Strategic Analysis of the iOS Jailbreak Development Community

Dino Dai Zovi
@dinodaizovi / ddz@theta44.org

Co-Founder/CTO, Trail of Bits, Inc.
Hacker-in-Residence, NYU:Poly

Why?

- Jailbreaks and attacks share many of the same underlying offensive technologies
  - Vulnerabilities, exploits, evading and disabling security mechanisms, “rootkits”
- Jailbreaker communities are more observable than attacker communities
  - The jailbreak communities develop and release jailbreaks in full public view
  - Malicious attackers develop and “release” their attacks as quietly as possible

Jailbreaks vs. Attacks

- Injection vector and an "untether"
- Allow OS "mods", run homebrew apps
- User-visible
- Developed in the open
- Released to much fanfare and press

- Remote exploits and rootkits
- Modifies operating system to hide, runs backdoors
- Invisible to user
- Developed in secret
- "Released" as quietly as possible

Thesis

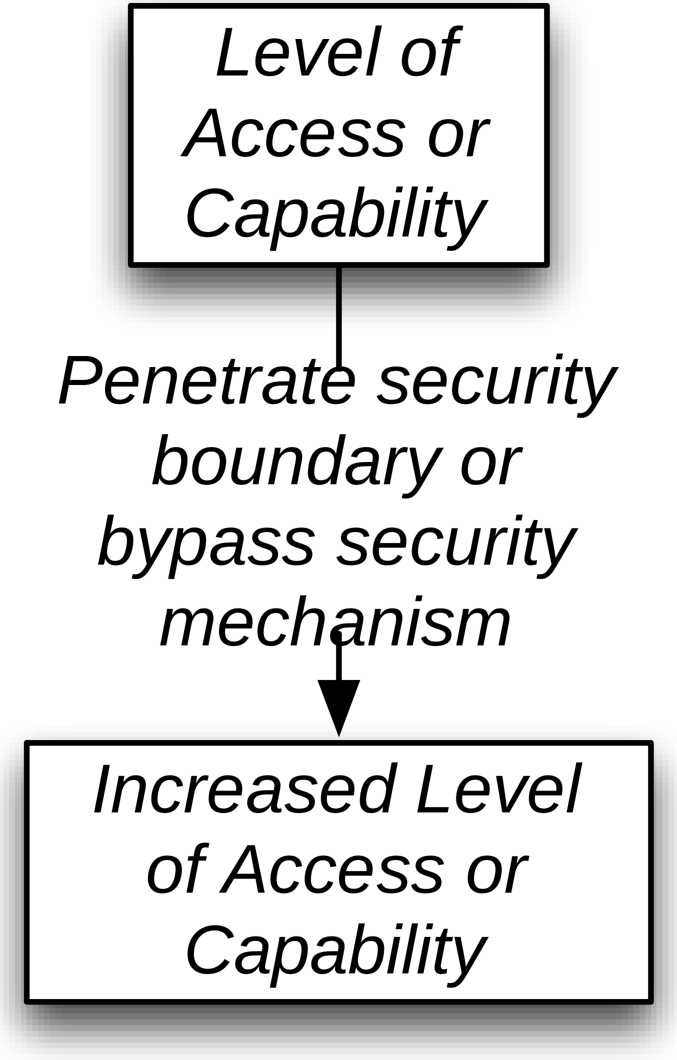
- Jailbreakers' public research, tools, and exploits reduce costs for malicious attackers
- Jailbreakers' and attackers' incentives differ
- Rather than fight jailbreakers, channel them
  - Jailbreaking of any sufficiently popular device is inevitable
  - Minimize the reusability of their research, tools, and exploits by malicious attackers

Analysis Methodology

- Identify security barriers and mechanisms in iOS that must be defeated in order to jailbreak device
  - Construct a model attack graph documenting hypothetical exploits necessary to penetrate defenses
- Analyze exploits in released jailbreaks to identify their *actual* path through the *hypothetical* model
- Path taken through attack graph indicates relative preference among alternatives

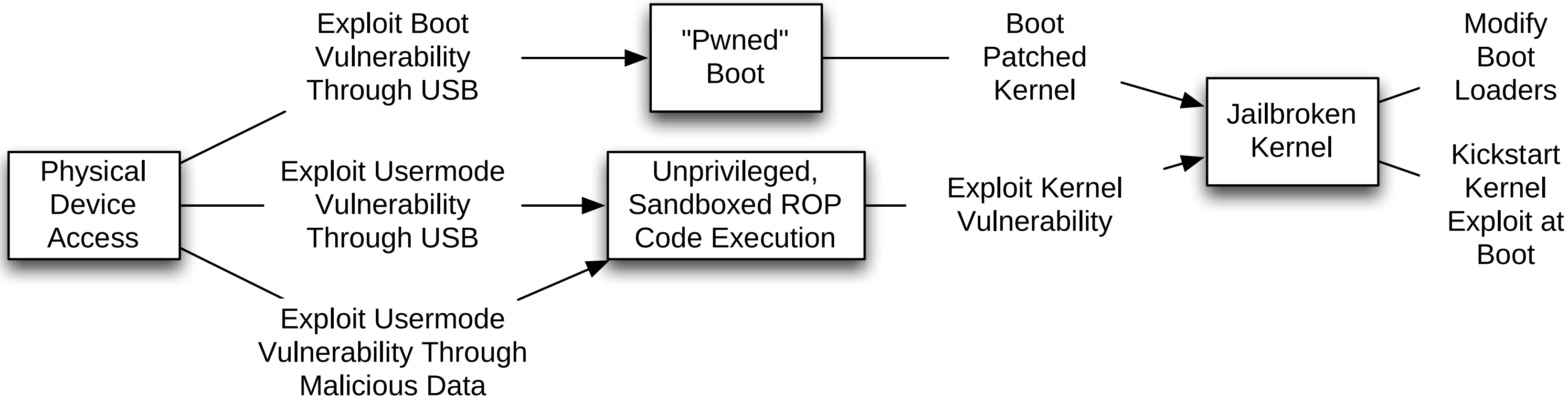
Attack Graphs

- Depict security defenses that must be defeated
- Show alternative paths from initial level of access or capability to goal
- Attacks are now more complex than a single remote root exploit



Level of Access or Capability
Penetrate security boundary or bypass security mechanism
Increased Level of Access or Capability

Jailbreaking Attack Graph



Physical Device Access

Exploit Boot Vulnerability Through USB

Exploit Usermode Vulnerability Through USB

Exploit Usermode Vulnerability Through Malicious Data

"Pwned" Boot

Unprivileged, Sandboxed ROP Code Execution

Boot Patched Kernel

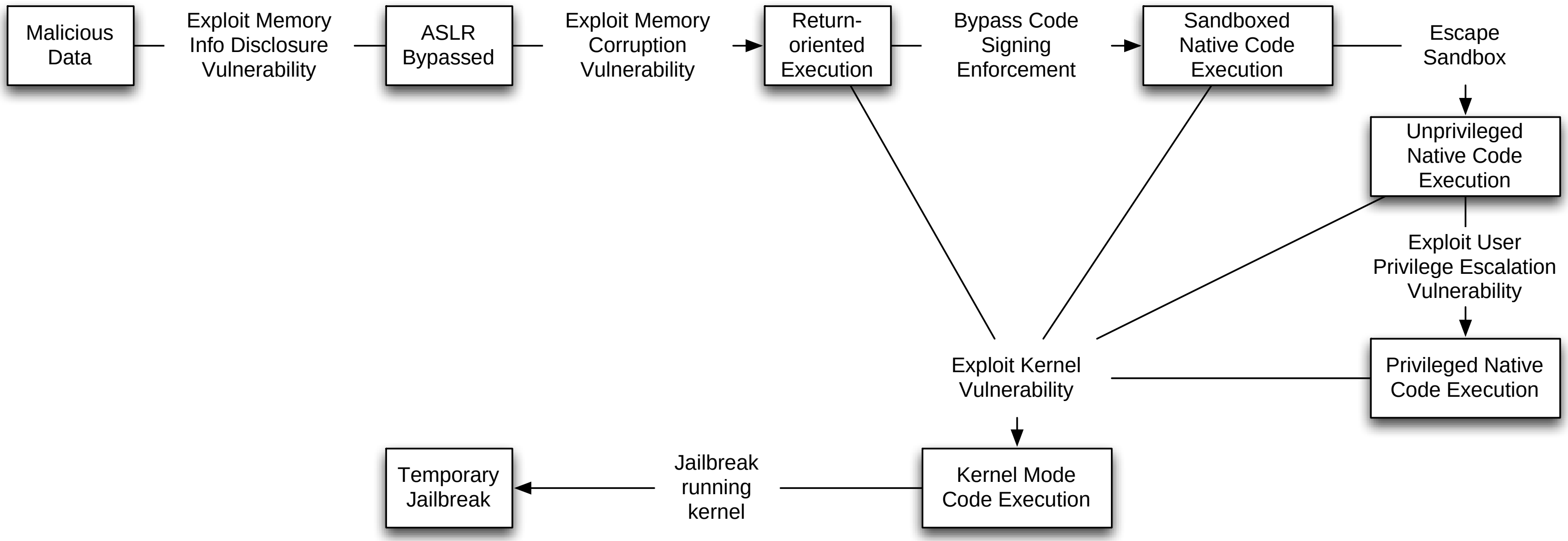
Exploit Kernel Vulnerability

Jailbroken Kernel

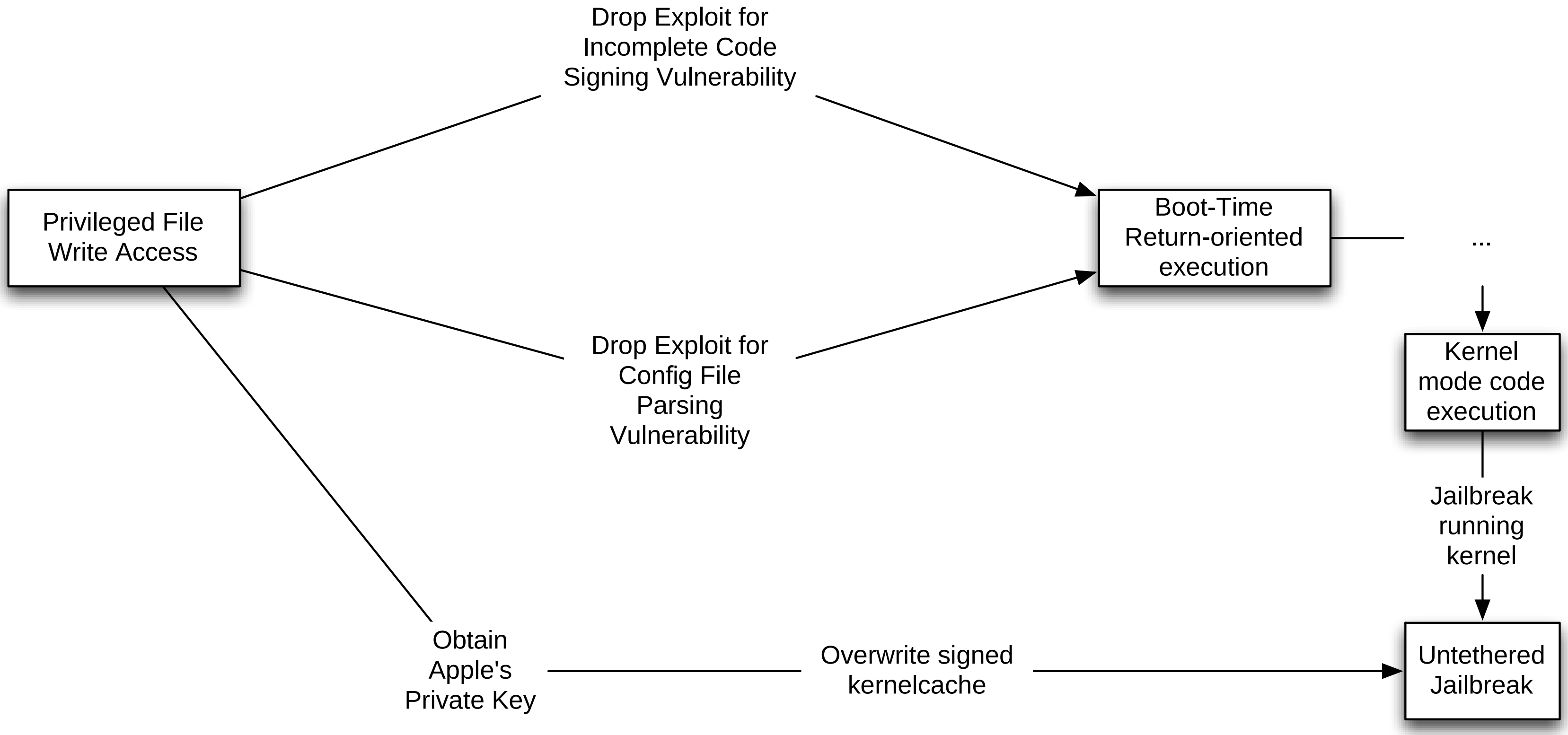
Modify Boot Loaders

Kickstart Kernel Exploit at Boot

iOS Userland Injection Attack Graph


Malicious Data
Exploit Memory Info Disclosure Vulnerability
ASLR Bypassed
Exploit Memory Corruption Vulnerability
Return-oriented Execution
Bypass Code Signing Enforcement
Sandboxed Native Code Execution
Escape Sandbox
Unprivileged Native Code Execution
Exploit User Privilege Escalation Vulnerability
Privileged Native Code Execution
Exploit Kernel Vulnerability
Kernel Mode Code Execution
Jailbreak running kernel
Temporary Jailbreak

iOS Userland Untether Attack Graph



Privileged File Write Access

Drop Exploit for Incomplete Code Signing Vulnerability

Drop Exploit for Config File Parsing Vulnerability

Boot-Time Return-oriented execution

...

Kernel mode code execution

Jailbreak running kernel

Obtain Apple's Private Key

Overwrite signed kernelcache

Untethered Jailbreak

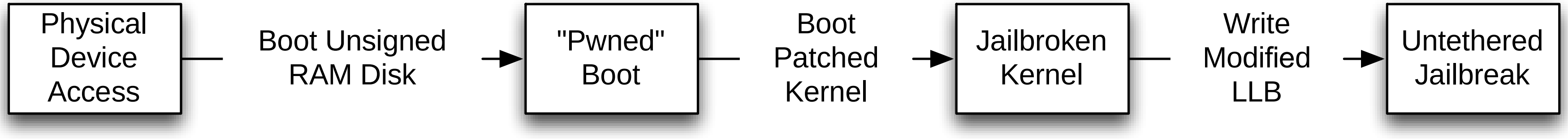
Types of iOS Jailbreaks

- Boot injection + untether
  - Pwnage, Pwnage 2.0
- Boot injection + Userland untethers
  - 24kpwn/Steaks4uce/Limera1n
  - ndrv_setspec(), HFS+ untether exploits
- Userland injection + untether
  - Spirit, Star, Saffron
  - A5 Corona (“Absinthe”)

Boot ROM Jailbreaks

- Boot ROM vulnerabilities cannot be patched in existing devices, are only fixed in new platforms (i.e. iPhone 4S vs. iPhone 4)
- Example: Pwnage 2.0
    - Boot ROM certificate stack buffer overflow
    - Affects iPhone, iPod Touch, iPhone 3G (old Boot ROM) *forever*
    - Patched in Boot ROM for iPod Touch 2G and later devices

Pwnage



Physical Device Access

Boot Unsigned RAM Disk

"Pwned" Boot

Boot Patched Kernel

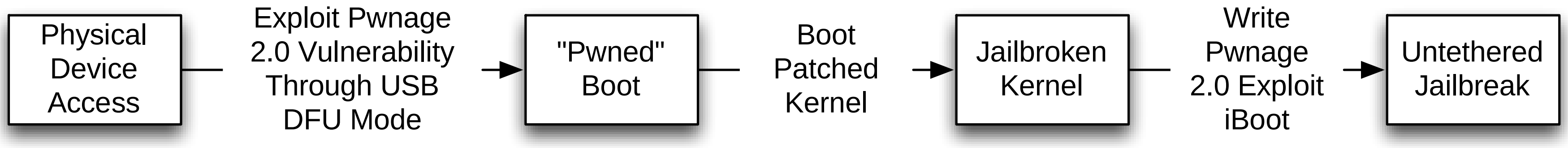
Jailbroken Kernel

Write Modified LLB

Untethered Jailbreak

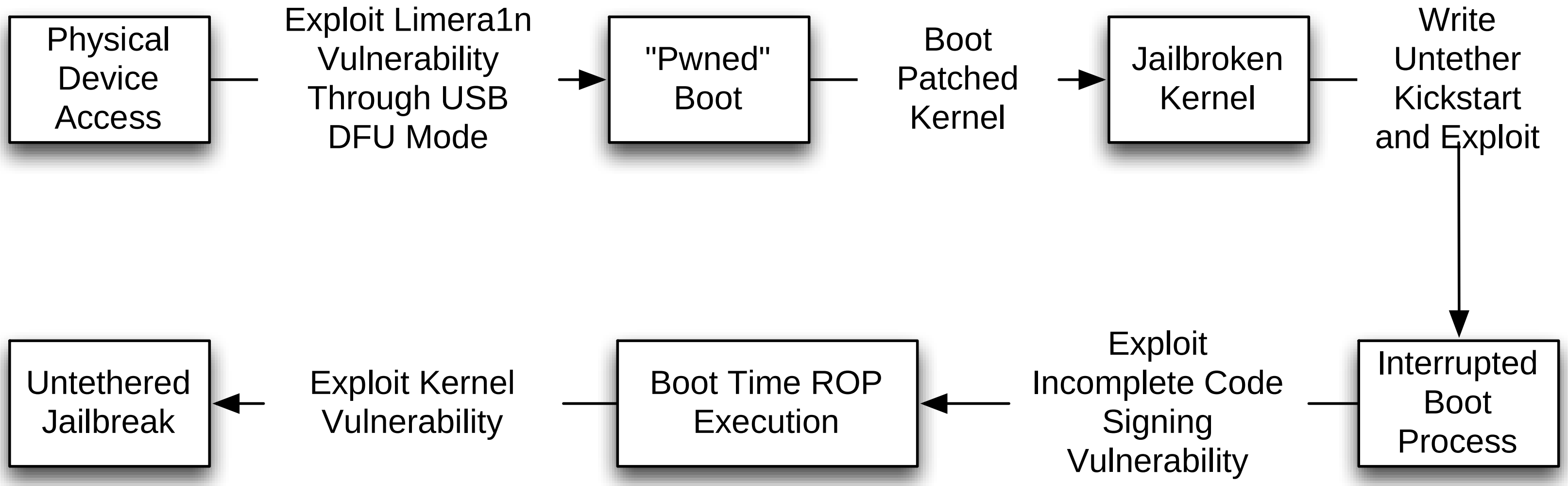
- Unsigned ramdisk vulnerability
  - pmd, vmd boot arguments allowed booting an unsigned ramdisk
- LLB boot loader was not signed, only encrypted
  - Booting an ramdisk with a patched driver allowed writing custom LLB

Pwnage 2.0



- Boot ROM DER certificate stack buffer overflow
  - No length check on certificate signature
- Affects iPhone, iPod Touch, iPhone 3G (old Boot ROM) *forever*
- Patched in Boot ROM for iPod Touch 2G and later devices

Limera1n


Physical Device Access
Exploit Limera1n Vulnerability Through USB DFU Mode
"Pwned" Boot
Boot Patched Kernel
Jailbroken Kernel
Write Untether Kickstart and Exploit
Interrupted Boot Process
Exploit Incomplete Code Signing Vulnerability
Boot Time ROP Execution
Exploit Kernel Vulnerability
Untethered Jailbreak

- Affects A4 and earlier iOS devices
  - iPhone 3GS – iPhone 4, iPad, Apple TV 2G
- Boot ROM DFU mode heap buffer overflow
  - Provides injection vector, but requires separate untether exploits

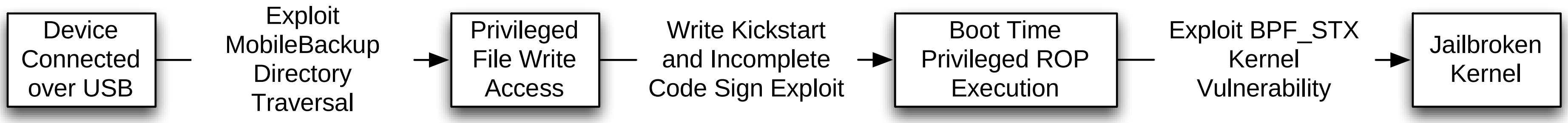
The SHAtter Incident

- Chronic Dev Team announces that GreenPois0n jailbreak tool with the SHAtter Boot ROM exploit will be released on 10/10/10 at 10:10:10
- GeoHotz releases Limera1n
- Chronic Dev Team replaces SHAtter exploit with Limera1n in GreenPois0n
  - Preserving SHAtter for future jailbreaks
  - SHAtter fixed by A5 Boot ROM

Userland Jailbreaks

- As Boot ROM vulnerabilities became rarer and/or non-existent, jailbreakers had to rely on fully userland jailbreaks
- Userland jailbreaks require more vulnerabilities that are patched quickly
  - Low-value vulnerabilities with low longevity
  - Absinthe 2.0 uses seven vulnerabilities and five exploits, any of which can be patched to break the jailbreak

“Spirit” Jailbreak

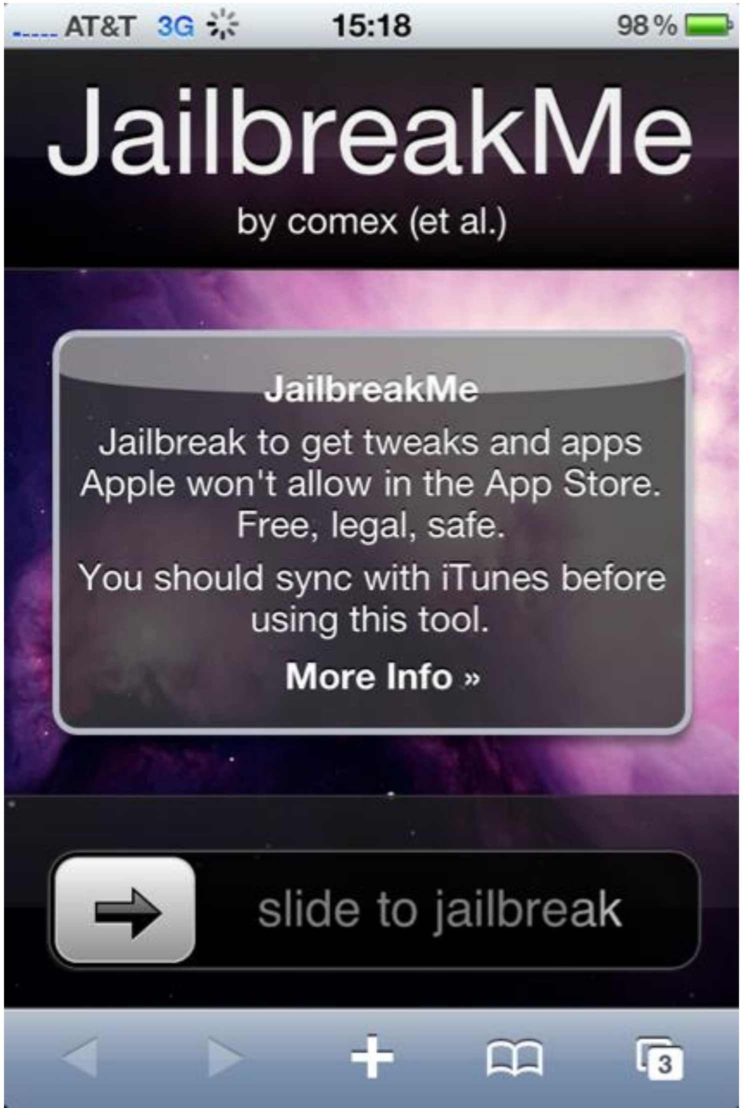


Device Connected over USB → Exploit MobileBackup Directory Traversal → Privileged File Write Access → Write Kickstart and Incomplete Code Sign Exploit → Boot Time Privileged ROP Execution → Exploit BPF_STX Kernel Vulnerability → Jailbroken Kernel

- Comex’ first pure userland jailbreak
- MobileBackup directory traversal plants untethering kickstarter and exploit on data partition
  - Takes advantage of launchd overrides and incomplete code signing
- Untether exploits kernel at boot to preserve jailbroken-ness

JailbreakMe 2.0 "Star"

- Web-based untethered userland jailbreak for iOS <= 4.0.1
- Timeline
  - 8/1/2010: Released
  - 8/11/2010: Patched by iOS 3.2.2 and 4.0.2



JailbreakMe

by comex (et al.)

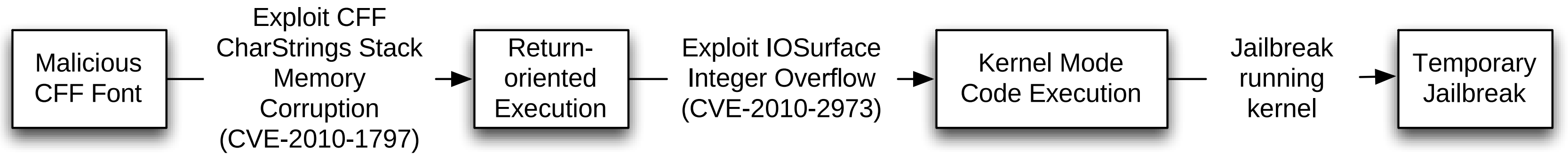
JailbreakMe

Jailbreak to get tweaks and apps Apple won't allow in the App Store. Free, legal, safe.

You should sync with iTunes before using this tool.

More Info »

“Star” Jailbreak Injection Vector


Malicious CFF Font
Exploit CFF CharStrings Stack Memory Corruption (CVE-2010-1797)
Return-oriented Execution
Exploit IOSurface Integer Overflow (CVE-2010-2973)
Kernel Mode Code Execution
Jailbreak running kernel
Temporary Jailbreak

- CharString font program copies return-oriented payload onto stack
- ROP stage1 payload exploits IOSurface kernel vulnerability to map kernel memory into user space
- ROP stage2 payload temporarily jailbreaks kernel and loads a dynamic library to complete installation of jailbreak

Type 2 CharStrings

- CharStrings are interpreted font programs that hint or instruct the rasterizer on how to draw glyphs proportionally
- CharString programs are interpreted by a stack-based virtual machine
- CVE-2010-1797
  - Argument stack depth limit is not enforced
  - Operators that push onto stack can exceed depth limit and write memory out-of-bounds, corrupting CPU stack

“Star” CharString

```
0x0 0x0 0x0 0x3404f931 0x0 0x0 0x0
0x301715bf 0x9000000 0x109748 0x3 0x1012
0x300e18ad 0x0 0x0 0x0 0x30014ad9
0xfffff998 0x33c43ff1 0x0 0x0 0x30009e85
0x9109744 0x0 0x301e235f 0x4d8 0x0
0x3002e135 0x3404ccb9 0x0 0x3008c6dd
0x9100000 0x9744 0x0 0x300e18ad 0x0 0x0
0x9100000 0x30006d49 0x0 0x0 0xf00df00d
0xc00000 0xc00000 0xc00000
random random or index (x 20 …)
random index (x 170 …)
index drop (x 42 …)
0x0 endchar
```

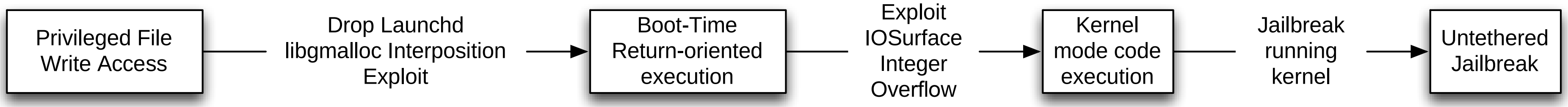
IOSurface

- IOSurface framework and kernel extension share graphic surfaces (pixel buffers) between processes
- CVE 2010-2973
  - In creating an IOSurface, integer overflow in calculation of surface size from allocation size, width, height, and bytes per pixel results in mapping kernel memory into user process
  - Kernel payload is injected by writing it directly into mapped kernel memory

“Star” IOSurface Properties

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>IOSurfaceAllocSize</key>
        <integer>242960</integer>
        <key>IOSurfaceBufferTileMode</key>
        <false/>
        <key>IOSurfaceBytesPerElement</key>
        <integer>4</integer>
        <key>IOSurfaceBytesPerRow</key>
        <integer>3049657408</integer>
        <key>IOSurfaceHeight</key>
        <integer>3221241675</integer>
        <key>IOSurfaceIsGlobal</key>
        <true/>
        <key>IOSurfaceMemoryRegion</key>
        <string>PurpleGfxMem</string>
        <key>IOSurfacePixelFormat</key>
        <integer>1095911234</integer>
        <key>IOSurfaceWidth</key>
        <integer>3983639824</integer>
</dict>
</plist>
```

“Star” Jailbreak Untether

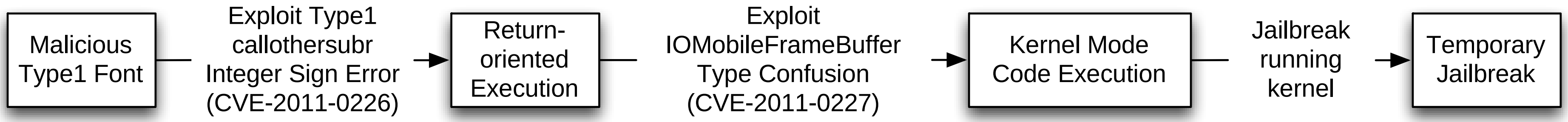

Privileged File Write Access
Drop Launchd libgmalloc Interposition Exploit
Boot-Time Return-oriented execution
Exploit IOSurface Integer Overflow
Kernel mode code execution
Jailbreak running kernel
Untethered Jailbreak

- Create /var/db/.launchd_use_gmalloc, causing launchd to load /usr/lib/libgmalloc.dylib
- Crafted libgmalloc.dylib has no executable segments (no code signature needed), uses dyld interposing to point exported functions to stack pivots
  - Pivot stack and execute ROP when called

JailbreakMe 3.0 "Saffron"

- Déjà Vu of "Star"
  - Malicious PDF file in web page exploits Type 1 font program interpreter memory corruption vulnerability (CVE-2011-0226)
  - ROP payload exploits kernel vulnerability in IOMobileFramebuffer (CVE-2011-0227)
- Timeline
  - July 2, 2011: Leaked
  - July 5, 2011: Released
  - July 15, 2011: Patched in iOS 4.3.4

“Saffron” Injection Vector


Malicious Type1 Font
Exploit Type1 callothersubr Integer Sign Error (CVE-2011-0226)
Return-oriented Execution
Exploit IOMobileFrameBuffer Type Confusion (CVE-2011-0227)
Kernel Mode Code Execution
Jailbreak running kernel
Temporary Jailbreak

- Type 1 font program gains programmatic read/write access to interpreter state structure
  - Reads pointer value, computes “slide”, adjusts ROP, and copies it onto stack
- Exploits IOMobileFramebuffer type confusion vulnerability in IOKit User Client accessible within MobileSafari sandbox

CVE-2011-0226

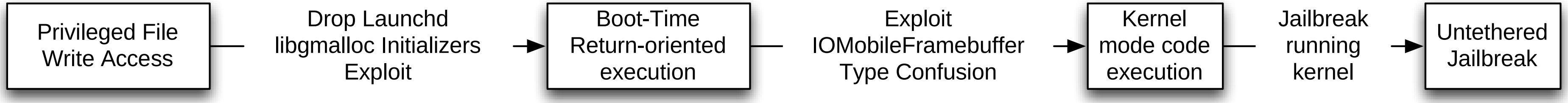
- Memory corruption vulnerability in FreeType interpretation of Type 1 CharStrings
- Argument count to `callothersubr` operation can be negative
  - Moves interpreter stack out of bounds
  - Stack operations now provide read and write access to font interpreter state structure (`T1_DecoderRec`)
  - "Weird machine" font program adjusts ROP for ASLR and transfers control to it

“Saffron” Type 1 Font Program

```
--------------------------------------------------------------------------------
File: at.bin                                 SHA1: 49b6ea93254f9767ad8d314dd77ecb6850f18412
--------------------------------------------------------------------------------
0x00000000  8e      push 0x3
0x00000001  8b      push 0x0
0x00000002  0c 21   op_setcurrentpoint
0x00000004  8e      push 0x3
0x00000005  0a      callsubr #03
0x00000006  fb ef   push 0xfea50000
0x00000008  b5      push 0x2a
0x00000009  0c 10   callothersubr #42 nargs=-347
0x0000000b  0c 10   callothersubr
0x0000000d  16      op_hmoveto
0x0000000e  16      op_hmoveto
0x0000000f  16      op_hmoveto
0x00000010  0c 21   op_setcurrentpoint
0x00000012  0c 02   op_hstem3
0x00000014  0c 02   op_hstem3
0x00000016  0c 02   op_hstem3
0x00000018  0c 02   op_hstem3
0x0000001a  0c 02   op_hstem3
[...]
```

http://esec-lab.sogeti.com/post/Analysis-of-the-jailbreakme-v3-font-exploit

“Saffron” Untether

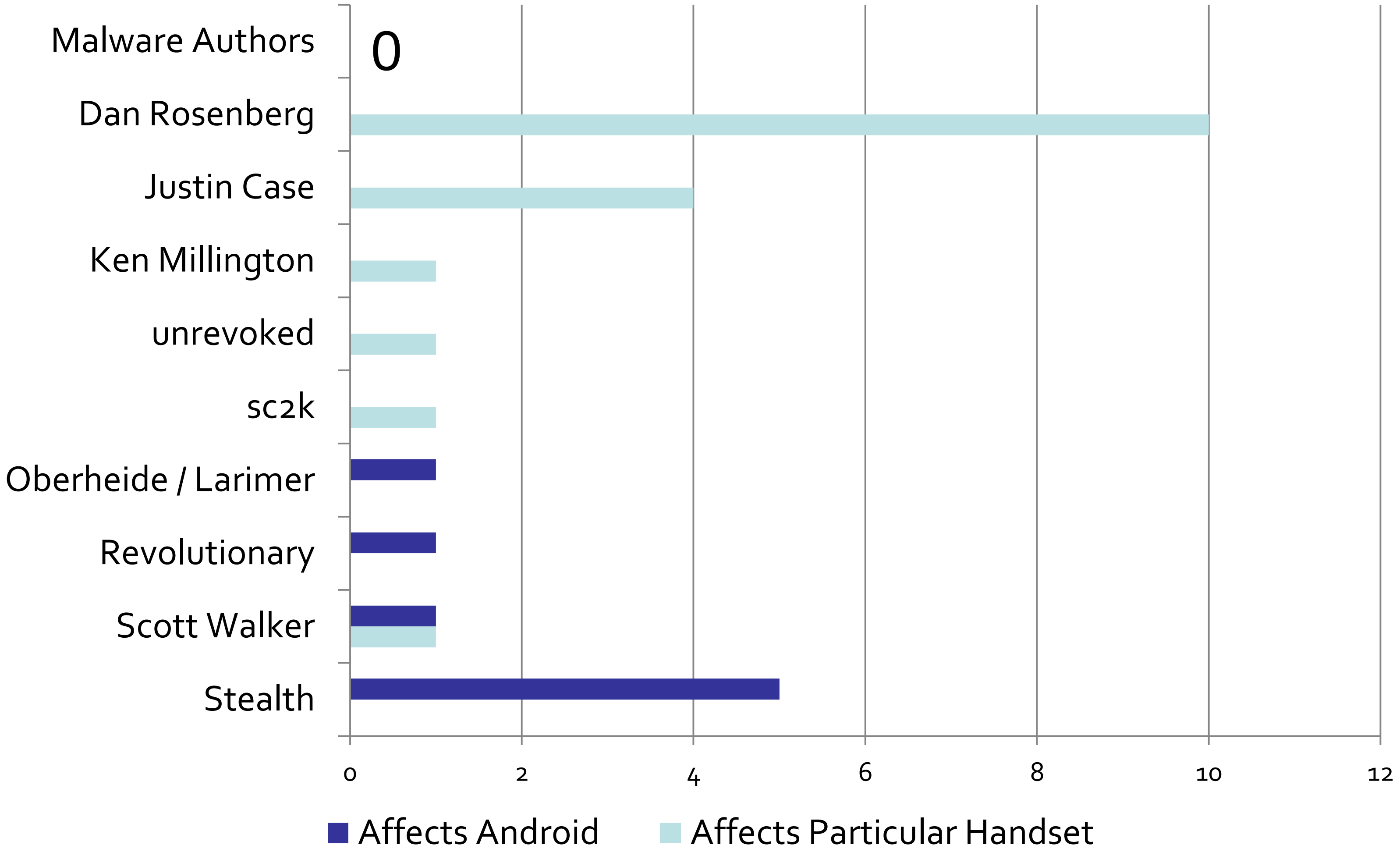

Privileged File Write Access
Drop Launchd libgmalloc Initializers Exploit
Boot-Time Return-oriented execution
Exploit IOMobileFramebuffer Type Confusion
Kernel mode code execution
Jailbreak running kernel
Untethered Jailbreak

- Create /var/db/.launchd_use_gmalloc, causing launchd to load libgmalloc.dylib
- libgmalloc.dylib has no text segments (no code signature needed), uses Mach-O relocations to adjust ROP addresses for ASLR
  - Initializers pivot stack and execute ROP payload to exploit IOMobileFramebuffer

What About Android?

- Most Android devices don’t prevent running apps not downloaded from Google Play, but do restrict root access
  - Jailbreaking simplifies to “rooting”, usually only requiring a single privilege escalation exploit
  - Many privilege escalation exploits may be easily bundled into Android apps
- See Dan Guido’s “Mobile Exploit Intelligence Project”

Privilege Escalations By Target

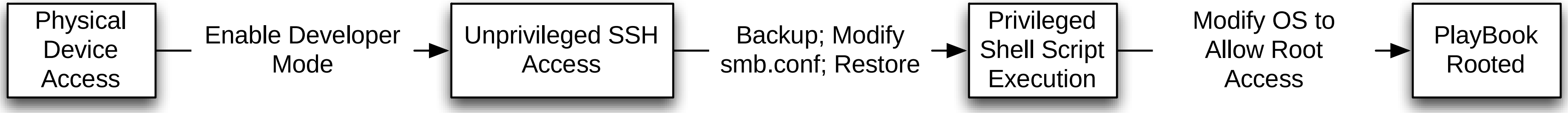


Guido, Dan and Arpaia, Mike. “Mobile Exploit Intelligence Project”

What About BlackBerry?

- No “jailbreaks” or “roots” for classic BlackBerry smartphone devices
  - Closed systems that are not well understood outside of RIM
  - High learning curve and barrier to entry
- Tablet OS / BBX / BlackBerry OS 10
  - Based on QNX Unix-like embedded system
  - Easier to understand and analyze
  - Jailbreaks and roots should be expected

DingleBerry


Physical Device Access
Enable Developer Mode
Unprivileged SSH Access
Backup; Modify smb.conf; Restore
Privileged Shell Script Execution
Modify OS to Allow Root Access
PlayBook Rooted

- Dingleberry smb.conf exploit
  - BlackBerry Desktop backups are not signed, can be modified and restored
  - Modification of smb.conf enables shell script execution as root
- Timeline
  - Released: 12/5/2011
  - Patched: 12/6/2011

DingleBerry smb.conf

```
[certs]
	comment = certificates
	path = /accounts/1000/certificates
	root preexec = /accounts/devuser/r.sh

[media]
	comment = media
	path = /accounts/1000/shared
	root preexec = /accounts/devuser/r.sh

[dtm]
	comment = DTM
	path = /accounts/1000/sys/dtm
	root preexec = /accounts/devuser/r.sh
```

Lessons

- Eradicating Boot ROM vulnerabilities pushed iOS jailbreak community towards pure userland and web-based jailbreaks
  - Provides all the exploits needed for drive-by rootkitting of iOS devices
- “Mobile EIP” showed that mobile malware authors did not develop a single privilege escalation exploit
  - All exploits in malicious apps were taken from the Android “rooting” community

Jailbreaks vs. App Priv Escs

- iOS untethers often exploit kernel vulnerabilities that require root privileges
  - Can't be used in a malicious app or after exploiting mail, browser, etc.
  - i.e. ndrv_setspec(), HFS+ vulnerabilities
- Dan Rosenberg makes a point to not release Android privilege escalations that can be used from within an app
- Patch App privilege escalations immediately, untethers with lower priority

Ideal

- Provide “developer mode” on all devices that disables secure boot and allows execution of “homebrew” software
  - Enables a visible indication to the user that the device is in developer mode that can’t be disabled from software, preventing surreptitious use
- Discourages the creation of jailbreaks utilizing vulnerabilities and exploits
- Attackers cannot “piggyback” on work of Jailbreak development teams

Questions?

References

1. http://www.theiphonewiki.com
2. http://www.ekoparty.org/archive/2010/ekoparty_2010-Monti-iphone_rootkit.pdf
3. http://esec-lab.sogeti.com/post/Analysis-of-the-jailbreakme-v3-font-exploit
4. http://www.trailofbits.com/resources/mobile_eip-04-19-2012.pdf
5. http://www.sourceconference.com/publications/bos12pubs/Lanier.Nell%20-%20Mobile.pdf