Apple iOS 4 Security Evaluation

Dino A. Dai Zovi
Trail of Bits LLC

Focus of This Talk

- What enterprise users need to know about iOS security features and properties to make informed deployment, configuration, usage, and procedure decisions
- Assorted iOS implementation details and internals
- Interesting places for reverse engineers and vulnerability researchers to look (if they are paying close attention)

iOS 4 vs. iOS 5

- Except where stated otherwise, everything in this talk refers to the latest versions of iOS 4
  - I’m still working on analyzing iOS 5
  - Updates on material for iOS 5 will be presented at a future conference

Overview

- Introduction
- ASLR
- Code Signing
- Sandboxing
- Data Encryption

Introduction

Security Concerns

- Sensitive data compromise from lost/stolen device
  - What data can be recovered by attacker?
- Malicious Apps
  - What is the likelihood of DroidDream for iOS?
- Remote attacks through web browser, e-mail, etc.
  - Is that a desktop (aka APT target) in your pocket?

Address Space Layout Randomization

Security Concerns

- How hard is it to remotely exploit built-in or third-party applications?
  - Malicious web page in Safari or third-party app with embedded browser (i.e Facebook, Twitter)
  - Malicious e-mail message or attachment in Mail
  - Man-in-the-middle and corrupt network communication of third-party apps

iOS 4.3 ASLR

- ASLR is a common runtime security feature on desktop and server operating systems and is a good generic protection against remote exploits
- iOS 4.3 introduced ASLR support
  - iOS 4.3 requires iPhone 3GS and later (ARMv7)
- Apps must be compiled with PIE support for full ASLR, otherwise they only get partial ASLR
  - iOS 4.3 built-in apps and executables are all PIE

Partial vs. Full ASLR

PIE	Main Executable	Heap	Stack	Shared Libraries	Linker
No	Fixed	Randomized per execution	Fixed	Randomized per device boot	Fixed
Yes	Randomized per execution	Randomized per execution (more entropy)	Randomized per execution	Randomized per device boot	Randomized per execution

PIE in Real-World Apps?

Top 10 Free Apps (July 2011)

App	Version	Post Date	PIE
Songify	1.0.1	June 29, 2011	No
Happy Theme Park	1.0	June 29, 2011	No
Cave Bowling	1.10	June 21, 2011	No
Movie-Quiz Lite	1.3.2	May 31, 2011	No
Spotify	0.4.14	July 6, 2011	No
Make-Up Girls	1.0	July 5, 2011	No
Racing Penguin, Flying Free	1.2	July 6, 2011	No
ICEE Maker	1.01	June 28, 2011	No
Cracked Screen	1.0	June 24, 2011	No
Facebook	3.4.3	June 29, 2011	No

Bottom Line

- All built-in apps in iOS 4.3 have full ASLR with PIE support
- Third-party apps are rarely compiled with PIE support and run with partial ASLR
  - Static location of dyld facilitates exploitation by providing known executable material at a known place (code reuse, return-oriented programming, etc)
  - Applications using a UIWebView are the highest risk (embedded browser in Twitter, Facebook, etc)

Code Signing

Security Concerns

- Can this application be trusted to run on my device?
  - Who (real-world entity) wrote it?
    - How do we know it's really them?
- Does it have any hidden functionality?
  - Can it change functionality at run time?

Code Signing

- Mandatory Code Signing
  - Every executable binary or application must have a valid and trusted signature
  - Enforced when an application or binary is executed
- Code Signing Enforcement
  - Processes may only execute code that has been signed with a valid and trusted signature
  - Enforced at runtime

iOS 4.3 Adds JavaScript JIT

Apple Introduces iOS 4.3

Update Includes Faster Safari Performance, iTunes Home Sharing, AirPlay Improvements & New Personal Hotspot

SAN FRANCISCO—March 2, 2011—Apple® today introduced iOS 4.3, the latest version of the world's most advanced mobile operating system. New features in iOS 4.3 include faster Safari® mobile browsing performance with the Nitro JavaScript engine; iTunes® Home Sharing; enhancements to AirPlay®; the choice of using the iPad™ side switch to either lock the screen rotation or mute the audio; and the Personal Hotspot feature for sharing an iPhone® 4 cellular data connection over Wi-Fi.

"With more than 160 million iOS devices worldwide, including over 100 million iPhones, the growth of the iOS platform has been unprecedented," said Steve Jobs, Apple's CEO. "iOS 4.3 adds even more features to the world's most advanced mobile operating system, across three blockbuster devices—iPad, iPhone and iPod touch—providing an ecosystem that offers customers an incredibly rich experience and developers unlimited opportunities."

The Safari mobile browsing experience gets even better with iOS 4.3. The Nitro JavaScript engine that Apple pioneered on the desktop is now built into WebKit, the technology at the heart of Safari, and more than doubles the performance of JavaScript execution using just-in-time compilation. With the Nitro JavaScript engine, Safari provides an even better mobile browser experience working faster to support the interactivity of complex sites you visit on a daily basis.

Dynamic Code Signing

- The **dynamic-codesigning** entitlement allows the process to map anonymous memory *with any specified protections*.
- Only MobileSafari has this entitlement in iOS 4.3
  - Necessary for JavaScript native JIT (“Nitro”)
  - Previously MobileSafari did bytecode JIT

Bottom Line

- Mandatory Code Signing in iOS is strong defense against execution of unauthorized binaries
  - Requires incomplete code signing exploits to bypass and obtain return-oriented execution
- Code signing forces attackers to develop 100% fully ROP payloads
  - DEP/NX only require a ROP stage to allocate new executable memory and copy shellcode into it
- JIT support in Safari reduces ROP requirements to a temporary stage, similar to what is needed to evade DEP/NX

Sandboxing

Security Concerns

- Can an exploited app or malicious third-party app...
  - Access or modify data belonging to other applications?
  - Access or modify potentially sensitive user information?
  - Break out of the sandbox and rootkit iOS?

App Container Profile

- See whitepaper for detailed description and tarball for fully decompiled profile
- Summary:
  - File access is generally restricted to app's home directory
  - Can read user's media: songs, photos, videos
  - Can read and write AddressBook
  - Some IOKit User Clients are allowed
  - All Mach bootstrap servers are allowed

Bottom Line

- Remote exploits are most likely able to break out of sandbox by exploiting iOS kernel or IOKit UserClients permitted by sandbox profile
- Rogue applications would need to exploit and jailbreak the kernel to escape sandbox
  - Could repurpose kernel exploits from Jailbreaks
  - Apple's review will likely not catch this, but they could use kill switch to remove app from all users' devices
  - OTA app distribution bypasses Apple's review (target user interaction required)

Data Encryption

Security Concerns

- What sensitive data may be compromised if a device is lost or stolen?
  - What data is encrypted?
  - What data is protected by the passcode?
- How hard is it to crack iOS passcodes?
  - Can they be cracked off the device?

Data Protection API

- Applications must specifically mark files on the filesystem and Keychain items with a Protection Class in order for them receive greater protection
  - Files and Keychain items can be made inaccessible when the device is locked (protected by Passcode Key)
  - Keychain items can be made non-migratable to other devices (protected by UID Key)

Data Protection Coverage

- In iOS 4, DP is only used by the built-in Mail app
  - Protects all mail messages, attachments, and indexes
  - Protects passwords for IMAP, SMTP servers
  - Protected items are only accessible when device is unlocked
  - Exchange ActiveSync passwords are accessible always to preserve remote wipe functionality
- DP is also used for automatic UI screenshots generated when user hits the Home Button.

Attacking Passcode

- With knowledge of passcode, you can decrypt the data protected by iOS Data Protection
- Increasing incorrect passcode delay and forced device wipe after too many incorrect guesses are enforced by UI
  - Springboard -> MobileKeyBag Framework -> AppleKeyStore IOKit UserClient -> AppleKeyStore Kernel Extension
- On a jailbroken device, you can guess passcodes directly using the MKB Framework or AppleKeyStore IOKit User Client
  - Jailbreak device using BootROM exploit, install SSH bundle, restart, and log in via SSH over USBMUX

Worst-Case Passcode Guessing Time (iPhone4)

Passcode Length	Complexity	Time
4	Numeric	18 minutes
4	Alphanumeric	51 hours
6	Alphanumeric	8 years
8	Alphanumeric	13 thousand years
8	Alphanumeric, Complex	2 million years

Assuming 26 lowercase letters + 10 digits + 34 complex characters = 70 chars

Bottom Line

- 6-character alphanumeric passcodes sufficient
  - Unless attacker can extract UID Key from hardware
- Lack of thorough Data Protection coverage is a serious issue
  - Wait to see what iOS 5 covers
  - Audit third-party apps for Data Protection usage
- iPad2 and iPhone 4S have no public Boot ROM exploits, making attacks on lost devices much more difficult and unlikely

Conclusion

Summary of Findings

- Although filesystem is encrypted with block-level encryption, exploiting the device's BootROM and booting jailbroken can be used to read the data
- In iOS 4, Data Protection only protects Mail messages and passwords (and screenshots) with user's passcode
- While passcode must be cracked on-device, default simple passcodes are brute-force cracked in less than 20 minutes

Attacks You Should Care Most About

- Lost/stolen device
  - How well is iOS and third-party app data protected?
- Repurposed jailbreak exploits
  - JailbreakMe PDF attacks via e-mail or web
- Stolen Enterprise In-House Distribution Certificate and social engineering OTA app links
  - Apps containing kernel exploit from JB

Questions?

- Slides, code, and paper available on blog: http://blog.trailofbits.com
- @dinodaizovi / ddz@theta44.org
- Shameless self-promotional plug:

  Pre-order *The iOS Hacker's Handbook* by Charlie Miller, Dion Blazakis, Dino Dai Zovi, Stefan Esser, and Ralf-Philip Weinman today!