printers/Libraries/PrintStarter

/System/Library/SystemConfiguration/PrinterNotifications.bundle/Contents/Mac
OS/makequeues

/Applications/Mail.app/Contents/MacOS/Mail
/usr/bin/mail

/System/Library/Filesystems/AppleShare/check_afp.app/Contents/MacOS/chec
k_afp

/usr/bin/fstat
/bin/df
/sbin/dmesg
/usr/bin/nfsstat
/usr/bin/uptime
/usr/bin/w
/usr/sbin/iostat
/usr/sbin/lsof
/usr/sbin/pstat
/usr/sbin/trpt
/usr/sbin/trsp
```

Table 7 – SetGID applications that can remain unchanged

The remaining SetGID applications are listed in Table 7. These are all probably fine left they way they are. If an administrator is concerned that these programs may be used for nefarious purposes, she can remove the users read and execute bits from the executables ('chmod o-rx applicationpath'). Listed here are printing utilities (most of which we've already discussed), mail applications, and general system health monitors.

Another area of concern on default installs with regards to permissions are the legacy files and folders used by the Classic environment (as well as any other pre-existing files and folders on a machine that has been upgrade to Mac OS X). By default all of the legacy files and folders are readable and writeable by the world. It is a very good idea for administrators to go through the legacy directories, move any user data into the appropriate home directories, and modify the permissions appropriately. It is also best to remove world write permissions from the remaining filesystem objects that comprise the legacy system.

Before I get away from permissions and on to services, I should take a moment to talk about NetInfo. As I previously stated, NetInfo is a distributable database containing system and user configuration information. Because of the way it operates users are allowed to have full access to the entire database, which as has been stated in many places allows a user to dump the user database, including password hashes using a simple command ('nidump passwd . /') (13). The passwords can then easily be decrypted. It is possible to configure lookupd (the server that performs lookups against the various data stores for configuration information) to look elsewhere, including using a shadowed password store (i.e., /etc/master.passwd) (6). The fact that this is possible is certainly of great comfort, but I have questions as to how well it will be supported by Apple, and so caution administrators in making such a change. Apple recently published a paper on how to integrate Mac OS X into Microsoft's Active Directory directory service (9). Either solution would certainly provide additional security beyond the current configuration.

Securing Services

Another task administrator should consider before deploying Mac OS X is upgrading the super server (inetd, as in most Unix-based operating systems) to a more secure solution. Inetd exists on the system to listen for connection attempts to services listed in the service database (in NetInfo by default). When an attempt is made to connect to one of these services, inetd checks to see if it is configured to start a server to respond. For various reasons, it makes sense to be able to add additional restrictions to this process (which computers and users can connect, at which times, and to which service, etc.). This is commonly done with a replacement super server named xinetd. Xinetd is freeware and can be downloaded from Xinetd.org and compiled and installed as it ships. Xinetd comes with a utility that creates a fully functional xinetd configuration file from the existing inetd configuration file. Beyond this configuration, the only other function an administrator needs to perform upon installation is to create a startup script for the new server. There is a readily available tutorial on MacSecurity.org that includes most of what an administrator will need to complete the installation process, including generating the necessary startup script (14).

If you need to add a service, and it can't be run from the super server (this includes servers like Apache and OpenSSH) you can also use Wietse Venema's tcpwrappers. Tcpwrappers is installed by default, though the libraries are not, so if you need to compile a program that requires libwrap you'll have to download and compile tcpwrappers yourself. Stepwise has instructions on their site as part of an article on installing OpenSSH (3).

Another server that is a good idea to upgrade is the rpc portmapper. This server has a history of problems, and though Mac OS X is not configured to run it by default (as there are no services configured to run with inetd either), it is a good idea to install the secure version (which also happens to have been created by Wietse Venema). After all, an administrator never knows when they might need a new service, and it's best to have the appropriate infrastructure in place ahead of time. Stepwise has directions on the installation of this freely available package as well, though they are somewhat out of date (performing the slight modifications required to make it apply to current versions of Mac OS X are left up to the administrator) (4).

Mac OS X, like many BSD variants, ships with a well-known built-in firewall, named IPFW. By default, the firewall is running, but unconfigured. If an administrator has any experience in configuring IPFW, they should have no trouble configuring it on Mac OS X. I highly recommend Brickhouse for those who may not have experience in configuring IPFW. It is a GUI application with many options that will aid an administrator in creating a functional ruleset. It can even help with setting up the Mac OS X computer to share Internet connections via natd. For more advanced configurations, it behooves an administrator to familiarize herself with how to configure and maintain the firewall herself, although even here a tool such as Brickhouse can provide a good base to start from.

Monitoring Critical File Integrity

Whether a Mac OS X computer's role is as workstation, or that of a server, it's important to ensure that the responsible administrator knows when critical system files have been modified. This function is part of the overall function of a typical host intrusion detection system (HIDS). Unfortunately there are currently no fully featured HIDS for Mac OS X. However, there are a number of file integrity checkers available that will allow an administrator to track any changes her system. One I am rather partial to (I must admit to being a bit biased as I was the progenitor of a much distant ancestor, and a friend of it's current code chef) is Osiris. It's a lightweight but capable file integrity checker that compiles for Mac OS X (and Darwin) out of the box. It creates a database of file and directories, including file permissions and checksums. Osiris also comes with a prebuilt configuration file for use with Darwin, which can be easily adjusted to include the additional important directories in Mac OS X (such as /Applications, /Library, /Applications (Mac OS 9), and /System Folder). Osiris is run periodically to generate a database, and then its counterpart, Scale, can compare databases and report on the differences it discovers, alerting an administrator to any unwanted changes.

Combating Viruses

No modern consumer operating system installation should be without some form of virus protection. While having a Unix-based access control system can reduce the risk of damage from worms and viruses (by removing regular users' ability to modify applications and sensitive data), it does not eliminate the risk. No matter how careful an administrator has secured a machine with permissions, there are ways even these controls can be circumvented. It is, therefore, diligent for administrators to install and configure and an anti-virus agent on their systems. Of all the steps (!!!!don't like that word, need another!!!) presented here, this is the only one requiring the purchase of commercial software. Symantec, McAfee, and Sophos are all currently shipping Mac OS X compatible versions of their anti-virus products. The choice of which of these to use is up to the administrator and their budget.

Conclusion

After an administrator has applied all of the appropriate patches, locked down the permissions to their and their users' needs, secured existing and potential services to the best of their ability (and according to their security policy, of course), and begun monitoring critical system files for unwanted changes, and installed an anti-virus solution, there is still more to be done. The virus definitions for whichever solution an administrator chooses must be kept up to date. Operating system and application patches must be run, and file integrity database gathered. On top of this, education must be had for the administrators and the users. To paraphrase an old cliché, the price of continued security is vigilance.

With Mac OS X, Apple has brought forth a powerful new operating system melding the best of both worlds (Mac and Unix), and potentially the worst of both worlds. Fortunately, they've gone a long ways in giving administrators a solid base from which to work. With the exception of the use of the NetInfo database for storing configuration data, Apple has done a remarkably good job in trying to create that balance between usability and security. The Secure Trusted Operating System Consortium has actually chosen Darwin (the open source Unix-based core of Mac OS X, which is an operating system in its own right) as the basis for their attempt to make a secure trusted operating system.

Between the solid base Apple provides and the procedures listed here, administrators can be confident that their systems can provide a secure environment in which their users can work. Their continued vigilance and education should keep it that way.

References

1. "An Introduction to Mac OS X Security." Web Development & Mac OS X. URL: http://developer.apple.com/internet/macosx/securityintro.html (1 Mar 2002).
2. Anguish, Scott. "Beware of Installers bearing package (Part II)." 09 April 2001. URL: http://www.stepwise.com/Articles/Technical/Packages/InstallerOnX.html (1 Mar 2002).
3. Anguish, Scott. "Building OpenSSH 3.0.2 on Mac OS X 10.1.1" 1 Feb 2002. URL: http://www.stepwise.com/Articles/Workbench/2001-12-17.01.html (1 Mar 2002).
4. Anguish, Scott. "Compiling secure portmap." 05 May 2000. URL: http://www.stepwise.com/Articles/Workbench/2000-04-28.01.html (1 Mar 2002).
5. Beale, Jay. "Shredding Access in the Name of Security: Set UID Audits." 14 Sep 2000. URL: http://networking.earthweb.com/netsecur/article/0,,12084_624001,00.html (1 Mar 2002).
6. Braun, Rob. "How do I change the order that lookupd resolves things?" The Unofficial Darwin FAQ 15 Dec 2001. URL: http://www.darwinfo.org/faq.shtml#resorder (1 Mar 2002).
7. D'Addario, Kyle and Wincent Colaiuta. "Unlocking The Power Of Groups In Mac OS X." 27 April 2001. URL: http://www.macobserver.com/tips/hotcocoa/2001/20010427.shtml (1 Mar 2002).
8. Hill, Brian. "Authentication and Authorization using the Security Framework." 28 May 2001. URL: http://www.stepwise.com/Articles/Technical/2001-03-26.01.html (1 Mar 2002).
9. "Integrating Mac OS X With Active Directory." 2002. URL: http://www.apple.com/macosx/server/pdf/MacOSXwithActiveDirectory.pdf (1 Mar 2002).
10. Lavigne, Dru. "An Introduction to Unix Permissions." FreeBSD Basics. 6 Sept 2002. URL: http://www.onlamp.com/pub/a/bsd/2000/09/06/FreeBSD_Basics.html (1 Mar 2002).
11. "Mac OS X 10.1: Chart of Available Software Updates." Kbase. 20 Feb 2002. URL: http://docs.info.apple.com/article.html?artnum=106713 (1 Mar 2002).
12. "Mac OS X: What Are All Those Processes?" 2000. URL: http://www.westwind.com/reference/OS-X/background-processes.html (1Mar 2002).
13. Norvell, Preston. "Netinfo and the passwd database." 19 June 2000. URL: http://www.shmoo.com/pipermail/macsec/2000-June/000044.html (1 Mar 2002).

14. Norvell, Preston. "An Unofficial Xinetd Tutorial." URL: http://www.macsecurity.org/resources/xinetd/tutorial.shtml (1 Mar 2002).
15. Orlowski, Andrew. "BSD '3 times as popular as desktop Linux' – Apple." 14 Feb 2002. URL: http://www.theregister.co.uk/content/4/24060.html (1 Mar 2002).
16. "Security: Mac OS X and UNIX." Web Development & Mac OS X. URL: http://developer.apple.com/internet/macosx/securitycompare.html (1 Mar 2002).
17. "Sliplogin(8)" System Manager's Manual. 5 Aug 1991. Computer Software. "man sliplogin".