Module 53
(iOS Hacking)

• At the end of this Module, students will be able to discuss the development history of iOS and the evolution of its security model. They will understand the two methods of jailbreaking. They will be able to identify the dangers inherent in jailbreaking and be able to cite several different attacks that have been launched against iDevices. They will understand the limitations of trustworthiness that are associated with the App Store and why one cannot be 100% certain that an App Store app does not contain malware. Finally, they will understand what measures can be taken to reduce the risk of malware exploitation.
iOS History

- Development began in 1980s at NeXT with NeXTSTEP operating system.
- OS Based on Mach kernel and BSD.
- Objective-C programming language adopted.
- Apple acquired NeXT in 1996 converting NeXTSTEP/OPENSTEP to Mac OS9, then Mac OS X
- 1977 iPhone iOS based on NeXTSTEP/Mac OS X family.
- iPhone followed by iPads and Apple TV
- All iOS devices based on ARMv6 and ARMv7
iOS Security

• After initial release, Apple said no third-party apps would run on the iPhone.
• Developers were told to use web apps instead.
• Hackers discovered how to jailbreak the devices and in 2008 Apple released iOS with support for the App Store which hosted third-party apps.
• Early iOS versions:
  - All processes run as superuser.
  - No sandboxing
  - No code signing
  - No ASLR
  - No position independent execution for libraries or apps
  - Few hardware controls
Security Added

• Improved security
  - Third-party apps run as *mobile* user
  - Sandboxing restricts apps to limited set of resources
  - Code signature verification added (Apple signing required) both load time and run time
  - ASLR for operating system components and libraries and pos'n indept. code compile time option for Xcode.
  - Now one of the most secure consumer-grade operating systems available. Evidence: price for iOS exploits on the grugq's menu is highest.
  - Still has weaknesses
Jailbreaking Caveats

• Typical rooting concerns apply.
• In addition, code signature validation is disabled. Thus, unsigned malicious code is able to be run.
• Jailbreaks may be
  – Boot-based
  – Remote
Boot-based Jailbreaking

• Uses the phone's firmware update procedure
• Phone firmware image stored in IPSW file.
• Download appropriate IPSW file, run jailbreak software to modify and install this firmware.
• Process is somewhat tedious and involves placing device in DFU (device firmware upgrade) mode.
• **Extreme** Cydia jailbreak vulnerability: default ssh password for root: alpine.
Remote Jailbreaking

• Example: jailbreakme
• Involves download of pdf file that exploits a browser vulnerability.
• Typically center on MobileSafari bugs.
Hacking the iDevices of Others

- Devices have small (practically nonexistent) network profile. No services.
- Exploits aim at client-side vulnerabilities, local network access, or physical access. The pineapple at Starbucks is a typical local network access attack. (Requires Wireless connectivity to be enabled.)
JailbreakMe3.0 (JBME3.0)

- Usually used by device owner, but same vuln can be used for remotely rooting another user's device.
- Exploits two vulnerabilities
  - CVE-2011-0226 FreType Type 1 Font-handling error allows arbitrary code execution
  - CVE-2011-0227 IOMobileFramebuffer bug leads to arbitrary code execution as superuser.
- Countermeasures:
  Don't read pdf documents? Update! Update! Update! Update!
iKee

• November 2009, IP blocks in Netherlands and Australia targeted by code that scanned for open port 22 using root:alpine as credentials.
• Would Rickroll your wallpaper.
• Countermeasures:
  – Don't jailbreak
  – Change default passwords!
FOCUS 11 Attack Demo

- Demoed at FOCUS 11 Conference
- Used Wifi MITM on iPad.
- Routed traffic through conference wifi to original destination.
- Trapped gmail request and served up modified gmail html code that included an iframe containing pdf with modified JBME3.0 code that acted silently and did not install Cydia icon.
- Installed ssh and vnc on device
- Countermeasures:
  - Update! Update! Update!
  - Select ask to join networks out of the box
Malicious(?) Apps in the App Store – Handy Light

• Purported to be a flashlight app
• Actually provided tethering capability (banned by Apple for iPhone apps)
• To initiate tethering, a special sequence of UI interactions was required.
• App store checking software did not identify this functionality and approved the app.
Malicious(!) Apps in the App Store – InstaStock

- Provided real-time stock ticker checking.
- Developed by Charlie Miller (iOS hacker, twitter employee, ex NSA employee)
- Exploited iOS signature validation bypass functionality intended to be used only in MobileSafari. Allowed Just In Time (jit) compilation of Javascript.
- InstaStock could connect to CNC server for receive and execute commands to upload images and contact info infected devices.
- Charlie Miller was banned from the App store.
Malicious Software for Jailbroken Phones

• FlexiSPY
  - Eavesdrop on phone calls
  - Collect SMS messages
  - Turn on microphone/camera
  - GPS
  - Remote install/uninstall

• Apple would content that this is not iPhone malware since it can only be installed on jailbroken phones. What do you think?
Countermeasures for App Store Vulnerability

- Apps cannot scan the filesystem (due to sandbox limitations)
- Intego Virus Barrier scans email attachments and Safari downloads (better than nothing)
- I can find no mention of malware scanning software for jailbroken iDevices.