iPhone security model & vulnerabilities

Cedric Halbronn
Jean Sigwald
Sogeti / ESEC
cedric.halbronn(at)sogeti.com
jean.sigwald(at)sogeti.com

HITB SecConf 2010

SOGETI

Introduction

- The iPhone is (one of) the most popular Smartphone(s)
- Enterprise features: VPN, Exchange, etc.
- Closed platform → jailbreak
- Owned this year at PWN2OWN
- Browser-based jailbreak released in August
  → Was patched one week later
- BootROM exploits for all devices since last week
- What are the possibilities for an attacker ?

Plan

iOS introduction

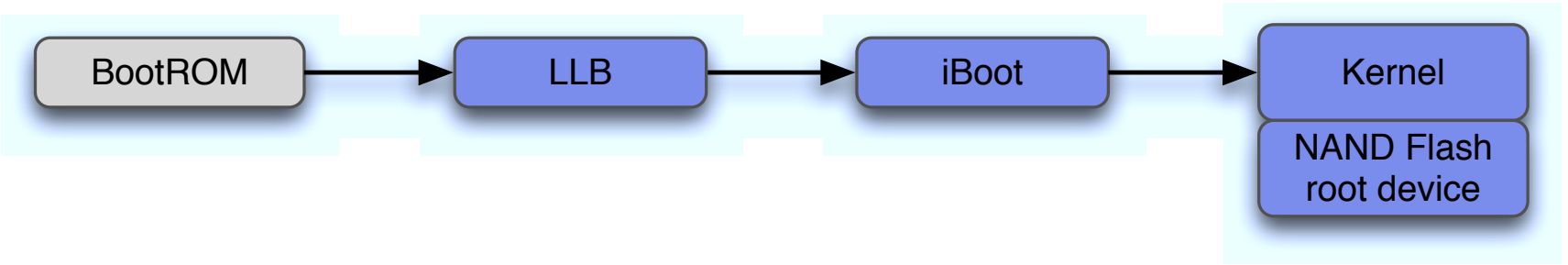
iPhone Operating System

- Runs on the application processor (ARM core)
- Based on Mac OS X
- 4 major releases

Components

- Bootloaders
- Kernel
- System software, shared libraries, built-in applications
- Uses 2 HFS+ partitions on flash: system (read only) and user data/applications

Trusted boot



Trusted boot

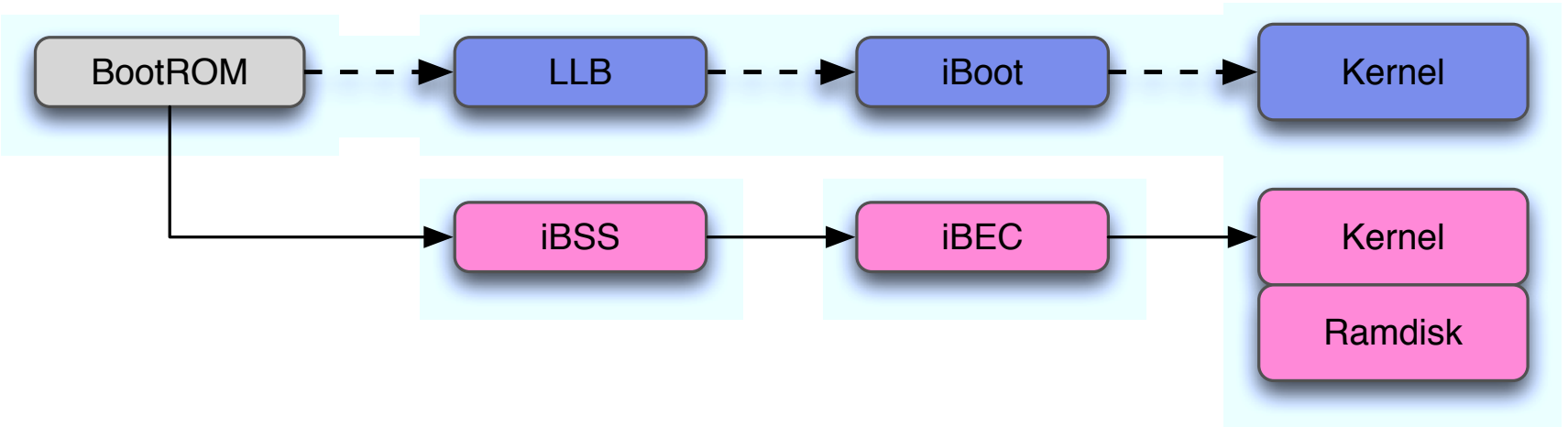
Chain of trust

- Apple root certificate embedded in the BootROM
- Firmware images stored in signed IMG3 containers
- RSA signatures checked before moving on to the next stage

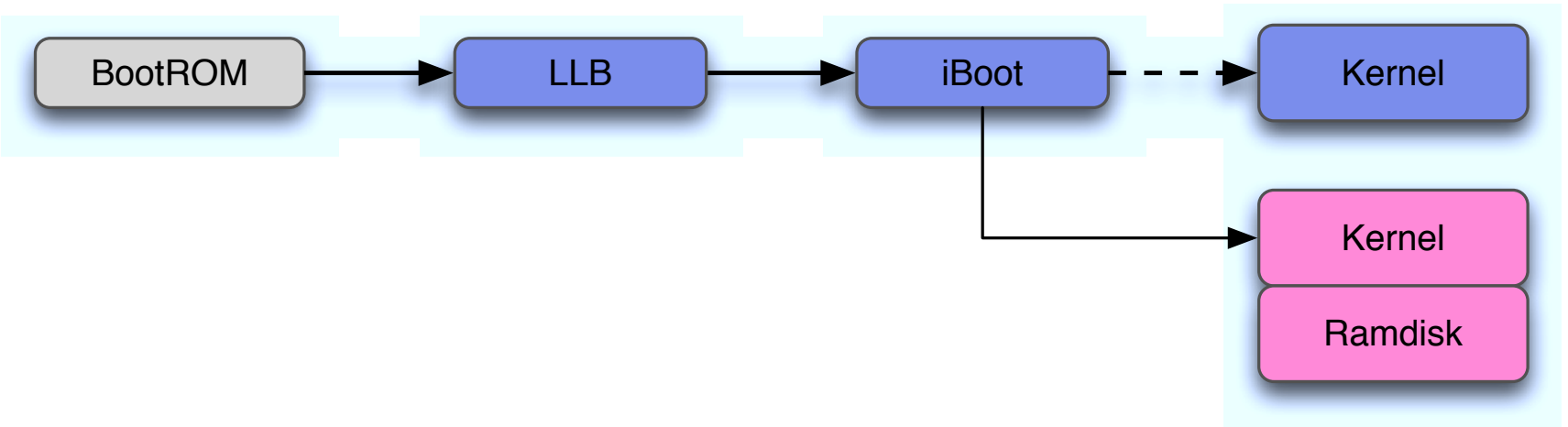
USB interfaces

- 2 interfaces available before iOS startup
  - DFU mode (BootROM)
  - Recovery mode (iBoot)
- Used to bootstrap ramdisk with flashing tool (update/restore)

Trusted boot - DFU mode



Trusted boot - recovery mode



Trusted boot - iOS startup

Important processes

- First userland process: *Launchd*
  - Starts daemons
  - Register IPC services
- *CommCenter*: interface with the baseband (AT commands)
- *Lockdown*: iTunes USB entry point
- *SpringBoard*: GUI

Code signing

Applications binaries

- MACH-O format
- Code directory structure with SHA-1 hashes of memory pages
- Code directory is signed
- PKCS#7 signature embedded for AppStore binaries
- For system binaries, code directory hashes are already cached in kernel

Entitlements

- Describes permissions for the application
  - Allow debugger to attach
  - Keychain access group
  - Sandbox profile
- XML document embedded in binary (signed)

Sandboxing & exploit mitigations

Sandboxing

- Seatbelt kernel extension
- Mandatory Access Control on files, sockets, etc.
- Predefined profiles with rules
- Mainly used to restrict filesystem access & isolate applications

Exploit mitigations

- Applications run with standard user account (`mobile`)
- Non-executable stack & heap
- `W^X` policy enforced on code pages
- No ASLR → Return-oriented programming (ROP) is possible

Keychain

Secure storage

- SQLite database
- Tables for passwords, certificates, keys
  - Email accounts, VPN certificates & keys, SIM card pincode, Wi-Fi keys, etc.
- Table columns: account, data, access group
- Data column is encrypted

Access control

- Exposed to applications through an IPC API
- Security Server translates IPC calls into SQL queries
- Restrict queries with caller access group
- System applications share the “apple” access group

Keychain encryption

iOS $\leq$ 3

- Data encrypted using AES with key 0x835 (unique for each device)
- Random initialization vector

iOS 4

- Random encryption key for each item
- Items have a new accessibility attribute (protection class)
  - always, after first unlock, when unlocked (screenlock)
- Item key is wrapped with the protection class key (master key)
- Part of a new feature called data protection

Data protection

Description

- Used to protect keychain items and data files
- Protection classes keys are grouped in keybags
- Keystore kernel extension manages keybags
- Unlocking the screenlock → class keys are unwrapped
- AES key wrap algorithm (RFC 3394)

Passcode derivation

- AES wrap key encryption key is derivated from user passcode
- Derivation involves use of the on-device UID AES key
- Makes passcode bruteforce impractical

Attack surface

Attack surface

- Bootloaders USB communication: DFU, recovery mode, restore process
- Bootloaders transitions
- iTunes services: Lockdown, AFC, BackupAgent, Sync, etc.
- Network: cellular, Wi-Fi
- Applications: Web browser, file formats, IPCs
- Kernel: BSD API, IOKit interfaces

Plan

Bootloader vulnerabilities - objectives

Objectives

- Extract data from the phone with physical access
  - Call logs, contacts, SMS messages, etc.
- Decrypt ciphered data if possible (keychain)
  - Passwords, certificates/keys, passcode, etc.

How?

- Bootloaders USB interfaces only accept signed binary images
- Need a vulnerability to execute arbitrary code
- Many vulnerabilities have been found in DFU mode and iBoot
  - Possible to use vulnerabilities from jailbreak tools

Blackra1n (geohot - October 2009)

Vulnerability

- Bad handling of USB control messages in iBoot

Exploit

- Send: `usb_control_msg(0x21, 2)`
- Result: `memcpy(0x0, LOAD_ADDR, 0x2000)`
- `LOAD_ADDR` contains USB received data
- Interrupt handler was overwritten so it executes shortly after
- Patches signature checks in iBoot and kernel

Limera1n/greenpois0n (geohot/comex - October 2010)

Vulnerability

- Bad handling of USB control messages in DFU mode
- Heap overflow

Exploit

- Send a specially crafted USB control msg
- Result: code execution thanks to a heap overflow
- Load original bootloaders and patch signature checks
- Do the same for the kernel

Forensics ramdisk

Realization

- Use exploit to disable signature checks
  - Blackra1n iBoot exploit (firmware $\leq$ 3.1.2)
  - Pwnage 2 BootROM exploit on older devices (iPhone $\leq$ 3G)
  - Limera1n/greenpoison BootROM exploit on newer devices (iPhone 4)
- Load our own ramdisk with extraction tool (same as Jonathan Zdziarski)
- Retrieve data over USB

Forensics ramdisk

Results

- Leave no trace (except the phone was rebooted)
- Took only a few minutes
- Allows extraction of SMS, contacts, etc.
- Extraction of keychain data
  - Possible on iOS < 4
  - Need passcode bruteforce on iOS 4
    - Always accessible items can be retreived

Remember

Demo

Plan

Browser vulnerabilities - objectives

Objectives

- Install a rootkit on a device
- Do it remotely
- Extract data from the device
- Keep control of the device

How?

- Need a remote exploit
- Star allows this

Star

Description

- Released by comex in August 2010
- Use the MobileSafari browser (jailbreakme.com)
- Userland jailbreak
- Remote code execution
- 1-week Apple response (to prevent misuse)

3 vulnerabilities

- PDF CFF fonts vulnerability (ROP)
- IOSurface kernel vulnerability
- Incomplete codesign: launchd interposition

Star - PDF CFF fonts vulnerability

Vulnerability

- Freetype font parser stack overflow
- Can be triggered by opening a PDF file

Exploit

- ROP payload exploits IOSurface kernel vulnerability
  → Code signing checks are now disabled
- Write `installui.dylib` in `/tmp`, load it and call `iui_go()`
- Repair stack and resume thread
- Display progress bar, download and install Cydia

Star - IOSurface vulnerability

Vulnerability

- IOSurface: pixel buffer managed by the kernel
- Integer overflow on width and height properties

Exploit

- Patch signature checks and sandboxing restrictions
- Patch `suser` function to allow MobileSafari to get root access

Star - incomplete codesign - launchd interposition

Launchd gmalloc

- Debug mechanism in Launchd
- At startup, Launchd checks if `/var/db/.launchd_use_gmalloc` exists
- If so, it loads "guard malloc dynamic library" (/usr/lib/libgmalloc.dylib)
  → Can be used maliciously to persist to a reboot

Star - incomplete codesign - launchd interposition

Exploit

- Use .dylib interposition to redirect execution through existing code fragments
- Make a stack pivot to have SP pointing to the .dylib data section
- Execute a ROP payload from now on
  → Runs as root in launchd and exploits IOSurface kernel vulnerability
- Restart launchd without .dylib once the kernel is patched

Vulnerability

- Dynamic library interposition allows modification of imported symbols
- Signatures only required on code pages
- NOT on dynamic library interposition

Malicious PDF

Realization

- Idea: modify Star payload
- Extract font stream (payload) from the original exploit
- Create a custom `installui.dylib` with a `iui_go()` function
- Replace `installui.dylib` in extracted payload
- Inject modified payload in any PDF file with origami (thanks Guillaume :-)
- Send the PDF to your victim

Malicious PDF

Rootkit

- Victim opens the PDF file
- `iui_go()` → download and run rootkit binary
- Poll orders and send data back to command & control server
- For now, only get contacts and SMS messages
- Can also steal keychain data when the phone is unlocked with standard API

Remember

Demo

Conclusion

Bootloader exploits

- Can be used for targeted physical attacks
- Data extraction only takes a few minutes
- BootROM vulnerabilities cannot be patched (Pwnage, limera1n/greenpois0n)
- New data protection feature helps protect data with passcode

Browser exploits

- Star remote exploit is one of a kind
- Made possible due to lack of ASLR
- Hopefully no serious malware on the iPhone yet

Thanks for your attention

References

- 25C3: Hacking the iPhone, 2008
- The iPhone wiki, http://www.theiphonewiki.com
- Ralf-Philipp Weinmann & Vincenzo Iozzo own the iPhone at PWN2OWN, 2010, http://blog.zynamics.com/2010/03/24/ralf-philipp-weinmann-vincenzo-iozzo-own-the-iphone-at-pwn2own/
- Post Exploitation Bliss: Meterpreter for iPhone, Charlie Miller and Vincenzo Iozzo, 2009, http://www.blackhat.com/presentations/bh-usa-09/IOZZO/BHUSA09-Iozzo-iPhoneMeterpreter-SLIDES.pdf
- iPhone privacy, Nicolas Seriot, 2010, http://seriot.ch/blog.php?article=20091203
- Apple WWDC 2010, Session 209 - Securing Application Data
- Star jailbreak, Comex, 2010, http://www.jailbreakme.com
- Star source code, http://www.github.com/comex/star
- Origami, http://esec-lab.sogeti.com/origami