Global Information Assurance Certification Paper

Inside Mac Security

GIAC (GSEC) Gold Certification

Author: Ben S. Knowles, adric@adric.net
Advisor: Manuel Humberto Santander Pelaez, Manuel.Santander@epm.com.co

Draft: March 2014

Abstract

Mac OS X includes many unique security technologies ranging from the Keyring system, integrated Kerberos, application and network firewalls, code signing, anti-malware and exploitation resistance technologies, and Internet client application security to many commands (client and server) specific to Macintosh systems that support the security systems including those for certificate management, firewall configuration, drive imaging and encryption. We introduce the Mac OS X security systems and discuss the built-in tools with deliberate focus on the system utilities and command line tools used by security professionals.

1. Introduction

Apple, Inc.'s OS X family is both the result of decades of operating system development and a collection of systems and features from many other systems combined with many unique ideas and implementations. It is necessary to examine the ancestor systems of OS X as well the history of the system to understand the different technologies and how they interact. Understanding the how and often the why of OS X technology is vital to being able to secure it effectively and troubleshoot problems.

The open source core of the system is itself an operating system called Darwin. Darwin is a variant of BSD Unix running on the Mach 3 microkernel from Carnegie Mellon University and the Open Software Foundation (Singh, 2007). Atop Darwin, Apple layers both open source and proprietary code to build a complex and sophisticated operating system that maintains some of the look and feel of classic Macintosh computers with almost no compatibility to historical (1980s) Macintosh underlying technology. In fact OS X is much more compatible with its sibling UNIX systems and with Microsoft Windows than with past Apple computers. And OS X is Unix: a UNIX 03 system as certified by the The Open Group and listed on their register (The Open Group, 2014). OS X moves forward rapidly and sometime controversially, embracing new technologies, devices, and interfaces and leaving others behind. For example as the animations and graphics used in the system became more sophisticated, requirements started to include hardware accelerated graphics capabilities. OS X has even entirely switched architectures once in its brief history, from PowerPC to Intel, which was followed quickly by moving to require 64-bit processors. At the same time iOS shares much technology and code with OS X and runs on still another architecture family (ARM).

The history of OS X is intertwined with that of Apple, Inc., its corporate and technical leadership, and their competitors. As such it is treated commonly more as melodrama or mythology than historical facts. There are even a few movies about the development of the original Macintosh and the life and career of the late Steve Jobs who co-founded Apple Computer with Steve Wozniak in 1976. Amit Singh devotes the first

36 pages of his seminal *Mac OS X Internals: A Systems Approach* (2007) to detailing the political and technological influences that contributed to the design of OS X in its pre-history before detailing that design in unparalleled depth. It is, in its way, a thrilling tale of secret projects, fantastic code names, cutting edge research, and difficult business decisions. Some years and eight OS X releases later Jonathan Levin's *Mac OS X and iOS Internals: To the Apple's Core* (2013) touches briefly on that same pre-history in his first chapter before continuing on through the past of OS X and iOS detailing each release in both systems' history and making some predictions about the future of both platforms. Many writers refer to past technology and obsolete, unavailable systems from this history when explaining either the history or current functions of various aspects of the OS X environment. For example, the decision by Apple to acquire NeXT and NeXTSTEP rather than Be, Inc. and the BeOS certainly affects how OS X was made and continues to develop and is still debated in some corners of the Internet. Some old yet still useful documentation may refer to Rhapsody or the blue or yellow box systems, and a few APIs from the previous incarnations do live on in the system.

Mac OS X includes much essentially unchanged from BSD Unix and many systems familiar to Unix users and administrators are present with very minor changes. One example is logging: OS X uses text logs throughout the system and includes Console.app, a GUI for log searching and viewing. A sophisticated Terminal application is included and many familiar GNU utilities are included with the BSD userland. OS X ships with the Perl, Python, and Ruby scripting languages. This is in addition to the suite of compilers and development tools available in Apple's Xcode package, which is a free download for OS X on the App Store. Networking utilities such as *ifconfig*, *netstat*, and *ping* are available from Terminal and work as expected on a BSD-derived system. Notably, the Network Utility application provides much of the same functionality from a GUI. OS X can even be booted in single user mode to a text console for troubleshooting.

With Xcode added, OS X supports native development of application in C, C++, and Objective C and provides library access and scripting hooks to other languages as well as to the native AppleScript automation language. Many popular free and open

source UNIX (and Linux) software builds easily on OS X systems giving OS X users access to entire universes of software. Several third party projects provide tools for managing the build, installation, and update of these packages including *fink*, *homebrew*, and *macports*.

Other significant technology in OS X comes not from BSD or Linux but from OpenSolaris. The Dynamic Tracing system monitoring and troubleshooting framework (DTrace) was ported from Solaris and included with OS X starting in version 10.5 Leopard. Dtrace provides programmatic live access to kernel and application state and behaviour. In contrast to traditional debuggers or system-call tracing facilities, "DTrace instruments *all* software," as explained on dtrace.org (DTrace, 2014). Apple also includes Instruments a sophisticated development utility based on DTrace in the Xcode suite. The Zettabyte File System (ZFS) from OpenSolaris was also ported by Apple to OS X. Though included in 10.5, Leopard was later removed and never officially supported by Apple. There are many free and commercial third-party filesystem drivers available including ports of the Linux user-space filesystem FUSE project and ZFS is a common topic of discussion (Wikipedia, 2014b).

However, several fundamental operating system components are entirely unique to Darwin (OS X and iOS). Perhaps the most important of these is launchd. First available in OS X 10.4 Tiger launchd replaces at least the rc, init, cron, and inetd systems from traditional BSD systems. It starts and stops services, receives and dispatches network connections, schedules and executes tasks, manages service dependencies, and is the equivalent in status to the *init* super-process on other Unix systems. The program */sbin/launchd* is always process identifier (PID) 1 on Mac OS X and the parent of all other processes. This is explained well in works such as Jepson, Rothman, & Rosen classic *Mac OS X for Unix Geeks* (2008) which explores launchd capabilities, details the uses of *launchctl*, and explains how to add periodic jobs and system start scripts to Mac OS X systems using launchd rather than cron or sysvinit in Chapter 4 (Jepson, Rothman, & Rosen 2008).

Another substantial difference between Mac OS X and traditional Unix systems is

the graphical user interface. The native desktop display and window management system for Mac OS X is not the standard X Window system (originally from MIT's Project Athena) but Apple's proprietary Aqua technology. Essentially all common Mac applications including the entire system as shipped by Apple, any of Apple's other software products, and the vast majority of commercial and open-source software for Mac use Aqua. This includes popular commercial software such as Microsoft Office and Adobe Creative Suite as well as many open source applications, frameworks, and libraries which feature Mac-native Aqua graphics rather than and in addition to X Window including QT and wxWidgets. In fact Apple no longer includes X Window system software with the operating system as an optional component or on the install media. In the most recent releases X users are directed by Apple to download the latest XQuartz distribution from the upstream project. The XQuartz Trac site on Mac OS Forge details the available releases, provides bug tracking and community support, and documents the history of X Window, X11.app, and OS X (XQuartz, 2014).

Many other OS X technologies are unique yet less visible to users. System administrators will find that many of the systems for installing and configuration applications and the system exist solely on OS X including the way bundles, the installer and disk images interact. Common OS X applications without background services, drivers, or other complexity may not use an installer program at all. Upon opening the downloaded disk image (DMG) or ZIP archive users are encouraged to copy the entire application "bundle" to their Applications folder. This bundle is actually just an Apple supported directory structure with the ".app" prefix. System applications display bundles as files rather than folders and they can be dragged and dropped in Finder. This is the install method for many applications today including, for example, Mozilla's Firefox and Thunderbird internet client applications.

Applications which need more control over their installation details or need to modify system files can package their software into PKG files for use with the built-in Installer application. Installer can check hashes and signatures, run pre-install and post-install scripts, request additional information or elevated privileges from users as needed,

and even queue up multiple installations, pausing and starting them as needed until they are completed. Though the distribution of software and some aspects of installation are changing with the availability of the OS X App Store, the underlying systems have changed little.

Traditionally UNIX systems favour configuration of applications and system software with small editable text files, such as those found in */etc*, rather than binary configuration data. Although there are text configuration files in use throughout OS X, the primary configuration format is the property list file (plist) as inherited from NeXTSTEP. In OS X plists come in either a text XML format or the newer binary plist format and, according to the Wikipedia page on Property List, the JSON was added as a supported format in 10.7 Lion (Wikipedia, 2014a) . Console utilities *defaults*, *plutil,* and *PlistBuddy* and developer tools including Xcode's integrated Property List Editor handle these well and are essential in any non-trivial configuration or analysis on OS X (and iOS) systems.

Some of OS X most sophisticated security features embody this mix of inheritance and innovation. Apple (2014a) has introduced process isolation, privilege separation, and mandatory access controls in OS X using software from the TrustedBSD project and their own proprietary technology. These technologies are integrated into the system and used by much of the core operating system and included utilities. Apple (2014a) advocates the use of these security technologies to developers and markets OS X on its security identifying them with the feature names Application Sandbox, XPC, and Gatekeeper (Apple, 2014e).

In addition to its role as the platform of choice for many technology and security professionals, OS X is tightly related to sibling system iOS and the two share much of their core technology. As detailed extensively in Jonathan Levin's book *Mac OS X and iOS Internals: To the Apple's Core* (2013) and the SANS Institute's forthcoming course *FOR518: Macintosh and iOS Forensic Analysis* much of the underlying system is identical between the two Darwin based systems. From the xnu kernel and Mach, the filesystems, development toolchains, libraries and application formats, up to the user

interfaces the two Darwin siblings are mostly identical. The user interfaces and the security functionality of iOS systems are where some of the greatest contrast is found, though even they share foundations with those of OS X. In recent OS releases, features and applications from iOS have been ported "back" into OS X including the Notification system and Launchpad application launcher. Other features are developed and released in tandem such as improvements to Safari and Mobile Safari and the roll out of the various iCloud services. OS X is also the required development platform for all iOS software, though some third party tools exist on other platforms.

As they transitioned from the original "fast cats" code names used for OS X releases as well as from PowerPC to 64 bit native Intel architecture, Apple formally and publicly adopted an annual release cycle for OS X matching that of sibling system iOS. In the same series of releases, updates to the system software became available only through the Apple App Store and all but ceased to be available on physical media. With the release of 10.9 Mavericks in late 2013 the Macintosh system software now promoted by Apple as simply "OS X" is a free download through the App Store to upgrade any compatible Macintosh computer.

Mac OS X Server was a distinct product for most of the first part of OS X history with a completely different pricing structure organized around the number of supported users not unlike competing systems. Server also required a license key, though OS X client never has. Although it consisted primarily of additional network services, directory system infrastructure, and the proprietary Apple tools to configure them, which could all be installed manually on a client system, OS X Server also had some significant differences including a different kernel with server optimizations. Apple once manufactured XServe servers and XRaid disk arrays that came pre-installed with and supported on Mac OS X Server. OS X server on Macintosh or XServe became an effective workgroup server for many smaller organizations and was also integrated into some enterprise environments. It continues to see use in these roles as evidenced by discussions and conferences in the Mac sysadmin and security community.

After Apple discontinued their computer server hardware products (leaving only

the Mini) and with the release of 10.7 Lion OS X Server instead became an inexpensive Server "app" available in the Apple OS X App Store containing many though not all of the configuration utilities and additional services. Server 1, 2 , and 3 were released simultaneously with OS X 10.7 – 10.9 and each adds more workgroup services. OS X Server has always been the canonical and supported way to manage Macintosh computers on a network and in recent releases of Server that functionality has been expanded to include managing devices running Apple's iOS such as iPhones and iPads.

The complexity and versatility of OS X have made it popular among technology enthusiasts, developers, and information security professionals. With understanding of the inner workings of the integrated security systems of OS X technologists and security professionals can take advantage of its unique qualities to protect users and environments and get better use from their own Macintosh systems for security tasks. A detailed examination of the core security technologies in OS X follows.

2. Mac Security systems

2.1. Remote Access

Mac OS X includes a set of standard systems for securing remote access including SSH and a built-in VNC client in addition to support for common file-sharing protocols. Apple offers Apple Remote Desktop as a separate product for system management. Remote Desktop clients for Microsoft Windows systems are available online and OS X includes clients for popular VPN services. The OS X Server product adds a set of Web-centric workgroup applications and server management utilities which can be made remotely accessible and integrate with Apple's own online services.

2.1.1 SSH

OS X includes SSH from the OpenSSH project under a BSD license and provides source for their version online along with their other open source publications (Apple, 2014d). SSH remote access is enabled in OS X using the Sharing applet in System Preferences where it is listed simply as Remote Login. The SSH service can be enabled

and disabled and the current hostname or primary IP address are displayed for reference. A chooser panel allows the selection of authorized local users and groups for SSH. Toggling the setting or adjusting the authorized users there modifies and reloads the appropriate launchd configurations in the background and takes immediate effect.

Remote Login in Sharing – System Preferences.app – Mavericks

SCP and SFTP subsystems are enabled and most of the configuration for sshd is from upstream with the Kerberos options enabled. The standard configuration file is in */private/etc/sshd_config*. If XQuartz is installed it adds a stanza to sshd-config for X11 authorization. Remote X application display then works as expected for X11 applications using ssh and X forwarding.

Standard command line *ssh, scp*, and *sftp* clients are included and the built-in Terminal application can be used for remote connections using these protocols as well as the old and insecure telnet and ftp.

Ben S. Knowles, adric@adric.net

2.1.2 Screen Sharing

OS X has included a simple native VNC client since version 10.5 (Edge, Jr. et al., 2009). It is identified as Screen Sharing in the GUI and documentation. When enabled, it works between Mac systems. Screen Sharing is integrated into Finder such that clicking on the icon of a Macintosh network host may attempt a connection and prompt for login. It is configured in the System Preferences Sharing pane much as Remote Login (SSH) is and is reasonably compatible with other VNC software. As with all VNC applications, the built-in authentication and connection security is minimal and should be supplemented by running it over an SSH or VPN tunnel as well as additional controls based on the assessed risks.

As part of the rollout of the newer version of Apple's online iCloud services, OS X gained iCloud file access and the ability to proxy Internet requests to a Macintosh with the “Back to My Mac” feature. Many of these services were previously available to paid subscribers of the Dot Mac service, a predecessor to iCloud. These heavily marketed services exclusively use Apple’s internet servers to store documents and proxy connections.

Apple (2014f) sells Apple Remote Desktop (ARD), an enterprise computer management system for Macintosh systems which also uses the VNC client functionality. Before Screen Sharing was included into OS X, an ARD client install was required for ARD system management. ARD version 3 goes well beyond just desktop sharing and remote login and provides asset inventory and configuration management, software installation and upgrades, user and system monitoring and extends OS X automation and scripting to managed systems by groups. An ARD administrator can also enable VNC-only access or view to managed systems with a administrator set credential. ARD is available in the Apple App Store and its features are explained in full at the ARD website (Apple, 2014f).

Although not included in Mac OS X, a native client for Microsoft Remote Desktop Protocol is developed and distributed by Microsoft. The current version is distributed on the Mac App Store. Previous versions were and are downloadable directly

from Microsoft but are not compatible with recent versions of OS X. In particular the older RDC application and some popular open source alternative applications are not able to negotiate a secure connection with the most recent Windows servers. The download site at Microsoft directs users to Remote Desktop on the App Store.

2.1.3 VPN client

OS X includes support for a few common VPN client types. Microsoft's PPTP and the L2TP variant of IPSec have long been supported and native support for Cisco IPSec VPN was added in 10.6 Snow Leopard (Edge, Jr, et al., 2010).

VPN connections are configured as a network connection in System Preferences GUI and the details needed depend on the VPN type to connect. Underneath that, the utilities and configuration also vary by VPN type. The IPSec implementation in OS X was originally from the KAME IPv6 project and many pieces are in OS X essentially unaltered and will be familiar to users of other IPSec and IPv6 systems such as those in FreeBSD and Linux.

2.1.4 Server

Installing the Server application for the App Store onto Lion, Mountain Lion, and Mavericks (10.7-10.9) systems adds an array of workgroup services which can all be made remotely accessible to local and remote networks. Many of the core service daemons are already part of OS X (or trivially installable) and the differentiator of Server.app (and the obsolete OS X Server system) is in providing integrated management and configuration utilities. For example, Mac OS X can share files simply with SMB/CIFS to other network devices but Server allows the configuration of complex share permissions across multiple protocols using directory services users and groups.

In the current releases these utilities are all essentially contained within Server.app though previously they were separate applications. All of the command line service management utilities (standard and Apple proprietary) are also now contained within the Server.app bundle making it simple to locate and identify the relevant utilities and libraries in */Applications/Server.app/Contents/ServerRoot/* . Many of the

configuration files and data are also there though some important ones are in the System domains of the OS X filesystem.

Server provides a set of core infrastructure services for network access, name resolution, and directory services. Basic communication and collaboration services are provided including servers for e-mail and chat. Workgroup services such as shared calendars are delivered by or integrated with web applications. Software update packages, and in the most recent release, Apple internet content can be cached and managed for network clients. Remote installation and management of Macintosh, and more recently iOS systems, are supported by a collection of services. Server adds VPN connection server capabilities to those included in OS X and supports PPTP and LT2P (IPSec) connections authenticated by Open Directory.

Apple routers can also be managed directly with the Server tools. This is integrated with service configuration such that administrators enabling services are asked if they would like the service made available to the Internet. If confirmed the router's firewall configuration will then be updated automatically to expose the service which is convenient for low-risk environments and easy experimentation.

Server can be registered with Apple Internet hosted servers for push notifications to compatible devices. This allows your workgroup server to remotely notify users through the Internet of events of interest on the local network such as a received e-mail message. This works via TLS certificates registered under an administrator's Apple ID. Once enabled through the Server app an email like this is sent:

```
        Dear Ben Knowles,

        The following Apple Push Notification Service certificates have been created for
AppleID adric@adric.net and will expire on November 17, 2014.

        lorelei.local - apns:com.apple.calendar

        lorelei.local - apns:com.apple.contact

        lorelei.local - apns:com.apple.mail

        lorelei.local - apns:com.apple.mgmt

        lorelei.local - apns:com.apple.alerts

        Thank You,
```

Ben S. Knowles, adric@adric.net

```
Apple Push Notification Service
```

Sample Apple Push Notification Service message – Server 2

2.1 Firewalls

Currently, Mac OS X includes two sophisticated firewall systems and installing Server adds one more. The traditional BSD firewall ipfw originally included in OS X has been replaced with OpenBSD's pf for the standard packet filtering network firewall though it is not enabled by default. Instead an application firewall has been included and enabled by default in some capacity since version 10.5.1 (Apple, 2010a). Server adds the adaptive firewall with sophisticated policy configuration and management through Server.app.

2.1.5 Packet Filtering

The transition from ipfw to pf is nearly complete. *pf* already in use as early as 10.7 Lion. The manual page for *ipfw* is clearly marked as deprecated, recommending *pfctl* instead, and in 10.9 Mavericks only the deprecation warning remains. Upstream FreeBSD and OpenBSD resources document both systems well, but OS X has some unique configuration and a few changes from upstream.

The upstream pf documentation from the OpenBSD project is extensive, detailed, and an excellent resource for the utilities and configuration language of the current version. Unfortunately, OS X uses an older version and there are some feature mismatches between OS X pf and the current version of pf on OpenBSD and other systems. A detailed explanation of the differences is available in an upgrade note for OpenBSD 4.7 where the change occurred upstream (OpenBSD, 2009).

Apple ships some pf rules for the built-in systems in OS X. Expressed in pf as anchors these are defined in */etc/pf.anchors/com.apple* and are explained by one of the *Enterprise Mac Security* authors Charles Edge, Jr. on his web log in the post "A Cheat Sheet For Using pf in OS X Lion and Up" (2012). Installing Server adds additional anchors for the added Apple services including the adaptive firewall.

OS X also has dummynet enabled in ipfw. As noted in *Enterprise Mac Security* chapter on firewalls, dummynet is a traffic shaping system that can be used for throttling and quality of service (QoS) policy enforcement (Edge, Jr. et al., 2009). It has upstream documentation including the manual page available online at the FreeBSD project.

2.1.6 Application layer firewall

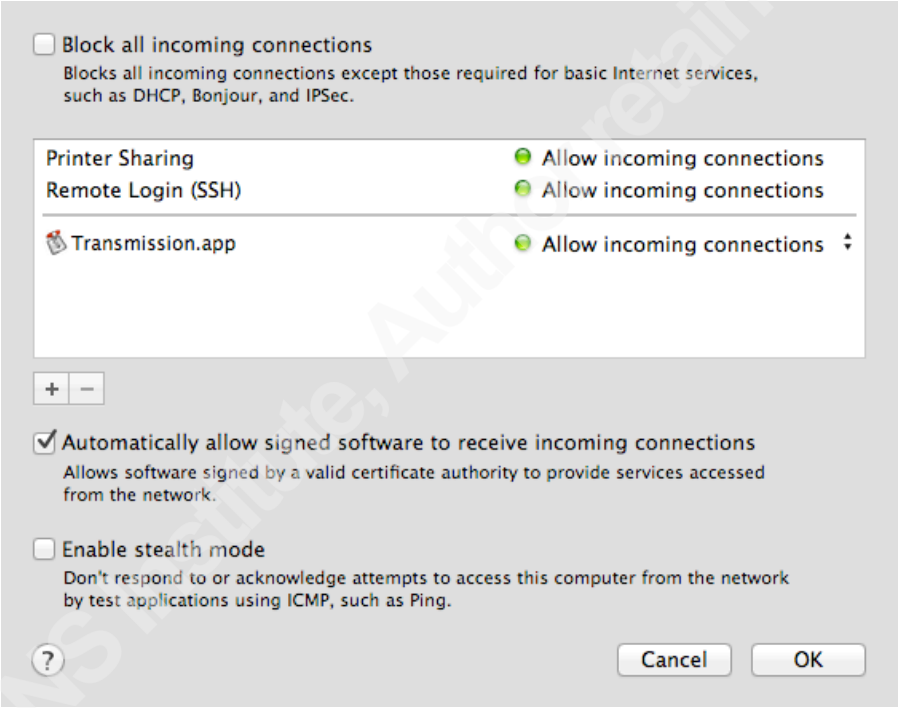
Instead of working on network addresses, protocols, and port numbers (as ipfw and pf do) the application layer firewall (ALF) in OS X considers applications in deciding what connections to allow. Another important difference is that ALF is entirely concerned with incoming connections.

Apple outlines the basic functions of ALF in a Knowledge Base article ”Mac OS X v10.5, 10.6: About the Application Firewall” (Apple, 2010a). The ALF is turned off by default in OS X but once enabled in the Security & Privacy preference pane the major configuration is primarily a two-position switch which has changed only slightly since its debut in 10.5 Leopard and the stealth option. The stealth option tries to disguise the existence of the firewall in the same manner as similar features in other firewall applications by not responding to network probes such as ping.

In 10.5 the ALF default setting was to allow all incoming connections. This was a good idea for the first release of ALF and for automated upgrades from 10.4. Since then, once the firewall is enabled, the default has been the second position, which allows connections to signed applications using the codesigning features of OS X, as well as to those applications manually enabled or disabled. Since in current releases the operating system and built in programs are all signed by Apple and all App Store apps are signed by their developers this covers quite a lot of software on modern systems. The authors of *Enterprise Mac Security* advise caution about this feature in their explanation of ALF (Edge, Jr. et al., 2009) noting that on current OS X systems the set of all software signed by a valid Developer ID is quite large and could include unwanted applications and allow them inbound network access.

In the Advanced Firewall preferences pane (in Security & Privacy) the ALF settings are clear and easily changed with administrator privileges. Additionally, the

Firewall and Sharing preferences are integrated. Enabling a service in Sharing such as Remote Login (ssh) will automatically add it to the list of allowed applications in ALF so that it can receive connections. Administrators can also add signed applications to the list and explicitly allow or deny them incoming connections as illustrated in this screen capture of OS X 10.8 Mountain Lion.



Advanced Firewall – Security and Privacy - System Preferences – Mountain Lion

The alf utilities and configuration are in */usr/libexec/ApplicationFirewall*. An XML format plist file there, *com.apple.alf.plist*, includes the detailed ALF configuration and contains a comprehensive list of the Apple services ALF recognizes as well as the other settings. The plist is versioned but has changed little between Lion (1.0a23) and Mavericks (1.0a25), which only added a logging option. The firewall program itself, *Firewall,* and its logging daemon *appfwloggerd* are there as well. To interact with ALF use the *socketfilterfw* command in the same folder. All of the settings available in the preference pane are easily accessed with this utility. It can be executed with the common

"–h" option for the detailed usage statement as an unprivileged user but all the commands require an administrator account so the utility can connect to the ALF service. A few examples below demonstrate the command line equivalents of the configuration in the screenshot above.

```
$ sudo ./socketfilterfw --getglobalstate
Firewall is disabled. (State = 0)
$ sudo ./socketfilterfw --getallowsigned
Automatically allow signed software ENABLED
$ sudo ./socketfilterfw --getloggingmode
Log mode is on
$ sudo ./socketfilterfw --getstealthmode
Stealth mode disabled
$ sudo ./socketfilterfw --listapps
$ sudo ./socketfilterfw --setglobalstate on
Firewall is enabled. (State = 1)
$ sudo ./socketfilterfw --getglobalstate
Firewall is enabled. (State = 1)
$ sudo ./socketfilterfw --listapps
$ sudo ./socketfilterfw --getglobalstate
Firewall is enabled. (State = 1)
$ sudo ./socketfilterfw -listapps
## and then add Transmission in the prefpane…
$ sudo ./socketfilterfw --getblockall
Block all DISABLED!
$ sudo ./socketfilterfw --listapps
ALF: total number of apps = 1
1 :  /Applications/Transmission.app
     ( Allow incoming connections )
```

socketfilterfw usage examples - Mountain Lion

2.1.7 Adaptive Firewall

The Adaptive Firewall (AF) is included with Server, though not configured in Server.app's panels. AF adds and removes system addresses to a blacklist based on observed behavior. A whitelist feature allows systems that should not be blocked to be exempted. The utilities and firewall configuration for AF are nested deep in the *Server.app* bundle. *afctl* is in the *AdaptiveFirewall.bundle* folder and linked into the */usr/libexec* directory of *ServerRoot* for ease of execution.

Ben S. Knowles, adric@adric.net

As explained by author Charles Edge, Jr. in the post "Manage The Adaptive Firewall in Mavericks Server" on his Krypted web log (2013) the default configuration provided in */etc/af.plist* instructs AF to uses text files in */var/db/af* for the blacklists and whitelists. The plist file also contains the default thresholds, timers, and logging behaviour of AF (Edge, Jr., 2013).

The installation of Server adds an anchor for AF into the system pf configuration. It loads the AF configuration from inside the server bundle and is succinct in explaining how AF works.

```
#
# anchor ruleset for the Adaptive Firewall
# anchor name: 400.AdaptiveFirewall
# see afctl(8), pfctl(8), pf.conf(5)
#

table <blockedHosts> persist file "/var/db/af/blockedHosts"
block in quick from <blockedHosts> to any
```

/Applications/Server.app/Contents/ServerRoot/private/etc/pf.anchors/400.AdaptiveFirewall – Mountain Lion, Server 2

2.2 Credentials

OS X provides both unique implementations and standard tools to secure the access credentials of users, applications, and systems. Local and cached credentials including passwords, certificates, keys, and notes are protected by the keychain system. Client systems can integrate with directory services and Kerberos systems and OS X Server systems can host directories and integrate with existing LDAP and Active Directory authentication domains.

2.1.8 Keychains

The Apple Keychain Services framework provides protected storage and managed access to all manner of credentials for OS X users and services (as well as for iOS applications). The Keychain services provide a native API and also implement the Common Security Services Manager (CSSM) interface from The Open Group's Common Data Security Architecture (CDSA) (Apple, 2012a). These services and interfaces

provide sophisticated capabilities for many authentication and authorization use cases and can be used by the graphical Keychain Access application, the *security* utility, or by developing to those APIs. Extensive details about the services, APIs, and sample code are in Apple's *Keychain Services Programming Guide* (Apple, 2012a).

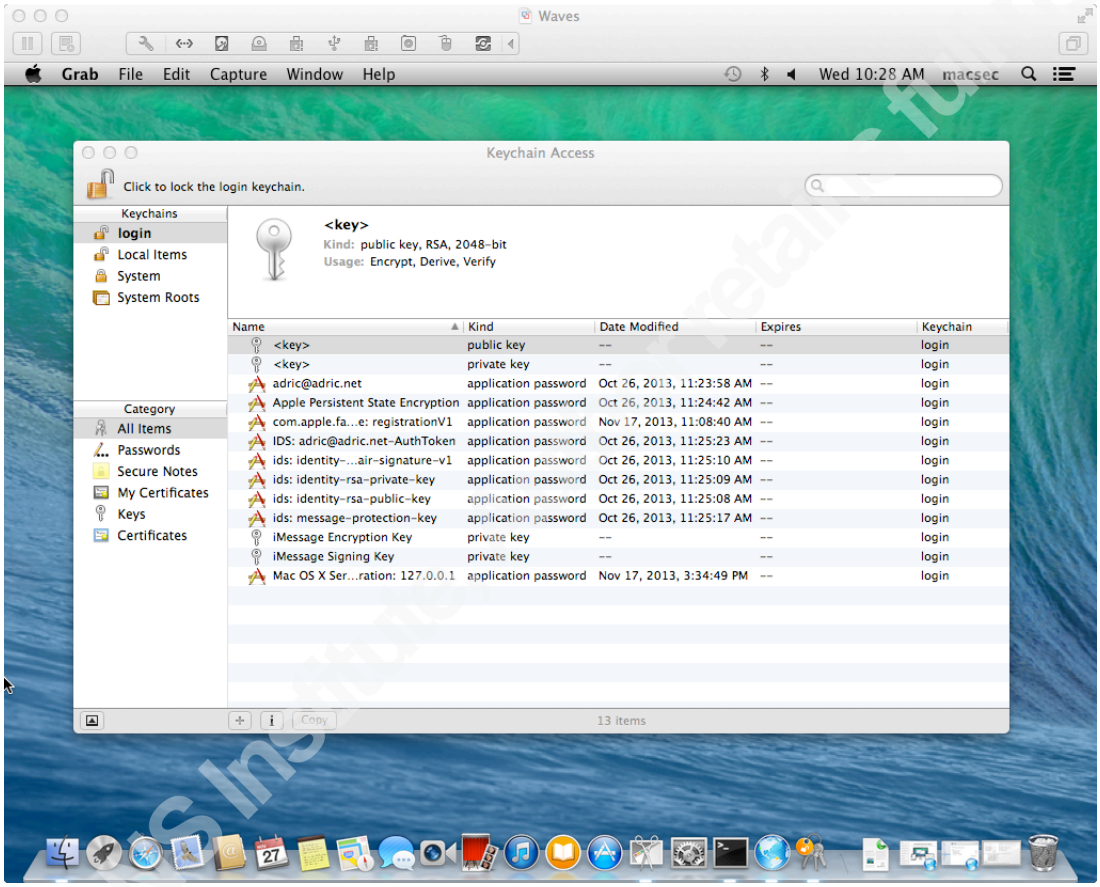
OS X systems have a minimum of three keychains: one for the system, one for the system provided root certificates, and one for each user. Users and applications can create and use additional keychains as needed. Each can contain username and password entries, certificates, keys, and secure notes. Keychains and their member items have access control lists managed by the framework APIs and utilities. In practice, by default, the system keychain is locked unless unlocked by an administrator and re-locks on a timer (default of 300 seconds), and user keychains remain unlocked during a login session.

Applications that want access to a user's keychain items have to explicitly request that access and it can be granted once (temporarily) or always (permanently). OS X users are probably used to these prompts (as might Windows users recognize UAC dialogs). They differ slightly by the name of the application, the items requested, and whether the user's keychain is already unlocked. The requesting application is hidden under the Details arrow.

Figure 1-4 from "Keychain Services Programming Guide: Keychain Services Concepts"

In fact, OS X users might not interact directly with their keychains much. Many keychain using applications use the services transparently. Programs including mail clients, chat programs, and web browsers store saved login credentials in keychains seamlessly. If users do make changes to keychains, such as for certificate management,

they use the graphical Keychain Access utility. Keychain Access provides options for managing keychains and items plus some utilities for certificate management and keychain maintenance.



Keychain Access – Mavericks

The *security* utility exposes all of the same functionality to command line and scripting applications. The more than forty subcommands are described in the extensive manual page. These examples show retrieving information about keychains. Others permit editing with appropriate privileges.

```
$ security list-keychains
"/Users/macsec/Library/Keychains/login.keychain"
"/Library/Keychains/System.keychain"
$ security show-keychain-info "/Library/Keychains/System.keychain"
```

Ben S. Knowles, adric@adric.net

```
Keychain "/Library/Keychains/System.keychain" lock-on-sleep timeout=300s
```

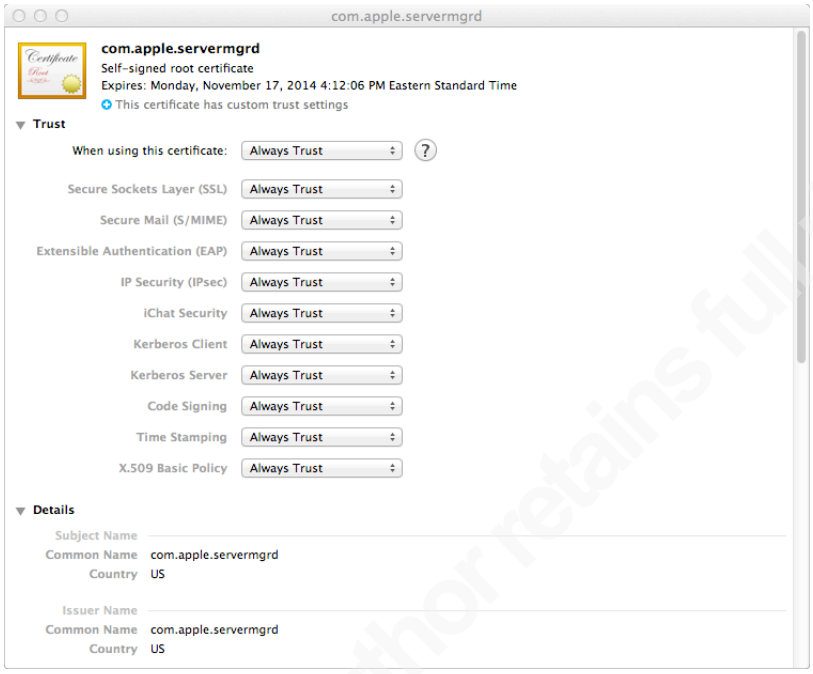
security(1) sample commands and output – Mavericks

2.1.9 System Keychain

The System keychain unique on each Mac OS X system is handled differently than user keychains. Even if no administrator login is active, system applications can unlock and access system keychain entries. OS X uses some utilities and code specifically for this, notably the *systemkeychain* utility. *systemkeychain* can create, lock, and unlock system keychains. A discussion on *Ask Different* (2012) of System keychain security explains the mechanism and references the manual page and released source of the utility. As *Ask Different* user Anon explains, "The libsecurity keychain framework allows regular processes to interact with the system keychain in an authenticated manner using Apple's XPC interprocess communication framework (IPC)." (Anon & Old Pro, 2012). The contributors there also note that it is trivially easy to restore the keychain and key file from an unencrypted backup to another system and then use the *systemkeychain* utility to unlock, recover, and use those credentials (Anon & Old Pro, 2012). This is a useful tip for system administrators as well as forensics examiners and is also certainly an important concern for backup security on OS X.

2.1.10 Certificates

The Keychain services, APIs, and utilities all support standard X.509 certificates natively. A brace of graphical utilities and wizards for certificate and certificate authority (CA) creation, validation, and management are included in Keychain Access (under Certificate Assistant in the Keychain Access root menu). Certificates are first class items in a keychain and can be viewed, edited, and manipulated easily. Complex ACLs and trust schemes can all be configured in the GUI.



servermgrd – Keychain Access – Mountain Lion

The *certtool* utility has all of the power and flexibility of the underlying frameworks hinted at in the GUI tools and explained in the developer manual. It also offers a scriptable alternative to the standard OpenSSL commands also available. *certtool* combined with *security* implement all of the frontend capabilities of the Keychain framework. The backend for all of this is handled primarily by *securityd* which, being a system service, is managed by launchd and *launchctl*.

2.1.11 Kerberos

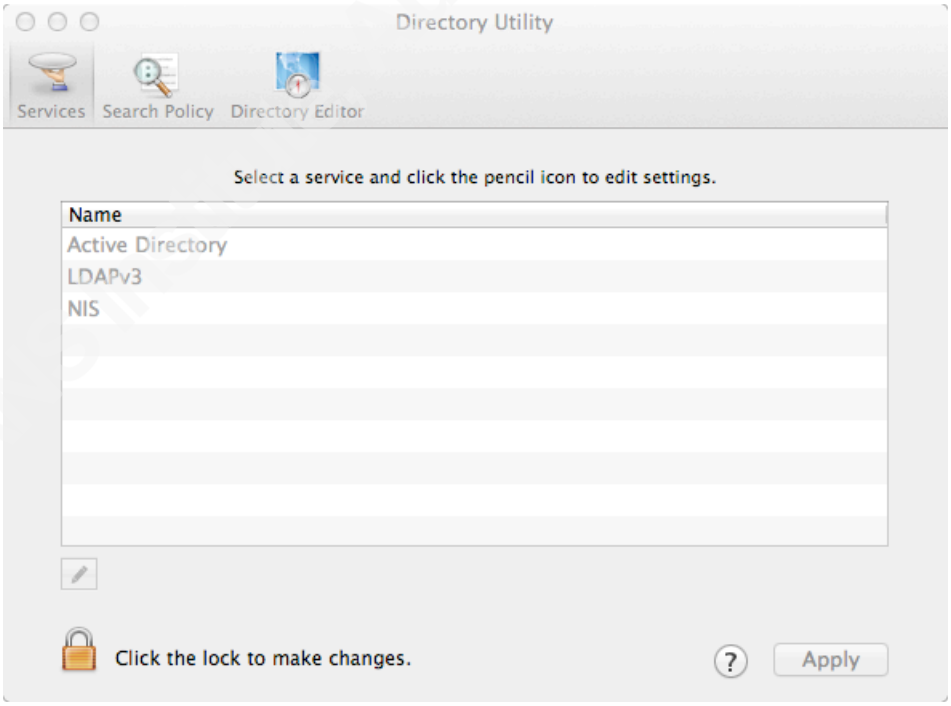
OS X includes support for the MIT developed shared authentication system Kerberos throughout the system. A Kerberos ticket utility is built into Keychain Access. Ticket Viewer is in the root Keychain Access menu. From there users can login to Kerberos systems and manage tickets and logins. Standard Kerberos command line utilities are also included including *klist* with similar functions to the Ticket Viewer application.

OS X Server uses Kerberos for authentication and authorization in the native

Open Directory system. The Server utility performs all of the setup and administration of the Kerberos configuration, master identities, database, and services in the background as part of the Open Directory initial configuration. Once completed the configuration files and keytab are visible on the Open Directory master server in */etc/krb5/* and expected variable data in */private/var/db/krb5kdc*. Many network services in OS X and Server are Kerberos enabled out of the box including SSH.

2.1.12 Directory Integration

On client systems another pair of graphical utility and command line utility applications handle configuring and managing directory services. Using Directory Utility (via the Users and Groups System Preference pane's Login options) or the *dsconfigad* and *dsconfigldap* utilities allows individual OS X system to be joined to an Active Directory, LDAP, or NIS directory.



Directory Utility – Mountain Lion

OS X Open Directory servers can connect to and federate with other directory systems, most notable Microsoft Active Directory (AD). AD and Open Directory use two

of the same core technologies: LDAP and Kerberos. Although it is tracking a moving target, Open Directory integration with Active Directory continues to improve and is generally suitable as the directory infrastructure for small workgroups. A common configuration is to use an OS X Server Open Directory master as a bridge between Macintosh computers and the Windows Active Directory environment. The OS X server can then manage the OS X clients using native tools to supplement the Group Policy Object (GPO) functionality of AD which does not work natively on OS X. Third party utilities provide more bridge features and integration into more complex environments enabling more complete management of OS X systems in an enterprise environment with stringent security requirements.

2.3 Software Installs and Updates

OS X includes frameworks, utilities, and supporting infrastructure for software installs in the native pkg format. Traditionally, *Installer.app* was used by the majority of applications that require installation. Simpler application bundles could just be dragged from the install media onto the drive. Background processes in OS X index all application bundles and so they need not be located in a Application folder. Installer.app as well as the OS X system install applications have a log window option in their menus that can be helpful in troubleshooting. Installed packages had their receipts saved to */Library/Receipts/* and you may find bill of materials and package files (*bom* and *pkg*) in that folder on older OS X computers or in restored backups. Complex applications still use Installer.app but many applications have migrated to the Apple Application Store (the App Store) and its distribution model, which replaces both the download and install processes.

2.1.13 Client apps

The separate Software Update functionality in OS X was combined with the Application Store when the App Store was launched. The App Store was installed on 10.6 Snow Leopard systems by the software update to 10.6.6 and was included in 10.7 Lion as reported by PC Magazine (Muchmore, 2011). The Software Update entry in the

Apple Menu now opens the App Store and switches to the Updates panel. Apple software including OS X, Server, and Xcode are now updated here as are any App Store installed applications. The *softwareupdate* utility is still present in 10.8 Mountain Lion but finds no new updates.

2.1.14 Server apps

OS X Server has long had the ability to manage patches for the workgroup systems it administers with Software Update Server. This includes downloading update packages from Apple, configuring their availability to managed clients, and pushing updates on a schedule. In Server 2, Caching Server is added and then extended in Server 3 (which requires 10.9 Mavericks) to caching other Apple web content including App Store apps and iBooks.

2.4 Anti-malware

2.1.15 File Quarantine

Apple and cooperating third-party applications record metadata on downloads into extended attributes in the filesystem. This is a bit like but more complex than the network zone ID recorded by cooperating browsers on modern Windows systems using NTFS alternate data streams (ADS). Mac OS X Internet clients record a subset of the available information about the download in the extended attributes (EA) of the download and carry those flags as the file is copied or unarchived. In the examples below Google Chrome captured and recorded the URL, a timestamp, and other details; Firefox and Transmission are identified as having downloaded the file with a timestamp; and *wget* does not record anything.

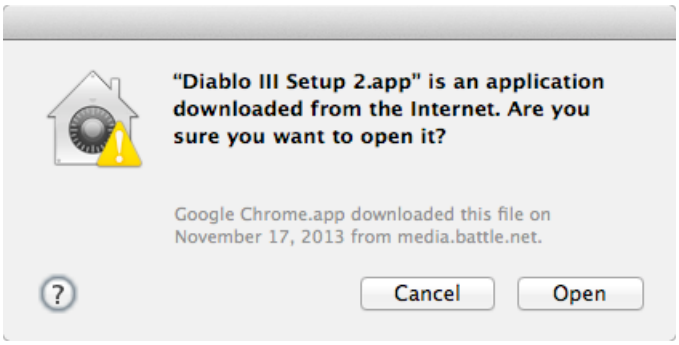
```
$ xattr -l Diablo-III-Setup-enUS.zip
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 5F 10 6E 68 74  |bplist00..._.nht|
00000010  74 70 3A 2F 2F 6D 65 64 69 61 2E 62 61 74 74 6C  |tp://media.battl|
00000020  65 2E 6E 65 74 2F 64 6F 77 6E 6C 6F 61 64 73 2F  |e.net/downloads/|
00000030  64 33 2D 69 6E 73 74 61 6C 6C 65 72 73 2F 30 64  |d3-installers/0d|
00000040  30 66 39 32 36 38 2D 36 65 39 32 2D 34 61 61 31  |0f9268-6e92-4aa1|
```

Ben S. Knowles, adric@adric.net

```
00000050  2D 38 35 35 36 2D 38 37 32 34 65 37 39 66 34 38  |-8556-8724e79f48|
00000060  33 33 2F 44 69 61 62 6C 6F 2D 49 49 49 2D 53 65  |33/Diablo-III-Se|
00000070  74 75 70 2D 65 6E 55 53 2E 7A 69 70 50 08 0B 7C  |tup-enUS.zipP..||
00000080  00 00 00 00 00 00 01 01 00 00 00 00 00 00 00 03  |................|
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7D  |...............}|
000000a0
com.apple.quarantine:
0001;528913a4;Google Chrome.app;A22D6EB1-5B3E-4BA0-8F97-2B4B3AC047A6
$ xattr -l -p com.apple.quarantine GIAC_Gold_Template.doc
com.apple.quarantine: 0000;52f6e293;Firefox.app;
$ xattr -l onapalehorse_audiobook_mp3_1389734129.zip
com.apple.quarantine: 0001;52e14737;Transmission.app;AD788C9B-A752-421B-A41A-
45C582207883
$ xattr -l substitute-courier-new-font.reg
```

File Quarantine EAs on recent Chrome, Firefox, Transmission, and wget downloaded files

Apple file quarantine EAs are recorded as a semi-colon delimited list of fields, unlike most EAs, which are property lists (binary or XML). The client application's name and timestamp are used by OS X to notify the user on opening of the downloaded file. If the information is complete this dialog can be quite informative.



Diablo installer File Quarantine notification – Mountain Lion

Once the program has been opened the flag that creates this warning is updated in the EA and the program will open without warning in the future.

Ben S. Knowles, adric@adric.net

```
$ xattr -l -p com.apple.quarantine Diablo\ III\ Setup\ 2.app/
com.apple.quarantine: 0001;528913a4;Google\x20Chrome.app;A22D6EB1-5B3E-4BA0-8F97-
2B4B3AC047A6
# application opened, screengrab taken, and then:
$ xattr -l -p com.apple.quarantine Diablo\ III\ Setup\ 2.app/
com.apple.quarantine: 0061;528913a4;Google\x20Chrome.app;A22D6EB1-5B3E-4BA0-8F97-
2B4B3AC047A6
```

File Quarantine EAs before and after program open dialog – Mountain Lion

The quarantine system can help defeat simple drive-by download attacks by warning of a program's source when executing it. Not all applications that may download potentially dangerous programs implement this system. It is important to understand the selective coverage of File Quarantine when examining how it integrates with XProtect and Gatekeeper.

2.1.16 XProtect

Apple (2013c) explains in their knowledge base entry "OS X: About the "Are you sure you want to open it?" alert (File Quarantine / Known Malware Detection)":

> OS X improves download validation by providing file quarantine in applications that download files from the Internet. This means that downloads are checked for safety (known malware) when you try to open them. (2013c)

OS X has included an anti-malware scanner and signature rule database since 10.6 Snow Leopard. Apple has published few details about the system or its capabilities. Researchers dubbed the feature "XProtect" based on the name of the configuration file it keeps signature rules in, *XProtect.plist*. Unlike most anti-malware software, XProtect covers only current OS X affecting malware (not, e.g., Windows malware) and Apple updates the signatures as needed rather than on a published schedule.

In the past, outbreaks of Trojan horse malware for OS X systems and updates to Adobe Flash have been some of the events that preceded XProtect updates. Many OS X updates also include updates to the XProtect rules. Determined system administrators have scripted additional rules and rule management onto XProtect, but without any

support from Apple. Apple deployed an automatic updating capability in Security Update 2011-003 for XProtect toggled by a System Preference (Apple, 2011).

When a File Quarantine check is triggered the XProtect list is consulted and any matching applications are given a different dialog with a more frightening "will damage your computer" message advising the user to delete the application and eject any implicated disk image (Apple, 2013c).

The combination of File Quarantine and XProtect are extremely effective at blocking execution of known malware on OS X systems.

2.1.17 Internet Clients

The default OS X web browser Safari includes many features to protect users from malicious content. Since 10.6 Snow Leopard, Safari many plugins including Adobe Flash, Microsoft Silverlight, Oracle Java, and Apple's own Quicktime have been sandboxed into separate processes using XPC. In 10.7 Lion Safari, Apple migrated the browser to WebKit2 and tabs are now isolated from each other much as in Google Chrome as explained by *Ars Technica* reviewer John Sircusa in his review of 10.7 Lion (Siracusa, 2011).

The most recent versions of Safari have increased the flexibility of plugin blocking, allowing users to configure which plugins to permit for selected sites as promoted on Apple's website for Safari (Apple, 2014c). The document viewer Preview and the Quick Look feature uses the same techniques to isolate the dangerous work of file parsing from the less complex and less vulnerable display engine (Edge, Jr, et al., 2010).

2.1.18 Plugins

In 2013, Apple started using the XProtect mechanisms to block known vulnerable versions of Internet plugins, specifically Adobe Flash and Oracle Java, known to be vulnerable to exploits in wide usage on the Internet (Tung, 2013). The minimum required version of the plugins are recorded in *XProtect.meta.plist* and checked by participating applications such as the Safari and Chrome browsers. The Mozilla Project (2014) uses its own plugin blocking for add-ons to Firefox and Thunderbird as explained in a support

article. These blocks cause some headaches for system administrators and users of applications that need vulnerable versions of software, but are effective at slowing the spread of drive-by malware and worms. Apple has briefly set the XProtect required minimal version of the Java plugin software to a version number that has not been released. OS X systems that download that update have the Java plugin blocked entirely until the new version is released as reported by ZDNet (Tung, 2013).

2.5 Exploitation Resistance

2.1.19 Internals

Apple has been steadily increasing the coverage of location randomization to make exploitation of OS X more difficult since initially adding it in 10.5 Leopard. In Mountain Lion and Mavericks "the kernel, kexts, and system frameworks" are all protected as noted in their respective b whitepapers (Apple, 2012b; Apple, 2013b).

Unfortunately, as *The Mac Hacker's Handbook* (2009) explains, OS X library locations are not randomized and are in fact listed in world-readable configuration files. Even if the library locations are different between different installs of the operating systems (such as on different machines) the configuration files ease custom compiling an exploit and, as the authors note, the stack location is not randomized (Miller & Dai Zovi, 2009).

2.1.20 Code signing

For several years Apple has offered a Developer ID in the form of cryptographic certificates to paid members of its OS X and iOS developer programs. With the iOS App Store, developer certificates are a requirement for app submission and with the debut of the App Store on OS X in 10.7 Lion and then Gatekeeper in OS X 10.8 Mountain Lion, OS X developers are strongly encouraged to sign their code with Apple's system. Apps submitted to the OS X app store are required to be signed and Gatekeeper equipped systems will not run any unsigned applications in their default configuration. Certificates expire and can be revoked by Apple if policies are violated as detailed in the Developer Support Center technical support article on certificates on Apple's developer support

website (Apple, 2014b).

Enterprise Mac OS X Security: Mac OS X Snow Leopard (2009) begins Part II with an extensive discussion of the implementation and applications of OS X code signing for ensuring applications' source and integrity before explaining its integration with Keychain Access, the application firewall, and client management features. With this foundation established, the authors explain the code signing process and the use of *codesign* and related utilities (Edge, Jr, et al., 2009).

Some quick checks of Apple and third party software with *codesign* are illustrative of the scope and capabilities of the codesigning system. A set of data is collected about each program and the hash of that data is signed by a key that must be found valid (via certificate chains) at time of check to pass. Apple's system *perl* binary is signed, but MacPort's is not. Blizzard Entertainment's official desktop update agent is signed but the troubleshooting utility LogGoblin is unsigned.

```
        lorelei:~ adric$ codesign -dvvvv /usr/bin/perl
        Executable=/usr/bin/perl
        Identifier=com.apple.a2p
        Format=Mach-O universal (i386 x86_64)
        CodeDirectory v=20100 size=222 flags=0x0(none) hashes=6+2
location=embedded
        Hash type=sha1 size=20
        CDHash=ec9902998bc7c2ee2af1ab114e5d099819263db8
        Signature size=4064
        Authority=Software Signing
        Authority=Apple Code Signing Certification Authority
        Authority=Apple Root CA
        Info.plist=not bound
        Sealed Resources=none
        Internal requirements count=2 size=156

        $ codesign -dvvvv /opt/local/bin/perl
        /opt/local/bin/perl: code object is not signed at all

        $ codesign -dvvvv
/Users/Shared/Battle.net/Agent/Agent.2689/Agent.app/Contents/MacOS/Agent
        Executable=/Users/Shared/Battle.net/Agent/Agent.2689/Agent.app/Contents/Ma
cOS/Agent
```

```
        Identifier=com.blizzard.agent
        Format=bundle with Mach-O thin (i386)
        CodeDirectory v=20100 size=55627 flags=0x0(none) hashes=2775+3
location=embedded
        Hash type=sha1 size=20
        CDHash=915ffaa759e5b20c51899fcdcac3a1e5c638aedd
        Signature size=8541
        Authority=Developer ID Application: Blizzard Entertainment, Inc.
        Authority=Developer ID Certification Authority
        Authority=Apple Root CA
        Timestamp=Feb 12, 2014 9:44:06 PM
        Info.plist entries=25
        Sealed Resources rules=4 files=2
        Internal requirements count=1 size=180

        $ codesign -dvvvv /usr/bin/perl
        Executable=/usr/bin/perl
        Identifier=com.apple.a2p
        Format=Mach-O universal (i386 x86_64)
        CodeDirectory v=20100 size=222 flags=0x0(none) hashes=6+2
location=embedded
        Hash type=sha1 size=20
        CDHash=ec9902998bc7c2ee2af1ab114e5d099819263db8
        Signature size=4064
        Authority=Software Signing
        Authority=Apple Code Signing Certification Authority
        Authority=Apple Root CA
        Info.plist=not bound
        Sealed Resources=none
        Internal requirements count=2 size=156

        $ codesign -dvvvv ~/Downloads/LogGoblin
        /Users/adric/Downloads/LogGoblin: code object is not signed at all
```

codesign output for four files – Mountain Lion

A few anecdotes from iOS security illustrate well the controversy over the effectiveness of requiring code signing. As reported by Sophos on their Naked Security web log in 2011, famed OS X attacker Charlie Miller had his Developer ID revoked by Apple after he demonstrated vulnerabilities in the app vetting process (Ducklin, 2011). ComputerWorld magazine reported in 2013 of a Georgia Tech research team whose

"Jekyll" app reconfigured itself to perform malicious activity only after approved and installed on unsuspecting users' phones from the App Store (Keizer, 2013). A Macworld online article summarizes recent reports of OS X malware that was signed with valid Developer IDs and was installed onto systems before Apple revoked those certificates (Macworld, 2013).

2.1.21 Sandboxing

OS X sandboxing runs applications in a restricted environment enforced by mandatory access controls (MAC) in the OS X kernel which came from the TrustedBSD project (Edge, Jr, et al., 2009). The sandbox system defines fine-grained privileges which participating applications must request specifically or be granted by an authorized user. In this way, the effects of both unintentional flaws and malicious exploitation are confined to the scoped resources unless the sandbox and MAC mechanisms themselves have flaws. In combination with privilege separation through XPC, this provides powerful defenses of software integrity and user data privacy.

The privileges available to Application Sandbox applications are defined by Apple as entitlements and are typically Boolean values (true or false). Application entitlements must be included in the Xcode project for an application if Application Sandboxing is enabled and there is much detail and sample code about how to design sandboxed applications or port existing applications to the sandbox in Apple's developer documentation (Apple, 2014a). Some OS X applications make use of sandbox and XPC in 10.8 Mountain Lion and later release. Also, all programs submitted to Apple for the OS X App Store must use sandboxing, though some temporary entitlements have been made available for some use cases.

It is important to note that sandbox entitlements are part of the application metadata monitored by codesigning so changes to an installed application's collection entitlement requests would be flagged and blocked. Apple provides APIs for running sandbox apps to request additional permissions from the user. Sandboxed apps are contained within a restricted filesystem and these containers are in the user's Library folder organized by bundle name. Here is part of output of the *asctl diagnose* utility

describing the container of Amazon kindle application for OS X.

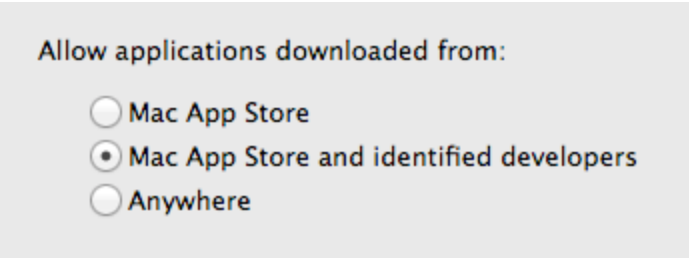
```
Container: com.amazon.Kindle [drwx------  4 adric  staff  - 136 Feb 13  2013
        /Users/adric/Library/Containers/com.amazon.Kindle]
ACLs:
1 code requirement acceptable for container
/Users/adric/Library/Containers/com.amazon.Kindle:
        (anchor apple generic and certificate leaf[field.1.2.840.113635.100.6.1.9]
/* exists */ or anchor apple generic and certificate
1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate
leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate
leaf[subject.OU] = "94KV3E626L") and identifier "com.amazon.Kindle"
```

containers.txt excerpt – Kindle.app - Mountain Lion

Although the internal mechanism of the Sandbox kernel driver is proprietary, the core Sandbox profiles themselves are available on OS X systems in */usr/share/sandbox.*

These profiles are compiled into the kernel as well as into applications, but the files themselves are readable. They are written in a Scheme derived language and with a little study can be used as examples to implement custom profiles as explained briefly in *The Mac Hacker's Handbook* (Miller & Dai Zovi, 2009) and extensively in *Enterprise Mac OS X Security: Mac OS X Snow Leopard* (Edge, Jr., et al., 2010). In this way it is easy enough to experiment with application sandboxing without using Xcode tools by writing profiles and trying them out on programs with *sandbox-exec*. Some experimentation is likely necessary to any study of the OS X Application Sandbox as it has advanced in effectiveness and complexity since those works were published and Apple has not published much about the system's internal workings.

2.1.22 Gatekeeper



Ben S. Knowles, adric@adric.net

General - Security & Privacy – System Preferences – Mountain Lion

Gatekeeper integrates the cryptography of the code signing system with system wide software management and the App Store policies. All App Store application submissions are required to be signed by the developers. More of the OS X system and included software files have been signed by Apple in each release. With Gatekeeper, unsigned application code is not normally executable by users and, in the shipping default setting, only App Store applications can be run by users. Applications can still be launched manually using context menus or manual changes to EAs.

As explained in a Trend Micro web log post about 10.9 Mavericks, the integration between File Quarantine and Gatekeeper exposes a simple design flaw (Lin, 2013). By making the file execution exception part of the file's metadata, an exempted file can be exchanged with other users or even packaged by attackers to execute on other OS X systems, avoiding Gatekeeper entirely.

2.6 Conclusion

OS X has many unique security systems mirroring the system's overall complex composition of standard open source UNIX technology, NeXTSTEP ideas, and classic Macintosh feel. Much as with the OS X environment as a whole, this collection of security frameworks, services, utilities and configuration grants flexibility not found in other systems, though, at cost of increased complexity and uniqueness. With some explicit understanding of the layers of security available in modern OS X systems and a handle on its quirks, information security professionals can effectively protect the confidentially, integrity and availability of data and systems under their care.

Ben S. Knowles, adric@adric.net

3. References

Anon, Old Pro. (2012). How is the System Keychain secured in OS X? [Online answer]. Retrieved from http://apple.stackexchange.com/questions/53579/how-is-the-system-keychain-secured-in-os-x

Apple, Inc. (2010a). *Mac OS X v10.5, 10.6: About the Application Firewall*. Retrieved from http://support.apple.com/kb/ht1810

Apple, Inc. (2010b). *Technology Brief: Mac OS X Security (Snow Leopard)*. Retrieved 2010 from http://images.apple.com/macosx/security/docs/MacOSX_Security_TB.pdf

Apple, Inc. (2011). *About Security Update 2011-3*. Retrieved from http://support.apple.com/kb/HT4657

Apple, Inc. (2012a). *Keychain Services Programming Guide: Keychain Services Concepts*. Retrieved from https://developer.apple.com/library/ios/documentation/Security/Conceptual/keychainServConcepts/02concepts/concepts.html

Apple, Inc. (2012b). *OS X Mountain Lion Core Technology Overview*. Retrieved from http://movies.apple.com/media/us/osx/2012/docs/OSX_MountainLion_Core_Technologies_Overview.pdf

Apple, Inc. (2013a). *Best Practices for Integrating OS X With Active Directory*. Retrieved from http://training.apple.com/pdf/wp_integrating_active_directory_ml.pdf

Apple, Inc. (2013b). *OS X Mavericks Core Technology Overview*. Retrieved from https://www.apple.com/media/us/osx/2013/docs/OSX_Mavericks_Core_Technology_Overview.pdf

Apple, Inc. (2013c). *OS X: About the "Are you sure you want to open it?" alert (File Quarantine / Known Malware Detection)*. Retrieved from http://support.apple.com/kb/HT3662

Apple, Inc. (2014a). *Apple Developer – App Sandboxing.* Retrieved from

Ben S. Knowles, adric@adric.net

https://developer.apple.com/app-sandboxing/

Apple, Inc. (2014b). *Apple Safari: Browse the web in smarter more powerful ways*. Retrieved from https://www.apple.com/safari/

Apple, Inc. (2014c). *Developer Technical Support: Certificates.* Retrieved from https://developer.apple.com/support/technical/certificates/

Apple, Inc. (2014d). *Open source - Releases*. Retrieved from http://www.opensource.apple.com/

Apple, Inc. (2014e). *OS X Mavericks – See everything the new OS X can do*. Retrieved from http://www.apple.com/osx/whats-new/features.html#security

Apple, Inc. (2014f). *Apple – Remote Desktop 3*. Retrieved from http://www.apple.com/remotedesktop/.

DTrace Community. (2014). Dtrace.org: About DTrace [Web log post]. Retrieved from http://dtrace.org/blogs/about/

Edge, Jr., C., Barker, W., Hunter, B., & Sullivan, G. (2010). *Enterprise Mac Security: Mac OS X Snow Leopard*. New York, NY: Apress.

Edge, Jr, C. (2012). A Cheat Sheet For Using pf in OS X Lion and Up [Web log post]. Retrieved from http://krypted.com/mac-security/a-cheat-sheet-for-using-pf-in-os-x-lion-and-up/

Edge, Jr, C. (2013). Manage The Adaptive Firewall in Mavericks Server [Web log post]. Retrieved from http://krypted.com/mac-security/manage-the-adaptive-firewall-in-mavericks-server/

Jepson, B., Rothman, E., & Rosen, R. (2008). *Mac OS X for Unix Geeks* (4th ed.). Sebastopol, CA: O'Reilly Media.

Keizer, G. (2013). Researchers outwit Apple, plant malware in the App Store. *ComputerWorld.* Retrieved from http://www.computerworld.com/s/article/9241742/Researchers_outwit_Apple_plant_malware_in_the_App_Store

Levin, J. (2013). *Mac OS X and iOS Internals: To the Apple's Core.* Indianapolis, Indiana: John Wiley & Sons, Inc.

Lin, L. (2013). Gatekeeper on Mac OS X 10.9 Mavericks [Web log post]. Retrieved

from http://blog.trendmicro.com/trendlabs-security-intelligence/gatekeeper-on-mac-os-x-10-9-mavericks/

Macworld. (2013). New digitally signed Mac malware confuses users with right-to-left file name tricks. *Macworld Online.* Retrieved from http://www.macworld.com/article/2044473/new-digitally-signed-mac-malware-confuses-users-with-right-to-left-file-name-tricks.html

Miller, C., & Dai Zovi, D. (2009). *The Mac Hacker's Handbook*. Indianapolis, Indiana: John Wiley & Sons, Inc.

Muchmore, M. (2011, January). Apple's Mac App Store: Hands On. *PC Magazine*. Retrieved from http://www.pcmag.com/article2/0,2817,2375320,00.asp

OpenBSD Project. (2009). *Upgrade Guide: 4.6 to 4.7: pf(4) NAT syntax change*. Retrieved from http://www.openbsd.org/faq/upgrade47.html#newPFnat

Singh, A. (2007). *Mac OS X Internals: A Systems Approach.* Upper Saddle River, NJ: Addison-Wesley.

Siracusa, J. (2011). Mac OS X 10.7 Lion: the Ars Technica review. *Ars Technica*. Retrieved from http://arstechnica.com/apple/2011/07/mac-os-x-10-7/

Stockley, M. (2011). Apple lets malware into App Store [Web log post]. Retrieved from http://nakedsecurity.sophos.com/2011/11/08/apples-app-store-security-compromised/

Mozilla Project. (2014). Add-ons that cause stability or security issues are put on a blocklist. Retrieved from https://support.mozilla.org/en-US/kb/add-ons-cause-issues-are-on-blocklist

The Open Group. (2014). The Open Brand - Register of Certified Products. Retrieved from http://www.opengroup.org/openbrand/register/

Tung, L. (2013). Apple's anti-malware blacklists Java 7 plug-in again. *ZDNet.* Retrieved from http://www.zdnet.com/apples-anti-malware-blacklists-java-7-plug-in-again-7000010686/

Wikipedia Community. (2014a). Property list. *Wikipedia.* Retrieved from http://en.wikipedia.org/wiki/Property_list

Wikipedia Community. (2014b). ZFS. *Wikipedia.* Retrieved from

http://en.wikipedia.org/wiki/ZFS#OS_X

XQuartz Project. (2014). XQuartz Trac. Retrieved from http://xquartz.macosforge.org/trac/wiki/WikiStart