# iOS JAILBREAKING
# Lecture 21

## COMPSCI 702
## Security for Smart-Devices

**Nalin** Asanka Gamagedara Arachchilage

Slides from Muhammad **Rizwan** Asghar

April 29, 2021

# JAILBREAKING

- Removing the restrictions, put by Apple, on iOS devices by manipulating the software stack!
- It provides full access to the OS and filesystem
- Using installers like Cydia, it enables installation of apps and themes not approved by Apple
- Jailbreaking is legal in the United States
  - Digital Millennium Copyright Act (DMCA 2010)
  - It can void the warranty

# WHY JAILBREAKING

- People jailbreak their iOS devices for many reasons, e.g.,
  - An open platform for developing software (before xcode 7)
  - Getting full control over the device
  - To bypass cellular carrier locks
  - To pirate iPhone apps
  - To evaluate the security and discover vulnerabilities
  - To do some fraud

# KINDS OF JAILBREAKS

- Depending on the vulnerabilities used, there are two kinds of jailbreaks with respect to jailbreak persistence
  - Tethered jailbreaks
  - Untethered jailbreaks

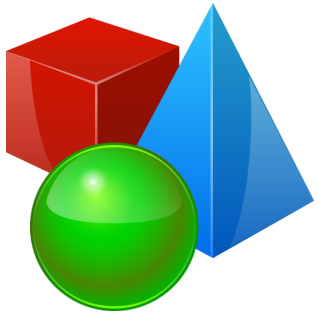
# TETHERED JAILBREAK

- Not permanent
- Every reboot requires the jailbreak to be installed again
  - But files installed by a previous jailbreak will still be there
- The device has to be connected to a computer via a USB cable
  - Sometimes revisiting a website or running an app can reinstall the jailbreak

# UNTETHERED JAILBREAK


- Permanent effect
  - Can reboot the device and switch it on and off
- Harder to do because it needs to make permanent changes to the boot sequence
- Used to be easy when there was a vulnerability in the bootrom
- Now requires multiple exploits
  - Start with a tethered jailbreak
    - Unsigned code needs to be executed
  - Find a way to install additional exploits on the root filesystem
    - Privileges need to be escalated to patch the kernel

# EXPLOIT TYPES

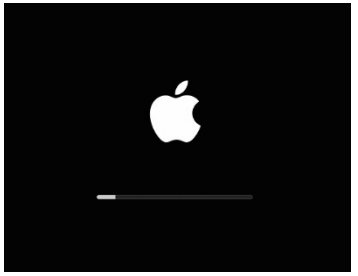


- The location of a vulnerability impacts the access level
- Vary from the device hardware to its software
- There are three levels of exploits
  - Bootrom level
  - iBoot level
  - Userland level

# BOOTROM LEVEL

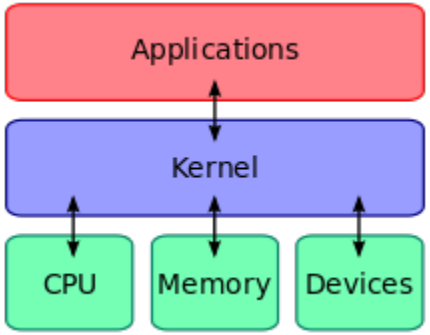

- Vulnerabilities inside the hardware
- The most powerful vulnerabilities from the point of view of a jailbreaker
  - Enables jailbreakers to modify the whole bootchain
- Cannot be fixed by any software update
- Can be fixed only within the next hardware revision
- Famous exploits, e.g., used to run unsigned code
  - SHAtter
  - Limera1n
- Effected iPod 3G, iPhone 3GS and all A4-based devices

# iBOOT LEVEL



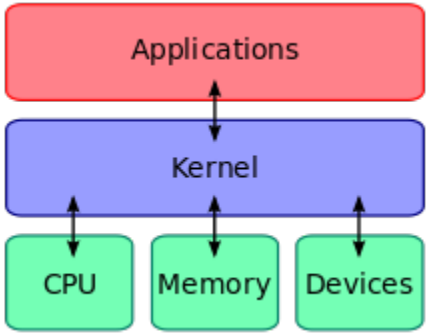
- Vulnerabilities inside iBoot
- As powerful as that of the bootrom in terms of features
  - Still early in the boot process
- Since iBoot is not baked into the hardware, they can be fixed by a software upgrade
- Famous exploit
  - iH8sn0w – A5 iBoot exploit
- iOS 7 untethered jailbreak affected iPhone 4S, iPad Mini and other devices

# USERLAND LEVEL




- Vulnerabilities in userland processes
- Less powerful
- Easier to fix
- These processes can run with any permissions
  - Root user – system processes
  - Mobile user – user applications
- In both cases, at least two exploits required to jailbreak the device
  - First to achieve arbitrary code execution
  - Second to escalate privileges

# USERLAND LEVEL CONT.




- Famous exploit
  - JBME: http://jailbreakme.com
- JBME 1 targeted earlier versions
- JBME 2 targeted iOS 4.0.1
- JBME 3 targeted iOS 4.3.3

# JAILBREAKME.COM: STAR

- Go to the website and jailbreak your phone, without attaching it to a computer or rebooting
- Original version worked on iPhone
- Version 2 more complicated
  - Targeted iOS 4.0.1
  - Other security techniques but no ASLR
- Stack overflow when MobileSafari handled a particular font
  - The error was in the FreeType parser when rendering PDFs
- This allowed the exploit to mount ROP within MobileSafari

# JAILBREAKME.COM: STAR CONT.

- This sophisticated payload proceeded to exploit another vulnerability
  - To increase its level of access
- The second vulnerability was an integer overflow in an IOSurface property
  - IOSurface is a private framework to share graphics between processes
  - The second attack allowed code execution inside the kernel
- From the kernel, disabled code signing
- The ROP downloaded an unsigned library that jailbroke the device

# JAILBREAKME.COM: SAFFRON

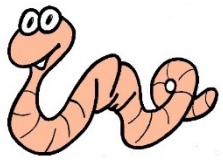
- Apple patched issues quickly
- Version 3 – worked on early ASLR versions up to 4.3.3
- The attacker defeated ASLR
- The exploit allowed the attacker to read and write memory
- Then, it could read the values of nearby pointers and work out where it was in memory
- Finally, it corrupted memory and got control of the process by writing to memory

# LIMERA1N

- limera1n bootrom vulnerability – before iPhone 4S (A5 devices)

- Heap-based buffer overflow in the USB Data Firmware Upgrade (DFU) stack of the bootrom
  - Connect the machine to a computer via USB
  - Force into DFU mode
  - Use the overflow to run the code that can patch the signature verification code
    - Boot of a ramdisk
    - Run the patched version of the low-level bootloader (iBoot) and kernel
    - These patches allow execution of unsigned code
  - This is only a tethered jailbreak but it can be used to create an untethered one

# JAILBORKEN DEVICES HIT WORMS



- Jailbreaking essentially reduces iOS security and makes devices vulnerable to different worms

- E.g., Ikee worm
  - Many jailbroken phones had an SSH server installed
  - Did not change the default root password “alpine”
  - SSH server was not sandboxed
  - What did the worm do?
    - Originally changed the wallpaper
    - Later, it was used to lock the phone, steal contents and create botnets

# PROBLEMS WITH JAILBROKEN PHONES

- Disables almost all security features
  - Code signing – unsigned code can run
  - Attack surface is increased
  - Shell and other utilities added
  - Can install root privileged code
  - Unsigned apps are not sandboxed
- Some jailbreaks leave code signing running
  - Allow self-signed code

# PROBLEMS WITH JAILBROKEN PHONES CONT.

- Easy to dump unencrypted versions of apps
  - Files are encrypted but are unencrypted when running
  - Load a program and then use a debugger to dump the program back unencrypted
  - All information, e.g., variables, methods and classes can be inspected or modified
- Protections turned off when jailbreaking
  - The code which ensures that pages cannot be executable and writable
  - Platform apps may run outside their sandboxes
    - This is amazingly dangerous, especially for MobileSafari and MobileMail
- Basically, jailbreaking breaks security architecture of the device

# SUMMARY

- Some people want to jailbreak their devices because of the restrictions Apple places
- But jailbreaking by definition has to stop some (if not most) of the iOS security measures from functioning
- Jailbreaking essentially reduces iOS security to the level of Android
- It is an arms race
  - It is getting harder all the time but iOS versions ultimately get jailbroken, albeit it might take some months for a public jailbreak

# RESOURCES

- **iOS Hacker's Handbook**
  Charlie Miller, Dionysus Blazarkis, Dino Dai Zovi,
  Stefan Esser,Vincenzo Iozzo, Ralf-Philipp Weinmann
  John Wiley & Sons, Inc., 2012

- **SHAttered Dreams: Adventures in BootROM Land**
  http://conference.hitb.org/hitbsecconf2013kul/materials/D2T1%20-%20Joshua%20'p0sixninja'%20Hill%20-%20SHAttered%20Dreams.pdf

- **limera1n Exploit**
  http://theiphonewiki.com/wiki/Limera1n_Exploit

# RESOURCES (2)

- **iH8sn0w**
  https://github.com/iH8sn0w
- **ikee worm**
  http://letsunlockiphone.guru/ios-viruses-iphone-ikee-b-worm/
- **Syringe**
  https://github.com/Chronic-Dev/syringe
- **Jailbreak**
  http://theiphonewiki.com/wiki/Jailbreak

# ACKNOWLEDGEMENT


- Some slides are based on the presentation shared by Robert Sheehan, thanks to him!

# Questions?

# Thanks for your attention!