DeepSight™ Threat Management System
Research Report

The Mac OS X Threat Landscape: An Overview

Version 1: July 25, 2006
Version 2: November 13, 2006

Analysts: Aaron Adams

Executive Summary

Historically, the Apple Macintosh OS X operating system has received little attention in the security industry. This trend is changing; in the last few years, research targeting OS X has increased significantly and media attention regarding its security has increased as well.

This document outlines the current OS X threat landscape as observed by the DeepSight Threat Analyst Team. It gives an overview of the advisories Apple has released and the trends researchers have made in their focus of the issues. The paper also discusses past, current, and future threats, including high-profile vulnerabilities, publicly available exploits, and rootkits.

This document does not disclose new threats or techniques for the OS X platform. It was written with the intention of laying out the realistic and current threat landscape that may affect OS X by analyzing what researchers around the world have already discovered. The document also tries to discuss future possibilities that attackers may focus on and the security mechanisms that can be put in place to effectively prevent exploitation attempts. Note that these speculations are based on current threats and research, rather than on evidence from actual cutting-edge attacks.

November 13, 2006 Update: This document has been updated with new information from the original release through to the end of October.

Table of Contents

Executive Summary 1
Overview 3
  The OS X operating system 3
  Security advisories 4
Current and past threats 4
  Vulnerabilities 4
    Local vulnerabilities 5
    Remote vulnerabilities 5
    Kernel vulnerabilities 7
    System design weaknesses 9
    OS X zero-day issues 11
  Malicious code 11
    MP3Concept 11
    OSX.Inqtana 12
    OSX.Leap.A 13
    SH.Renepo.B 13
    OSX.Macarena 13
  Exploit development and research 14
    OS X exploit development 14
    OS X exploit research 14
  Rootkits 19
    WeaponX 19
    OSXRK 20
    Togroot 20
Defenses 21
  Detecting malicious code 21
  Detecting rootkits 21
  Protecting against vulnerabilities and exploits 22
  Locking down services and firewall policies 22
OS X future threats and defenses 22
  Malicious code 22
  Rootkits 23
  Vulnerabilities and exploits 23
  OS X security technologies and preventive measures 24
    Stack canary values 24
    Secure heap implementation 24
    Address Space Layout Randomization 25
    Non-executable memory 25
  Future threats and malicious advancements facing OS X 26
Conclusion 27
Resources 27
Community Credits 28
Change Log 28
Glossary 29
Contact Information 29
About Symantec 29

Overview

The OS X operating system

Apple released OS X Server in 1999, soon after the initial release of Darwin. This was followed by a Public Beta release in 2000 and the first official desktop release of 10.0 (nick-named "Cheetah") in March of 2001. Afterward, there was the release of Puma (10.1), Jaguar (10.2), and Panther (10.3). At the time of writing, the latest release of OS X is Tiger (10.4), which was released in April of 2005. The release date of the upcoming OS X Leopard (10.5) has not yet been announced. **Figure 1** illustrates the timeline of these releases.

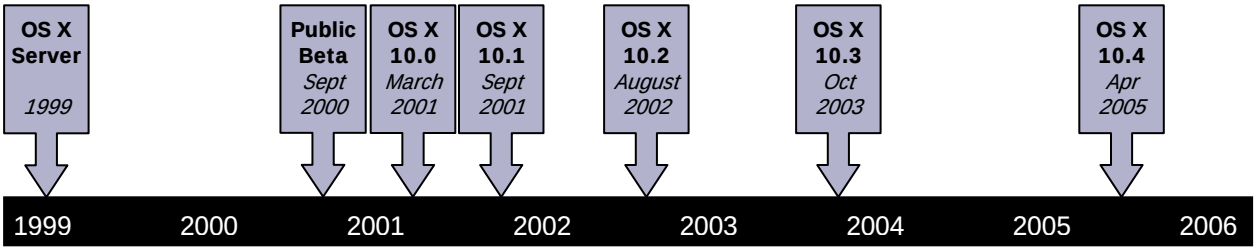

Figure 1: *Apple OS X release timeline*

The OS X operating system is based on FreeBSD, with a set of additional tools and frameworks (such as Core Foundation) built on top. The underlying kernel used by OS X is Darwin, a Mach-based kernel. Because Mac OS X is a UNIX-based operating system, it inherits all its built-in security features, such as a well-designed multiuser infrastructure as well as process and file attributes. It integrates functionality from BSD and Mach kernels, allowing both to interoperate independently.

Security advisories

Apple releases regular security updates for OS X, but does not follow any formal release schedule. At the time of this writing, Apple has released over 100 security-related updates since the initial release of OS X.

These security updates range from addressing a single vulnerability to more than 25 separate vulnerabilities. In the beginning, advisories typically focused on vulnerabilities in third-party applications that were included with OS X. However, as security researchers gained more interest in the operating system, vulnerabilities in Apple-specific libraries, utilities, and implementations began to be reported (**Figure 2**).

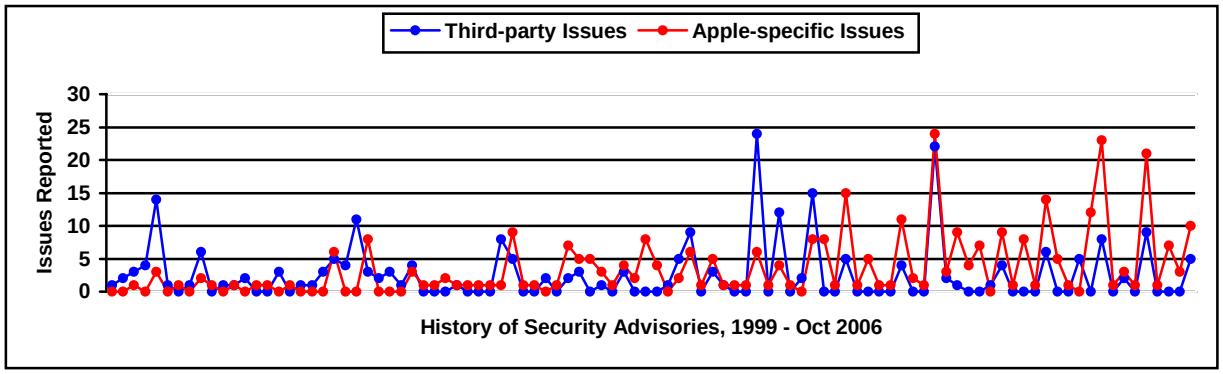

Figure 2: *Apple OS X security updates in third-party vs. Apple software; 1999 - Oct 2006*

This data was taken from the publicly available information on the Apple Security Update website[1]. The third-party issues (blue) represent how many vulnerabilities affecting non-Apple applications were addressed. The Apple-specific issues (red) represent how many vulnerabilities affecting Apple software were addressed.

The data clearly shows that over the course of time security updates have begun to address more Apple-specific issues than third-party applications, which in turn suggests an increase in focus on OS X specific software by security researchers.

Current and past threats

Vulnerabilities

Vulnerabilities discovered in Apple software have fallen into the typical range of categories including local privilege escalation, client-side code execution, and remote code execution. As with most operating systems, these issues range from low to critical risk and affect a variety of software including some more obscure applications to remotely accessible default services. Exploiting these vulnerabilities on OS X is not notably more or less difficult than doing so on most other platforms.

The following is a list of interesting or critical vulnerabilities discovered in OS X, by category. This list is by no means exhaustive, but provides a good overview of significant vulnerabilities affecting OS X to date.

[1] Apple Security Updates (http://docs.info.apple.com/article.html?artnum=61798)

Local vulnerabilities

Over the past few years quite a number of local vulnerabilities have been discovered in OS X; the impact ranges from local denial of service to root or kernel privilege elevation. The classifications of the vulnerabilities range from historic environment-variable hacks to the currently more common integer-handling bugs seen in most software. The following is a list of some of the more significant and interesting local vulnerabilities discovered for the platform, all of which have been patched by Apple.

Apple Mac OSX passwd Tool Arbitrary File Modification and Corruption
http://www.securityfocus.com/bid/16910

In 2006, iDefense disclosed a vulnerability allowing an attacker to force the `passwd` application to corrupt files with the privileges of the root user. An attacker could corrupt any file on the filesystem by supplying a specific argument to the application. An attacker with local user access can leverage this issue to gain root privileges. Security researcher Vade 79 wrote an exploit for this issue soon after its disclosure.

Apple Mac OSX CF_CHARSET_PATH Environment Variable Buffer Overflow
http://www.securityfocus.com/bid/13224

This is an example of classic local system vulnerabilities allowing trivial privilege elevation. In this case, the Apple Core Foundation library contains a buffer-overflow vulnerability when parsing the CF_CHARSET_PATH environment variable. An attacker could exploit this vulnerability in any setuid application linked to Core Foundation, allowing local users to trivially obtain root access. Vade 79 developed an exploit for this issue, which iDefense disclosed in 2005.

Apple malloc Insecure Environment Variable Use
http://www.securityfocus.com/bid/14939

Due to a vulnerability in the malloc function used by Mac OS X, an attacker could supply a debugging environment variable to force an arbitrary file on the system to be overwritten. Attackers could trivially exploit this by targeting any setuid application making use of the malloc function (which most do). This could be exploited to gain the privileges of the targeted setuid application, typically root.

Remote vulnerabilities

Remote vulnerabilities tend to be much more serious than local issues. These are what the security of a system is often gauged from, simply because remote attackers can exploit remote vulnerabilities without requiring authentication credentials or trust relationships with an affected target.

The following is a list of some of the more significant remote vulnerabilities affecting OS X in the past. Keep in mind that any significant vulnerabilities in third-party software, such as Samba or Apache, may also serve as remote exploitation vectors against OS X computers. For example, an exploit targeting the Samba 'call_trans2open' Remote Buffer Overflow Vulnerability (BID 7294) on OS X was released as part of the Metasploit[2] framework (discussed later in the OS X Remote Exploits section).

Apple Mac OS X Server Administration Service Undisclosed Remote Buffer Overflow Vulnerability
http://www.securityfocus.com/bid/9914

[2] The Metasploit Project (http://www.metasploit.com/)

This issue affects the administration service listening on TCP port 610. No specific technical details regarding this issue were ever released; however, a remote attacker could trigger a buffer overflow by delivering excessive data to the service. An attacker could possibly exploit this issue to execute an arbitrary payload.

Apple OS X Multiple Unspecified Large Input Vulnerabilities
http://www.securityfocus.com/bid/10268

Apple reported a vulnerability in the native OS X Apple File Server service as part of a cumulative update that was related to handling of long password strings. The details of this vulnerability were also never disclosed; however, an unauthenticated remote attacker could possibly have exploited this issue to execute arbitrary code.

Apple Mac OS X AppleFileServer Remote Buffer Overflow Vulnerability
http://www.securityfocus.com/bid/10271

This vulnerability received some of the most attention by security researchers and is one of the only significant remote vulnerabilities in a native Mac OS X service for which a publicly available exploit was released. The exploit for this issue is mentioned in the section "Exploit development and research" in this document. This vulnerability is a stack-based overflow when processing a pathname supplied in a LoginExt packet. These packets are used to perform authentication using a clear-text authentication method. The actual vulnerability can be triggered by supplying a string longer than the length reported in the packet. The issue likely occurs because the software copies the user-supplied string until a NULL byte is encountered, rather than using the supplied length value.

Apple Mac OS X AppleFileServer Remote Integer Overflow Vulnerability
http://www.securityfocus.com/bid/12478

An additional Apple File Server vulnerability allowed a bounds check to be bypassed by supplying an overly large signed integer as a length parameter. In the bounds-checking evaluation, this value would be interpreted as a negative value. Although this condition can result in the corruption of sensitive data and could likely be exploited to execute arbitrary code, a public exploit was never released. Nevertheless, given the nature of the bug, exploitation would be similar to the previously described issue and is thus assumed to be possible.

Apple Mac OS X BlueTooth Directory Traversal Vulnerability
http://www.securityfocus.com/bid/13491

The Apple Bluetooth protocol stack was found vulnerable to a directory-traversal vulnerability when accepting files pushed to a system via the OBEX transfer protocol. This could allow a malicious person to upload a file to an arbitrary location on a victim's hard drive, leading to remote access. This vulnerability was eventually deployed in a proof-of-concept worm dubbed Inqtana.A by the researcher KF. For more information regarding this worm, see the section "Malicious code" later in this document. This vulnerability affected both OS X 10.3 and the first release of OS X 10.4.

The above should show that OS X is in no way immune to or less likely to be affected by critical remote vulnerabilities than other operating systems. While this is by no means an exhaustive list and is limited to software specific to OS X, it is clear that vulnerabilities that could facilitate worm propagation have been discovered, and, in some cases, exploited. All of the vulnerabilities listed above have been patched by Apple.

Kernel vulnerabilities

Numerous vulnerabilities have been discovered in the kernel used by OS X. Like most other kernel vulnerabilities, these typically are found in system calls and also in the processing of file formats, such as the native Mach-O executable format. Here are a number of examples that have been patched by Apple.

OS X Kernel Mach-O File Loading Denial of Service Vulnerability
http://www.securityfocus.com/bid/13222

In 2005, Nemo, of the group Feline Menace, disclosed a kernel vulnerability that occurs when processing a Mach-O file. Due to a failure to properly verify the contents of the number of commands stored within a Mach-O file, it's possible to enter a loop for an excessive number of iterations, eventually causing a kernel panic. A local unprivileged user could trivially exploit this to crash an affected system.

OS X Kernel semop() System Call Signed Integer Stack Corruption Vulnerability
http://www.securityfocus.com/bid/13225

Immunity disclosed a vulnerability affecting the `semop()` BSD system call in OS X in 2005. This vulnerability could allow an attacker to corrupt sensitive values within kernel memory and execute arbitrary code. A local or a remote attacker via a sophisticated second-stage[3] payload could exploit this issue to elevate their privileges to root or introduce new code (such as a rootkit) into the kernel. This is a very standard type of vulnerability and would likely be quite trivial to exploit.

OS X Kernel mach_msg_send() Kernel Heap Corruption Vulnerability
http://www.securityfocus.com/bid/17056

A vulnerability was discovered in the `mach_msg_send()` function in the Mac OS X kernel. Due to a failure to check the size of user-supplied size variable, the researcher was able to cause an integer to wrap and force the kernel to allocate insufficient heap-based memory. The original size value was then used in a copy operation, allowing for the corruption of kernel data with user-supplied data. An attacker could potentially exploit this issue to execute arbitrary code within the context of the kernel and then install a rootkit, or take other malicious actions.

Other kernel vulnerabilities, including other classes such as information leaking, were described in a presentation at Black Hat Europe 2005:

http://www.blackhat.com/presentations/bh-europe-05/BH_EU_05-Klein_Sprundel.pdf

OS X Kernel Mach Exception Ports Local Privilege Escalation Vulnerability
http://www.securityfocus.com/bid/20271
http://www.matasano.com/log/530/matasano-advisory-macos-x-mach-exception-server-privilege-escalation/

This vulnerability occurs because the kernel fails to properly sanitize inherited data from a process invoking a more privileged process. Specifically, an unprivileged process can set up a Mach exception handler that will be inherited by the privileged process. To exploit this, a local attacker could set up a malicious handler in an unprivileged process and then invoke a setuid application in such a way that an exception would occur and the attacker-supplied function would be executed with elevated privileges.

[3] As a second-stage payload, exploitation would depend on the successful exploitation of an existing remote vulnerability.

OS X Airport Wireless Driver Remote Kernel Memory Corruption Vulnerabilities
http://www.securityfocus.com/bid/20144

On September 21, 2006, Apple released a security update addressing three memory-corruption vulnerabilities in the Apple OS X AirPort wireless drivers that could be exploited remotely to execute arbitrary code. Research has conclusively shown that remote kernel vulnerabilities are in fact exploitable in practice; this has been observed on Windows, FreeBSD, Linux, and OS X.

The debate about the presence of vulnerabilities within the wireless drivers distributed with OS X has caused a lot of controversy. What is important, however, is that researchers have demonstrated that they are capable of executing arbitrary code through the use of a wireless driver on OS X, the native AirPort driver or not, and Apple has acknowledged and fixed numerous remote kernel vulnerabilities in the AirPort drivers. These two facts together show that an immediate threat existed and that there likely was, at least to Apple, the real possibility of remote kernel-based code execution on default installations of Apple laptops.

AirPort Update 2006-001 and Security Update 2006-005
http://docs.info.apple.com/article.html?artnum=304420

In addition, a paper has been released regarding 802.11 driver fingerprinting that further exacerbates the potential threat of this type of vulnerability. This allows a remote attacker within geographic proximity of a target system to remotely identify specific drivers known to contain vulnerabilities, such as unpatched versions of the issues reported by Apple, and subsequently to construct an exploit specific to that driver. This allows for an increase in reliability especially when combined with the precompiled distribution, and therefore predictable layout, of those same drivers.

http://www.uninformed.org/?v=5&a=1&t=sumry

System design weaknesses

Mixing Mach and BSD

One of the security benefits often associated with OS X is the inherited system design of BSD. This includes *securelevels*, the ability to run applications in a jailed environment, the UNIX multiuser environment access control model, etc. Although OS X does in fact inherit these security features, a recent disclosure by Nemo noted that it's possible to circumvent the security features of the BSD system because of a flaw in the integration of both the Mach and BSD features into the OS X kernel. Released as part of the Uninformed e-zine, Nemo's paper describes how to use Mach-specific system call services to carry out actions that would normally be restricted when using BSD-equivalent services.

As an example, the securelevel feature of BSD is a rudimentary form of mandatory access control designed to prevent local users from carrying out specific actions when at specific securelevels. This includes lowering the securelevel (even as root), accessing kernel memory, loading modules, etc. However, it turns out that these securelevel restrictions apply only to BSD system services; by careful use of Mach system calls, it's possible to carry out what should be restricted activities, including lowering the securelevel value.

Another example is the `chroot()` system call, which is part of BSD. This call restricts a process to an isolated environment designed to prevent access to files, processes, and devices that are external to that environment. Again, however, these restrictions are applied only to BSD services. Using Mach-specific kernel services, it's possible to effectively break out of the restricted environment and carry out activities on the root of the system.

Further information regarding attacks on these types of weaknesses can be found in the following e-zine article:

Abusing Mach on Mac OS X
http://uninformed.org/?v=4&a=3&t=sumry

Default services and firewall policies

Research has been conducted and made publicly available at security conferences and websites on default services and firewall policies for OS X. On Mac OS X 10.3 (Panther), the firewall is disabled by default; after activating it, only TCP rules are implemented. UDP and ICMP activity is permitted without restriction to the system. On Mac OS X 10.4 (Tiger), it's possible to activate UDP and ICMP filtering, but simply enabling the firewall will not turn them on.

Furthermore, on systems where firewalls are enabled and UDP filtering has been enabled, there are holes in the rules that allow an attacker to contact any UDP service, namely by setting their source port to 67 or 5353, ports associated with the DHCP and Bonjour services.

Research has also shown that by probing Bonjour it's possible to fingerprint the Security Update status of a target host, revealing to the attacker the security posture of the host.

Further information on these discoveries can be found in the following conference given by Jay Beale at Defcon 14.

http://bastille-linux.sourceforge.net/jay/dc14.pdf

mDNSResponder

OS X suffers from a few bad security policies, which are aggravated by the fact that the user can't easily change them. As mentioned previously, the firewall distributed with OS X allows access to UDP services, even after effort has been made to prevent it. Furthermore, at least one application associated with Apple's Bonjour (mDNSResponder) listens by default on UDP port 5353.

A quick look at the changelog of mDNSResponder suggests it's been prone to remotely exploitable vulnerabilities in the past, despite no official Apple advisories specifically mentioning it. The following changelog entries show that significant issues have been present in it before:

```
Revision 1.92  2003/03/27 03:30:55  cheshire <rdar://problem/3210018> Name
   conflicts not handled properly, resulting in memory corruption, and
   eventual crash

Revision 1.26  2004/04/14 19:36:05  ksekar Fixed memory corruption error
   in deriveGoodbyes.

Revision 1.13  2005/09/07 18:23:05  ksekar <rdar://problem/4151514> Off-
   by-one overflow in LegacyNATTraversal

Revision 1.9  2004/10/27 02:25:05  cheshire <rdar://problem/3816029>
   Random memory smashing bug

Revision 1.143  2003/11/14 21:18:32  cheshire
<rdar://problem/3484766>: Security: Crashing bug in mDNSResponder Fix code
   that should use buffer size MAX_ESCAPED_DOMAIN_NAME (1005) instead of
   256-byte buffers.

<rdar://problem/3484766>: Security: Crashing bug in mDNSResponder Fix some
   more code that should use buffer size MAX_ESCAPED_DOMAIN_NAME
(1005) instead of 256-byte buffers.

Revision 1.18  2003/08/15 00:38:00  ksekar
<rdar://problem/3377005>: Bug: buffer overrun when reading long rdata from
   client
```

In addition to the unsettling changelog entries, a quick look through the comments of the code show the following snippet, suggesting that the current version could in fact still be affected by at least one memory-corruption bug.

```
n = recvfrom(g_sUDP, buf, sizeof(buf), 0, // ### !!!Buffer overflow !!!
   // Should be sizeof(buf)-1 to allow for "buf[n] = '\0';" below !!!
                    (struct sockaddr *)&recvaddr, &recvaddrlen);
```

A remotely accessible default service that is accessible despite disabling UDP via the firewall – one that has had a history of security vulnerabilities – poses a fairly high security risk to users. Note that mDNSResponder appears to be specifically designed to handle multicast requests, and therefore connectivity to it may be limited to a local subnet. Nevertheless, the service could be potentially targeted remotely in a variety of scenarios, such as on a wireless network.

serialnumberd

To effectively filter connectivity to UDP ports on OS X you must enter rules manually. However, it was found on OS X Server 10.4 and later that a newly introduced daemon overrides user-supplied firewall rules to allow it to function. Specifically serialnumberd, a service designed to detect unauthorized licenses, automatically adds a firewall rule for itself to receive connections over UDP 626, even if an attempt is explicitly made by the user to prevent it. If serialnumberd were found to contain vulnerabilities, the service would persistently be exposing the owner of the system to attack despite filters that they may have manually put in place. The only way to effectively block access to this service is to use a third-party firewall. More information regarding this serialnumberd behavior is available from the following location:

http://rentzsch.com/macosx/serialnumberd

OS X zero-day issues

A zero-day vulnerability is one that has been discovered but is not yet known by the public or by the vendor. With respect to Apple Mac OS X, it would be safe to speculate that a select group of researchers who have been spending their time researching this platform are aware of numerous zero-day vulnerabilities that affect most or all users. This can be said for almost any, if not every, platform.

An example that supports this theory can be found in a paper written for the Phrack E-Zine (described later) by the researcher Nemo. The paper uses, as examples, undisclosed vulnerabilities in the Safari web browser that were exploited as part of the research for the paper. This is just one known instance of researchers being aware of vulnerabilities not publicly disclosed.

In another instance, Christian Klein and Ilja Van Sprundel (in a presentation on OS X kernel exploitation at Black Hat Europe 2005) hinted that there are kernel-level vulnerabilities that had not yet been patched.

Because of the likely presence of undisclosed vulnerabilities in OS X, users (and Apple in future releases of its operating system) should consider security measures beyond regularly applying patches. These measures are described later in "OS X security technologies and preventive measure" and in "Future threats and malicious advancements facing OS X" below.

Malicious code

To date there has been little in-the-wild malicious code targeting OS X. In February 2006, two new threats were discovered: OSX.Leap, which propagated through iChat, and OSX.Inqtana, a proof-of-concept worm that propagates by exploiting the Apple Mac OS X BlueTooth Directory Traversal Vulnerability (BID 13491).

Older malicious code includes MP3Concept, a harmless program that was never seen in the wild, and SH.Renepo, which was an unsophisticated malicious shell script specifically designed to target OS X. Both MP3Concept and SH.Renepo were discovered in 2004.

MP3Concept

MP3Concept was not significant in any respect. It is harmless, and was never released in the wild. It is included in this document only for completeness.

MP3Concept (Symantec Security Response)
http://securityresponse.symantec.com/avcenter/venc/data/mp3concept.html

OSX.Inqtana

At approximately the same time that OSX.Leap.A was discovered at large, the researcher KF disclosed his proof-of-concept OS X worm to antivirus companies. OSX.Inqtana demonstrated how a worm could be used to propagate over Bluetooth using a vulnerability in the OS X Bluetooth subsystem. Although it was never seen in the wild, the worm also is quite significant because it unveils several attack vectors against the platform that are readily accessible to malicious code. Specifically, the following techniques were discussed and/or demonstrated:

- **Objective C MethodSwizzling**
  Lets you patch/hook functions very easily.
- **InputManagers**
  Hook input; also auto-executes associated bundle.
- **LaunchD Facility**
  Service that launches programs automatically on the system.
- **User-wide Environment Variables / DYLD_INSERT_LIBRARIES**
  This is a central location for adding environment variables to all processes started by a certain user. Combined with the DYLD_INSERT_LIBRARIES environment variable, it allows automatic code injection into all processes created by the affected user.

All of these points discuss legitimate features of the OS X operating system that will be interesting or beneficial to malicious code authors. Many of these facilitate the escalation of an arbitrary file-dropping vulnerability to arbitrary code execution.

Although OSX.Inqtana posed little or no threat for OS X users (because it has been crippled by its author), it did highlight the fact that OS X has many system design quirks that can be leveraged by future malicious code. A more verbose explanation was distributed on security mailing lists and can be found at the link below:

InqTana Through the eyes of Dr. Frankenstein (Bugtraq)
http://www.securityfocus.com/archive/1/425794

As of October 23, 2006 Inqtana has been updated to work against OS X 10.3 and 10.4. Furthermore, its source code has been made available. This was done as part of a hack.lu conference presentation on exploiting Bluetooth vulnerabilities. Since the source code has been made available, new and more malicious variants of the worm could be created. However, it should be noted that the directory-traversal vulnerability at the heart of the Bluetooth exploitation, which allowed the worm to propagate, has been fixed for some time. Creating a malicious new variant would require the addition of an exploit for a new vulnerability of a similar nature to the one that has been fixed.

The source code for Inqtana, a demo of exploitation, and the conference presentation can be found at the following locations:

Inqtana Source and Demo
www.digitalmunition.com/hacklu.html

Hack.lu Bluetooth Exploitation Presentation
http://secdev.zoller.lu/research/hack_lu_2006.pdf

OSX.Leap.A

OSX.Leap.A marked the first in-the-wild worm discovered for OS X. Although not very advanced, this worm is significant because it was not simply a proof of concept developed by a researcher. The worm was found being distributed on a Mac enthusiast forum, pretending to be a collection of images. Observing the responses to the post on the forum indicates that the victims were not carrying out proper security precautions, and in some cases were running the application with root privileges in an attempt to view what they thought were images.

This worm spread using the iChat instant-messaging protocol, a commonly used program distributed with OS X. Although the execution of the worm requires user interaction, this doesn't mean the worm would be ineffective, as proven by the success of mass-mailing worms for other platforms. Upon execution, OSX.Leap uses the native-application-hooking functionality of OS X and ensures that the malicious code will run every time any application is run.

The worm also attempts to identify the four most recently used applications, in order to infect them. This infection is carried out without actually manipulating the file; rather, it carries out a companion infection by manipulating data within the application bundle (directory hierarchy). Specifically, this is done by copying the original application from the data fork to the resource fork, and, in turn, copying the worm code to the data fork.

To propagate itself, the worm monitors the system's use of the iChat program and attempts to send itself in a tarball to every user on the contact list. The worm will carry out this action only if the user runs the iChat program; it does not actively run the iChat program or implement the protocol on its own.

SH.Renepo.B

Renepo is a shell-script that carries out numerous actions on a host system. It is not self-propagating and does not attempt to communicate with a remote system. The most noteworthy functionality of this malicious code includes keylogging, allowing remote SSH access, disabling security software and auto-updates, cracking passwords, and storing logged information and cracked passwords in a hidden folder in a publicly accessible directory.

The main purpose of this virus seems to be to function as a backdoor and to access passwords. Although not self-propagating and relatively unsophisticated, the virus is significant because it shows that even in 2004 malicious scripts for OS X were being distributed in the wild.

SH.Renepo.B (Symantec Security Response)
http://www.symantec.com/avcenter/venc/data/sh.renepo.b.html

OSX.Macarena

OS X Macarena is a proof-of-concept file infector that targets all i386 Mach-O files in the directory where it is run. There are copies that run on both Windows and UNIX. It makes use of a previously known entry point hooking technique, as well as a novel cavity infection technique, to seize control of an application's execution as it is loaded by the OS. It inserts 528 bytes of data to each file infected. This virus was found in October 2006.

Although the public version of the virus appears to be purposely limited to infecting the single directory, modifications to allow for the traversal of directories would likely be trivial. Note that this malware will not work on universal binaries.

http://www.symantec.com/enterprise/security_response/writeup.jsp?docid=2006-110217-1331-99&tabid=1

Exploit development and research

Almost every networked computer operating system in existence has been attacked via system vulnerabilities. Given the number of vulnerabilities that have been discovered in OS X in the past, there will certainly be exploits targeting at least some of them. Many researchers will also develop private exploits for issues they discover and never release them to the public. Currently, only a few — when compared to other platforms — publicly available exploit programs target OS X. This is partly because many of the applications being affected by historic attacks don't require the development of an actual exploit program, but also because this platform has been a less popular target than more widely deployed systems such as Microsoft Windows. Nevertheless, we see an ongoing and increasing volume of research and exploit development targeting OS X.

OS X exploit development

Exploit development on OS X is similar to any other platform. To effectively exploit most bugs, the researcher must be familiar with the processor architecture and the system internals. Attackers and researchers already familiar with exploit development on other UNIX platforms will find OS X almost second nature.

The largest hurdle for most exploit writers and researchers is likely the foreign processor architecture. Apple chose to develop with the PowerPC (PPC) chip, a Reduced Instruction Set Computer (RISC) design, which wasn't previously used on a large scale for desktop or server computers. Most researchers operate on the Intel x86 family of processors, using a Complex Instruction Set Computer (CISC) design. The large differences in the assembly languages may pose a significant hurdle to an attacker, which we believe is one of the reasons more security research on the operating system hasn't been carried out. In the end, however, PowerPC is a relatively straightforward assembly language, and many researchers have begun to overcome the hurdle quite quickly.

Aside from these details, exploit development is identical to that on any other platform. Furthermore, the Metasploit Exploit Framework is readily available to any attacker or researcher wishing to download it, and contains full support for exploits targeting PPC-based systems. Using the framework, an attacker would not even need to write their own payload (a set of instructions in the target platform's assembly language, designed to take control of the computer after triggering a vulnerability). The framework supplies a wide variety of OS X payloads. It is likely that, as a result of this, many of the current OS X exploits have been developed using the framework. These will be described in later sections in this paper.

Although the Metasploit framework would be the logical development platform for many researchers, exploits written in various languages are available. In the end, almost any language can be used to trigger and exploit a vulnerability in OS X.

OS X exploit research

One of the most important indicators of emergent exploit development activity is the active research on the subject. In the past few years, an increasing number of papers and presentations have been released on the topic of attacking OS X. In the last year alone, the amount of information has increased even more.

Stack overflows / payload development

The stack data structure is very simple and identical across most platforms, so new research isn't really

necessary when exploiting a stack-overflow issue on a new platform. As long as the underlying architecture includes branching instructions that save function pointers on the stack (which most, if not all, do), then the basic stack-overflow techniques will work.

When discussing basic exploitation tactics such as stack overflows, research papers often focus on payload development. Numerous OS X payloads are available by the researcher B-r00t, who also wrote a paper discussing the development of such payloads. Other OS X payloads are available in various places online: Metasploit includes a small library, the researcher Dino Dai Zovi has numerous payloads available on his page, and so on.

For a less talented attacker, sometimes the most difficult portion of an attack will be the payload. This is something that could take days or weeks to research and develop. The ready availability of payloads for a platform makes the job easier for a non-sophisticated attacker to construct their own exploits for more basic issues.

http://packetstormsecurity.org/shellcode/PPC_OSX_Shellcode_Assembly.pdf

With the recent move to Intel-based processors, new payloads having been developed to make use of the new instruction set. Furthermore, some payloads have already been released that make use of multi-architecture instruction encodings to allow for a PowerPC and Intel payload to be executed depending on the exploited platform. This is especially valuable to an attacker exploiting a vulnerability for which the architecture is not known but offsets and addresses for either setup can be supplied in a single exploit attempt.

Heap overflows

The researcher Nemo (mentioned throughout this document) released a paper called "OS X heap exploitation techniques" as part of the online Phrack E-Zine. This paper goes into the actual heap algorithm used in the userland heap memory management system that all processes on the system use. After reading the paper, an attacker would have an in-depth understanding of the heap algorithm and would most likely be able to exploit common heap overflows on the platform. The paper describes just one of the more simple, although extremely effective, techniques of heap exploitation and likely represents only a small percentage of the possibilities available when exploiting such heap-related bugs.

http://felinemenace.org/papers/p63-0x05_OSX_Heap_Exploitation_Technqiues.txt

It should be noted that the vm_allocate integer wrapping kernel bug was addressed by Apple soon after the release of this paper. However, the majority of the paper discusses manipulation of the actual heap algorithm itself and therefore cannot be trivially addressed, as the techniques rely on a separate program flaw to be leveraged in the first place. Possible solutions for hardening the algorithm however, could include integrity checks prior to carrying out heap-related functionality to detect memory corruption or anomalous states.

Kernel attacks

Although only a small amount of information has been published about actual kernel exploitation targeting the OS X kernel, numerous papers about attacking other BSD kernels are available, and this information and knowledge can be carried over.

At the Black Hat Europe 2005 security conference, Christian Klein and Ilja van Sprundel, who both work for SureSec, discussed the exploitation of kernel vulnerabilities and demonstrated real-life examples. In the slides presented at the conference, rudimentary kernel shellcode was also released to elevate a userland process's privileges to root (uid=0). Ilja van Sprundel also presented similar and additional information in a separate talk at PacSec 2005.

http://www.itunderground.org/pl/prezentacje/06_slides.pdf
http://pacsec.jp/core05/psj05-vansprundel-en.pdf

At Blackhat 2006, David Maynor and Johnny Cache also demonstrated the possibility of successfully exploiting remote kernel vulnerabilities in wireless drivers to execute a supplied payload. The demonstration was done on a Mac OS X laptop. The payload or exploit used in the attack was never released.

HD Moore has also released a proof-of-concept exploit for the Metasploit Framework, designed to trigger an unpatched memory-corruption vulnerability in the Apple AirPort driver. The issue is said to be exploitable to execute a supplied payload. As soon as kernel-level payloads have been developed and made available to the public for OS X, it is likely that this exploit could be extended to execute code in the future.

Non-exec bypass

Research has been released regarding mechanisms to bypass the newly available non-exec functionality of OS X 10.4 when running on the Intel architecture. A real-world exploit has been released demonstrating the methods required to carry out this attack. More information regarding the non-executable memory protection and how to bypass it is included in the section "Non-executable memory" later in this document.

OS X local exploits

The following list shows some of the notable OS X local exploits available:

Mac OS X CF_CHARSET_PATH Environment Variable Overflow Exploit
http://downloads.securityfocus.com/vulnerabilities/exploits/xosx-cf.c

Mac OS X passwd Utility File Creation and Corruption
http://downloads.securityfocus.com/vulnerabilities/exploits/xosx-passwd.pl

Adobe Version Cue Privilege Escalation Exploit
http://downloads.securityfocus.com/vulnerabilities/exploits/xosx-abode-vcnative-dyld.c

This targets a flaw in a third-party application, demonstrating that non-native applications pose just as much risk to the security of the system as do native features and services.

Mac OS X launchd syslog Format Sting Privilege Escalation Exploit
http://www.digitalmunition.com/FailureToLaunch.pl
This exploit is especially noteworthy as it marks the first publicly available real-world exploit that executes arbitrary code despite the presence of the new non-exec memory technology deployed on OS X when running the Intel Core Duo processor.

OS X remote exploits

The following list shows some of the notable OS X remote exploits available. Some of these vulnerabilities affect native OS X applications developed by Apple. Others affect third-party software for which vulnerabilities get exploited on multiple platforms. Third-party exploits are noteworthy, because they show that researchers are more than capable of porting current, or writing new, exploits to target OS X.

Apple Mac OS X Archive Metadata Command Execution
http://www.securityfocus.com/data/vulnerabilities/exploits/safari_safefiles_exec.pm

This exploit takes advantage of an issue described in BID 16736. It is a high-profile vulnerability because it is reliable and requires no memory corruption. This particular exploit, which is part of the Metasploit framework, allows a webserver to be auto-generated to serve the malicious archive to a victim Safari user.

OS X AppleFileServer Pathname Buffer Overflow Exploit
http://downloads.securityfocus.com/vulnerabilities/exploits/afp_loginext.pl

This exploit targets a stack-based buffer overflow in the OS X Apple file server. Successful use of the exploit will result in the remote execution of a payload by an unauthenticated attacker. The exploit is somewhat limited because it supports only a single target (OS X 10.3.3), but could be trivially modified for additional targets. This is an older exploit, but demonstrates that remote exploits for Apple services are possible and available. This exploit has the advantage of using the availability of the Metasploit payload library and encoders as well.

Samba 'call_trans2open' Remote Buffer Overflow Vulnerability
http://www.securityfocus.com/data/vulnerabilities/exploits/samba_trans2open_osx.pm

This is an exploit for a vulnerability in the Samba third-party service, specifically targeting OS X. This vulnerability saw widespread exploitation on other operating systems, and DeepSight honeypots continue to see exploitation to this day. This exploit is significant because it shows that these types of high-profile vulnerabilities can easily be exploited on a platform like OS X as well as its counterparts. This exploit also has the advantage of using the availability of the Metasploit payload library and encoders.

Although the above list is not at all exhaustive, it demonstrates that client-side and remote exploits in native and third-party vulnerabilities are not only possible, but currently real. We can assume that exploits for other issues patched in Apple advisories have similar, private exploits as well.

OS X kernel exploits

The following list shows some notable OS X kernel exploits that are available. The kernel can be exploited locally and remotely; however, kernel exploits are often considered to be at a technical level above and beyond normal exploits. Reliably exploiting kernel vulnerabilities typically takes much more expertise than exploiting a basic vulnerability. Therefore, kernel exploits are highlighted to show that this area is indeed getting attention. Although no OS X kernel exploits that actually elevate privileges are known to be publicly available, they are known to exist in private. At least one such exploit has been demonstrated at a public security conference.

OS X Mach-O Loading Denial of Service Vulnerability
http://www.felinemenace.org/~nemo/exploits/fm-nacho.c

OS X semop() System Call Proof of Concept
http://downloads.securityfocus.com/vulnerabilities/exploits/macosx_kernel.c

OS X Kernel Mach Exception Ports Local Privilege Escalation
http://cds.xs4all.nl:8081:tmp/excploit.c
This exploit claims to have been written in November 2005, however was disclosed to the public and patched in September 2006.

Apple AirPort Driver Remote Kernel-Level Denial of Service Proof of Concept Exploit
http://downloads.securityfocus.com/vulnerabilities/exploits/daringhucball.rb

This is a PoC exploit released for the Metasploit framework. It triggers an (at the time of writing) unpatched vulnerability in the Apple AirPort code. The vulnerability it targets is said to allow for remote code execution within the context of the kernel. It is possible that this exploit will eventually be updated to contain this functionality.

Rootkits

Rootkits allow an attacker to persist covertly on a system following an initial compromise. Rootkits offer tools that may be valuable to the attacker when using the system for nefarious purposes and can be used to retain unauthorized access to a system for use at a later time. They can also hide system data so they can remain invisible while on the system.

Although OS X has been less of a target for compromise than other operating systems, there are at least three publicly available rootkits: WeaponX, OSXRK, and Togroot. The following describes what these rootkits allow an attacker to accomplish and how easy they are to use.

WeaponX

WeaponX was developed by Nemo, a member of the group FelineMenace, in 2004. The rootkit is designed to reside in the kernel as an extension, rather than simply as a set of modified userland tools. It offers a small set of features that allow local privilege elevation, hiding the presence of users and open ports, and, of course, hiding its presence in the kernel. The following is taken from the rootkit's README file, which lists the kit's exact features:

```
[*] setuid(1337) will leave user with uid=0.
[*] Hides itself from kextstat.
[*] Give a chosen process uid=0 with signal 1337.
[*] Hides a given port from netstat.
```

```
[*] Hides a user from w and who.
```

WeaponX hooks system calls, allowing the attacker to manipulate the data that would normally be returned. In some cases, the original function won't be called at all. This common technique is valuable since it allows the information-hiding to be more or less application independent, blocking data from any utility that uses the system call.

This rootkit is open source and could be easily extended to include new and more advanced functionality.

The author of this rootkit is known to still be actively carrying out OS X research. As such, we assume that private versions of WeaponX beyond v0.1 have likely been developed and are possibly available in the underground. Other researchers and attackers have also likely created their own private versions of the kit.

> **NOTE**: Due to significant changes in the kernel used by OS X 10.4, this rootkit will no longer work. It was originally designed for use with the 10.3 kernel; numerous changes would be required to make the code work on 10.4.

The rootkit is simple to use, requiring only compilation and installation using the `kextload` utility. This process is automated by simply running the `make` and `make install` commands.

A new version of this rootkit is allegedly under development. Also written by Nemo, this new version avoids the use of kernel modules completely and accesses kernel memory directly through the use of Mach system calls. The kit will employ some of the research available in a paper mentioned earlier in this document.

OSXRK

Also developed in 2004, the "OS X Rootkit" (OSXRK) was written by a researcher named g@pple. It is freely available to the public. Unlike WeaponX, OSXRK is a userland rootkit. Userland rootkits are typically considered less advanced, although in some cases more reliable, because they are typically designed to replace commonly used applications with modified versions that will prevent certain information from being displayed. OSXRK doesn't even replace system binaries or attempt to hide information; instead, it simply downloads a set of handy tools for the attacker to use on the system. These tools include a backdoor, system log cleaner, password cracker, sniffer, etc. These are simply installed using a collection of shell scripts that are run on the system following a compromise.

Overall, OSXRK is quite primitive and doesn't grant an attacker much covertness. Also, the tools do little to hide the attacker's actual presence, allowing for a compromised user or administrator to detect their presence quite easily. Because this particular kit is simply a set of tools and scripts, it would be easily extensible.

Togroot

This is the simplest of the rootkits discussed in this document. Originally released in 2004, Togroot is the only rootkit for which no source code is available. The kit is a simple kernel extension that, once loaded, allows an attacker to obtain root privileges on the system by carrying out a simple sequence of commands. Specifically:

```
$ /givemeroot
$ su
```

This particular rootkit is distributed as a pre-compiled kernel extension with a simple set of loader and unloader scripts. An attacker could easily install the kit on a system by simply running the scripts.

The author suggests that further development of the rootkit was being carried out in private and that a more advanced version would eventually be released. The DeepSight Threat Analyst Team has not found any newer versions, but we assume that more advanced private versions have been developed. Note that the currently available version was developed for OS X 10.3 and therefore may not function on the newer 10.4 release.

Defenses

Administrators and users of OS X systems should follow the same basic common-sense principles that apply to securing any information system. Although OS X-specific firewalls and other everyday security tools and practices are not discussed in this document, we may analyze and discuss them in future publications.

The following are the best ways to identify and remove threats that may affect OS X computers.

Detecting malicious code

Numerous antivirus products are available for the OS X platform and can be used to preemptively detect known threats before they're installed or executed on a targeted system. Other common practices (such as running applications with the lowest privileges possible and using strict ingress and egress firewall rules) can greatly help prevent malicious code from running on an OS X computer.

One of the most important precautions to fight against malicious code targeting OS X is *user education*. Possibly because OS X has historically been untargeted by malicious code authors, many Mac users may be unsuspecting of potential nefarious or malicious activities being carried out by attackers. In some cases (such as the unsophisticated distribution of OSX.Leap.A on a forum telling people to open a file), infection could have easily been avoided if users had taken basic security precautions. A concerning trend among OS X users is relatively carefree downloading of applications, disk images, desktop widgets, and other potentially malicious data. These may serve as distribution vectors for malicious code in the future. Simply downloading random programs and files is an easy way to get infected with Spyware, Adware, or malicious code on Windows and other systems. This threat is compounded by an unfortunate perception of immunity to malicious code and a general lack of understanding of basic computer security, reinforced by a notable lack of high-profile incidents and popular opinion. The same caution should be exhibited by users and administrators as with any other information system.

Detecting rootkits

The `chkrootkit` tool, which is freely updated, has recently been enhanced (as of 0.46A) to include support for Mac OS X. Although it discovers a threat *after* a compromise, we still highly recommend it as part of a secure system implementation. The `chkrootkit` tool can be obtained from the following site:

chkrootkit
http://www.chkrootkit.org/

Current antivirus technologies (such as ClamAV, Symantec AntiVirus, and others) also can detect OS X rootkits.

Protecting against vulnerabilities and exploits

OS X provides an auto-updating feature that applies all security patches as soon as they are released. This is the best method of ensuring that vulnerabilities will not be exploited. Strict access controls and ingress/egress filtering will also help prevent remote exploits from affecting your hosts.

The Intel processor deployed in newer Mac systems makes use of the XD bit (explained later in this document) to force non-executable memory in some portions of process address space. Since this may hamper an attacker's ability to exploit certain vulnerabilities, we highly recommend employing this technology.

Locking down services and firewall policies

Jay Beale has developed a version of Bastille that is designed to lock down the problems with services and firewall policies discussed in this document, as well as many others. Bastille is available for OS X 10.2 through 10.4. Jay has given several talks at security conferences detailing the security enhancements Bastille provides to OS X (see the locations below). Some of these enhancements include:

- Disabling unused services that are normally left on
- Deactivating Bonjour (including mDNSResponder)
- Implementing fully configurable and functional firewall
- Enforcing more strict system and user permissions

Additional information on Bastille, as well as the relevant packages to download, are available on the Bastille website and in the Defcon 14 presentation given by Jay Beale.

http://www.bastille-linux.org/osx.html
http://www.bastille-linux.org/jay/dc14.pdf

Note that it is not currently known if Bastille disables the serialnumberd daemon mentioned earlier in the paper.

OS X future threats and defenses

As with any platform, there is room for improvement in ways to defend against threats and ways for attackers to exploit systems. With the release of OS X on the x86 platform, the operating system has become more accessible to researchers; it can now be run on a variety of emulators and virtual machines that support the 64-bit format. At the time of writing, unauthorized OS X installation disks and images are available that allow OS X to be run within VMware. This new development will likely increase research into the OS, since access to the platform no longer requires the purchase of special hardware.

Malicious code

All other possibilities aside, the previous sections of this document demonstrating the existence of numerous critical vulnerabilities, exploits, and rootkit technology should show that simply combining these readily available tools and problems with a more destructive and viral payload could easily yield higher-profile malicious code. Other research in the area of traditional virus techniques is also taking place in the security community.

For instance, Nemo, the author of the WeaponX rootkit described earlier, gave a presentation on file infection of the Mach-O file format. The Mach-O file format is used by Mac OS X as the "native" file format, as other BSDs and Linux use ELF and Windows uses PE. The presentation describes the creation of trivial Mach-O concatenation viruses, resource fork infectors, traditional entry-point hooking infection, kernel infection, objective C infection, multiple-architecture binary infection (FAT format), and other topics. The document provides links to publicly available proof-of-concept examples of each topic.

Some of the techniques described in the presentation have already been deployed by public malicious code (such as the previously described Inqtana and Leap malware). Some of the more advanced methods have yet to be observed in the wild, but will likely be used in the future. The presentation is available from the following location:

http://www.felinemenace.org/~nemo/slides/mach-o_infection.ppt

Rootkits

As of the release of Mac OS X 10.4, additional security precautions have been added to the OS X kernel. These precautions prevent the currently public versions of OS X kernel-based rootkits from interfacing with the kernel. As time goes by, public rootkits capable of functioning on the newer kernel will likely be released. We have looked into the changes required to make current open-source kernel rootkits function on 10.4; an attacker with a decent skill level would be able to make the changes in a short period of time.

Another area that may begin to gain more popularity is the coupling of kernel-level vulnerabilities with rootkit-like payloads. This would allow an unprivileged user, possibly one exploiting a unprivileged process, to leverage a local kernel vulnerability to install a rootkit into privileged memory and then further elevate privileges across the system. Note that this is an area where all operating system platforms will likely see advancements.

It should be noted that for a brief period after the release of OS X for the Intel architecture, the kernel source code was made unavailable. It has since be made available again. This may have presented a possible hurdle for some rootkit developers, however it is believed that it would've been overcome fairly quickly.

In a recently released paper in the Uninformed journal, Nemo published information on a new method of accessing kernel memory on both the Intel and PowerPC architectures and has stated that a new version of WeaponX is being developed to leverage this new technique. It is possible that the new version will be developed to work against both architectures.

Vulnerabilities and exploits

Having faced a set of established threats, every platform will have developed some security technologies to protect against those threats. Furthermore, most platforms have numerous areas of improvement regarding the security technologies deployed to further reduce the severity of threats. On the flip side, attackers will always seek areas of improvement in their malicious code and exploits and will develop ideas and techniques that may be unconventional or that can circumvent certain established security measures. This constitutes an arms race that is ever present between security researchers and operating system vendors.

More vulnerabilities will be discovered in Mac OS X as long as new features keep being included. Every increase in complexity increases the likelihood of introducing bugs. For example, see the following link,

which describes all the additional features (and thus increased complexity and code) that OS X Tiger has over its predecessors.

Tiger Feature Comparison
http://www.apple.com/macosx/upgrade/compare.html

OS X security technologies and preventive measures

Given that there are numerous security threats targeting OS X and that this number is only set to increase, it is possible that Apple may look into implementing some system-level security enhancements to OS X. Currently, aside from the usual user-privilege-based security measures, Apple offers little in the way of exploit prevention in their operating system. We analyze some of these security measures that may help and will hopefully be implemented in the future.

One of the most important steps in keeping a platform secure is carrying out extensive source-code audits to proactively remove vulnerabilities. This is an area where Apple has definitely been improving. Older versions of the Darwin kernel used by OS X, for example, were littered with integer-related bugs. These have been fixed; and looking over the source code shows that, although the code is not perfect, these types of vulnerabilities have been widely addressed. This shows that Apple is aware of the threats, understands them, and has proactively eliminated many potential instances.

Stack canary values

A *stack canary* is technology designed to prevent a buffer overflow. It is typically implemented as a compiler modification that will store a special value before sensitive memory values in process memory. The value is typically situated before a function return address and saved base pointer, since these pointers are both the normal targets when exploiting stack-based buffer overflows. Like a canary in a coal mine, the namesake of this technology, disruption of this special value means that something bad is happening. When a function exits, the special canary value is tested against a saved value to ensure it has not been corrupted; if it has, then the program terminates due to the detection of a buffer overflow.

Stack canaries are not a new technology. However, they have only recently been adopted by some of the larger operating systems. Windows 2003 and Windows XP SP2 native libraries were compiled with built-in canary checks. By default, OpenBSD now compiles all applications using ProPolice, an implementation of stack canary protection built into the gcc compiler, but normally disabled by default. Numerous Linux vendors enable this ProPolice functionality or offer special secure installations that do as well.

Since OS X is a BSD-based operating system, ProPolice functionality should be accessible to Apple and will hopefully be enabled by default in OS X applications at some point. Although not a complete protection, this security measure presents a significant hurdle for an attacker to overcome and still successfully exploit a stack-based vulnerability.

Secure heap implementation

A number of security mechanisms that can be implemented into a heap algorithm can help prevent exploitation. Many heap algorithms have already begun to incorporate security features: `ptmalloc2` used by Linux, OpenBSD's new `mmap`-based heap, and the current Windows NTDLL (and upcoming Vista) heap have all recently made changes to help prevent exploitation. Although not offering guaranteed protections, these also add significant protection against attackers and may make many bugs unexploitable.

Address Space Layout Randomization

ASLR is a technology that has had mixed integration into operating systems, but presents a significant hurdle in the reliable exploitation of memory-corruption vulnerabilities. It also has a significant effect on the impact and proliferation of worms relying on specific vulnerability classes. Currently, a number of operating systems are beginning to implement rudimentary forms of ASLR; others have full-blown implementations available.

As its name suggests, ASLR works by randomizing the contents of memory for each process. Every time a process is invoked, these addresses change position. This technology plays on the relocatability of shared libraries and other executable files, which is beyond the scope of this paper. However, what is important is that with randomized memory, an attacker can no longer simply place a static memory address into an exploit program and have it reliably exploit multiple systems. The predictability of memory locations is one of the biggest security problems, because successful prediction can allow worms to spread so prolifically. Currently, like most releases of Windows[4], OS X contains no memory randomization and also distributes precompiled binaries. This poses a significant risk where worms are concerned, since a single exploit could work across multiple systems and possibly operating system releases.

OpenBSD, the PaX Linux kernel patch and some native Linux kernel code, and some newer Windows releases all contain forms of memory randomization. This technology does not adversely affect the functionality of programs and requires little overhead, yet poses a significant hurdle in the exploitation of all memory-based exploitation vectors. Other higher-level vulnerability classes are not affected, however, such as remote command execution, SQL injection, etc.

Non-executable memory

Of all the security measures discussed, non-executable memory has historically been the most difficult to implement on most common operating systems. This is due to the ubiquity of the Intel 32-bit x86 platform, which does not natively support this technology. Some groups, such as the Linux PaX developers, have come up with methods for successfully emulating non-executable memory, but many groups feel that it isn't stable enough or is too severe a workaround to implement in their operating systems.

Mac OS X, however, has until recently made exclusive use of the PowerPC processor, which is not affected by this limitation. Although some versions of PowerPC processors do support non-executable memory, we are not aware of this technology being implemented by OS X. If versions of OS X continue to be released for the PowerPC architecture, it would be a significant improvement in security if Apple were to take advantage of this technology.

As of the official release of OS X for x86, which makes use of the 64-bit Intel Core Duo processor, non-executable memory is taken advantage of, to a small extent. Although the x86 platform does not natively support non-executable memory, the Duo makes use of the EM64T 64-bit extensions, which are compatible with AMD64. The EM64T extensions include support for the XD bit (equivalent to the AMD64 NX or non-executable bit), which effectively allows for non-executable memory to be implemented. So far, it would appear that OS X implements this non-executable memory feature only within stack memory. At the time of writing, the researcher KF has already demonstrated that this feature can be trivially circumvented. Further information on this can be found from the following location:

Non eXecutable Stack Lovin on OSX86
http://www.securityfocus.com/archive/1/434831/30/0/threaded

[4] Windows XP SP2 and Windows 2003 SP1 have small implementations of ASLR. Windows Vista makes more thorough use of this technology.

Future threats and malicious advancements facing OS X

One of the biggest threats posed against the OS X platform is the non-randomized memory (described earlier) combined with pre-compiled binary programs used across most users' systems. Like Windows, this grants attackers a good opportunity for creating malicious code that can propagate effectively across numerous systems, possibly running different versions, with little extra effort and hardly any sophistication. To date, this area of OS X has not been widely targeted. This is likely because the target OS has not been as popular as other systems and because the underlying hardware architecture is unfamiliar. With the recent move to Intel x86 processors, however, Apple users can almost assuredly be considered at a higher risk of attack than ever before. For a long time, the x86 platform has been the most widely exploited, and therefore most attackers and researchers specialize in programming for that particular assembly language. The move to the x86 architecture will make converting already built payloads and other programs that target x86 much easier for attackers to aim at OS X.

As with security mechanisms, there are usually many areas that attackers can improve on to make their attacks more covert, more malicious, and more sophisticated.

Conclusion

Security vulnerabilities affect all operating systems. This document has described the current threat landscape affecting OS X, showing that:

- Critical vulnerabilities are being discovered on a regular basis.
- Exploits for these flaws are being actively developed and released to the public.
- Post-compromise tools are being researched and released.
- Self-propagating malicious code is being created for the platform, although it is still in its infancy.

In the end, users and administrators of OS X should be taking the same security precautions as with other platforms and should carry out secure computing practices to avoid exploitation or infection.

Although this document has discussed many areas of security affecting the OS X platform, each individual section of content was not exhaustively covered and deserves additional research and analysis. As new threats and security measures are introduced for the platform, the DeepSight Threat Analyst Team will continue to carry out analysis of these topics and to inform customers of any new information of concern or interest.

Resources

Mac OS Release Timeline
http://www.macos.utah.edu/Documentation/MacOSXClasses/macosxone/macosxhist.html#server

Mac OS X
http://en.wikipedia.org/wiki/Mac_OS_X

Apple Security Updates
http://docs.info.apple.com/article.html?artnum=61798

OpenDarwin
http://www.opendarwin.org/

SecurityFocus Vulnerability Database
http://www.securityfocus.com/bid/

Community Credits

The DeepSight Threat Analyst Team would like to acknowledge the following security researchers for their publicly available information and ongoing efforts regarding OS X.

Andrew G
http://www.felinemenace.org

Neil Archibald (Nemo)
http://www.felinemenace.org/~nemo/

Ilja Van Sprundel
http://blogs.23.nu/ilja/

Dino Dai Zovi
http://www.theta44.org/main.html

Kevin Finisterre (KF)
http://www.digitalmunition.com/

Metasploit Developers
http://www.metasploit.com

Change Log

Version 1: July 26, 2006
Initial Research Report released.

Version 2: November 13, 2006
Updated with new information / public release.

Glossary

If you are unfamiliar with any term this report uses, please visit the SecurityFocus glossary at http://www.securityfocus.com/glossary for more details on information security terminology.

Contact Information

World Headquarters

Symantec Corporation
20300 Stevens Creek Blvd.
Cupertino, CA 95014
U.S.A.
+1 408 517 8000
www.symantec.com

About Symantec

Symantec, the world leader in Internet security technology, provides a broad range of content and network security software and appliance solutions to enterprises, individuals, and service providers. The company is a leading provider of client, gateway, and server security solutions for virus protection, firewall and virtual private network, vulnerability management, intrusion detection, Internet content and e-mail filtering and remote management technologies, as well as security services to enterprises and service providers around the world. Symantec's Norton brand of consumer security products is a leader in worldwide retail sales and industry awards. Headquartered in Cupertino, Calif., Symantec has worldwide operations in 38 countries. For more information, please visit www.symantec.com.