

➤ Information Security Decisions



Mobile Exploit Intelligence Project

Dan Guido, Trail of Bits
Mike Arpaia, iSEC Partners



What's This Talk About?

- Extension of recent talks on intel-driven defense
 - Exploit Intelligence Project, SOURCE Boston 2011
 - Attacker Math, SOURCE Boston 2011
 - Intel-driven: collect concrete data on actual attacks
 - Think like an attacker and model their choices
 - Use our model to predict future behavior (or lack of it)
 - Make better defense decisions with data
 - Focused on iOS and Android mobile OS's
 - We will be updating our analysis as time goes on
 - Latest version always at www.trailofbits.com
-

Thesis

- Mobile devices are loading up with data
 - E-mail, line of business apps, login credentials...
 - Lots of possibilities to compromise mobile devices
 - Insecure data storage, app-to-app, NFC, TEMPEST, ...
 - Very few vectors explored in actual attacks
 - Why is that? What motivates attackers? Isn't it easy?
 - What attacks do I need to defend against *now*?
 - Actual vs. Probable vs. Possible
 - How will things change (or not) tomorrow?
-

Map of Malware Campaigns to Exploits

Malware Campaign

- Android Pjapps
- Android Droid Dream
- Android Zeahache

Distributed via: Android Market
Exploits Phone? Yes
Exploits Apps? No
Exploit: Exploid
Exploit: RageAgainstTheCage

CVE: NoCVE (common)

Author: "stealth"

Target: Root-owned Android Userland (adbd)

Blame: Google

Technique: RLIMIT_NPROC

Affects: Android ??? - 2.2 (difficult to identify)



Mobile Attacks Through 2011

500+

of Attacks

81

of Malware Families

16

of Malware Families
that Escalate Privileges

1

of Attack Vectors

Mobile Attacks Through 2011

3

of Unique Privilege Escalations
used w/ Malware

1

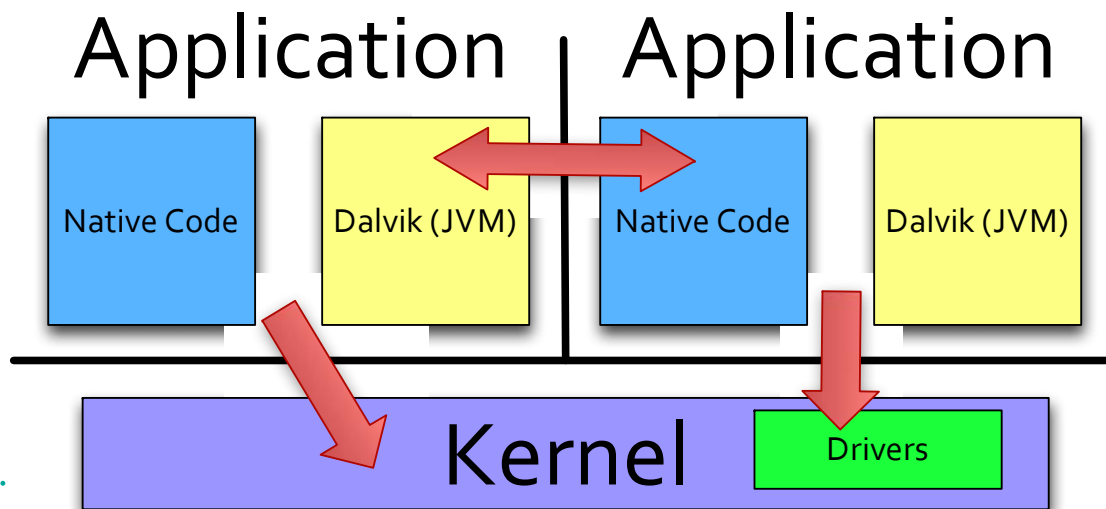
of Privilege Escalation Authors
used w/ Malware

Mobile OS Background

- 90% of what you need to know about mobile in two slides

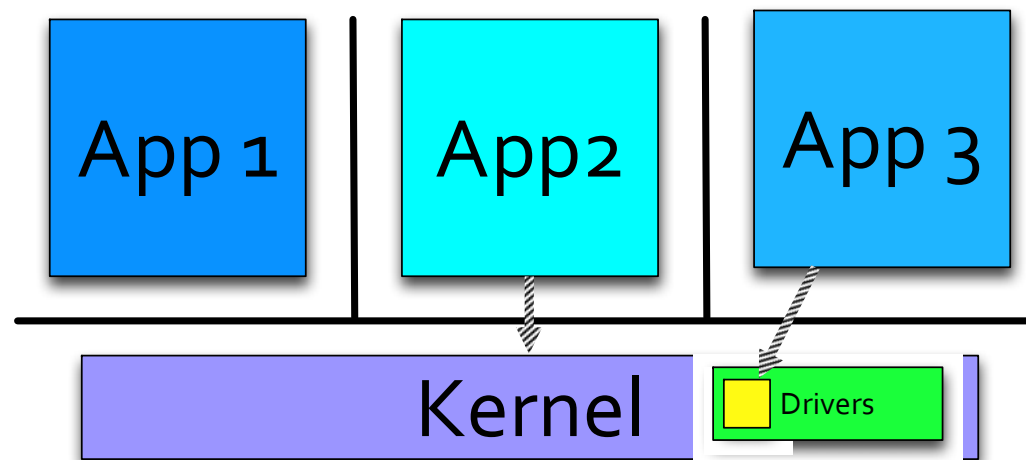
Android Security Model

- Each app runs as a different user and group
 - Apps cannot access data from other apps*
 - Permissions determine ability to perform RPC*
- Apps can access any other resources they want
 - Apps can access the kernel, drivers, syscalls, etc.
 - No Security Manager, no Java Sandbox



iOS Security Model

- Apps run as the same user, but ...
 - Apps must be signed by Apple
 - Apps are given a unique ID and directory by Apple
- Seatbelt restricts apps from accessing anything else
 - Apps cannot access data from other apps (mandatory)
 - Attack surface of kernel is reduced via Seatbelt



Privilege Escalation is Essential

- Two ways malware can get your mobile data
 - You give them permission (inside your control)
 - They jailbreak the phone (outside your control)
 - We only care about attacks that get access to data
 - E-mail, login credentials, other application data...
 - Not concerned w/ attacks that don't compromise data
 - Let's look at two recent attacks
-

Why Jailbreaks Matter Most



Jester Android + iOS Walkthrough

1. Download public exploit code
2. Add support for iOS + modified shellcode



3. Deliver exploit
4. On connect, execute local root exploit
5. Steal data and exfiltrate



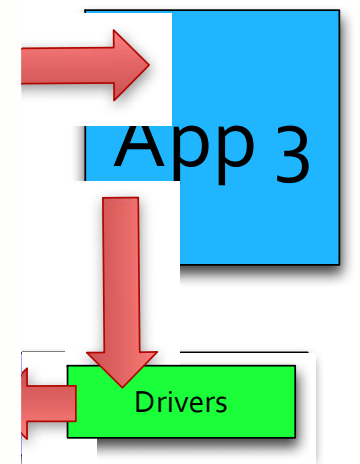
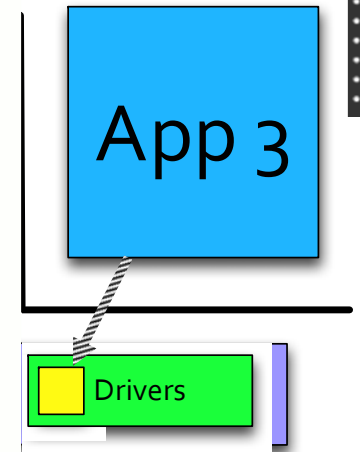
This is

This is
on J

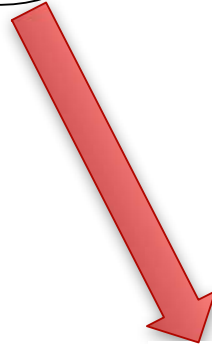
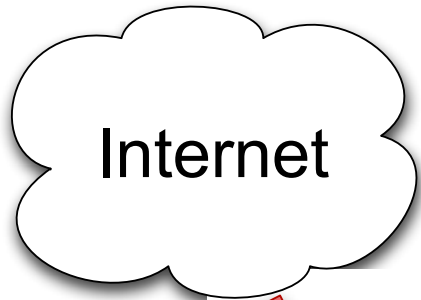
No Code S
No Sandbc
Courtesy S

Attack only

Attack only worked on Android 2.0 / 2.1 (6.6%)



CrowdStrike Exploit Walkthrough



Kernel

Drivers



Browser Permissions

- .INTERNET
- .ACCESS_FINE_LOCATION
- .ACCESS_COARSE_LOCATION
- .ACCESS_FINE_LOCATION
- .ACCESS_DOWNLOAD_MANAGER
- .ACCESS_NETWORK_STATE
- .ACCESS_WIFI_STATE
- .SET_WALLPAPER
- .WAKE_LOCK
- .WRITE_EXTERNAL_STORAGE
- .SEND_DOWNLOAD_COMPLETED_INTENTS

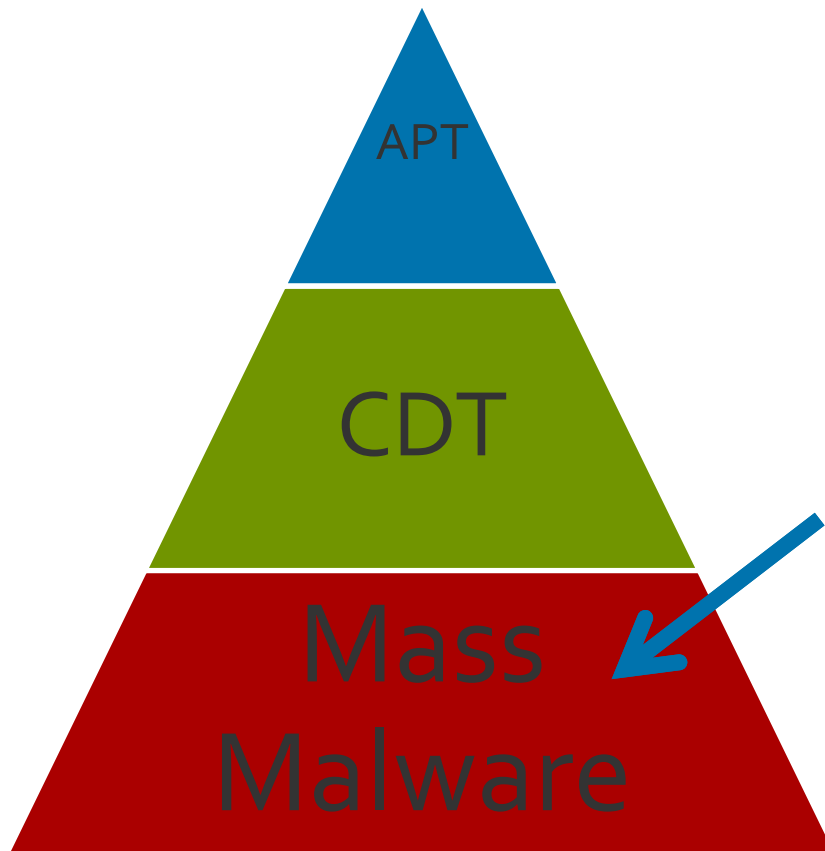
What Was Left Out

- CrowdStrike had to escalate privileges to do anything
 - Probably GingerBreak although they didn't specify
 - Reliable public jailbreak that works in Android < 2.3.4
 - They can access any functionality after jail breaking
 - Install backdoors, record phone calls, etc.
 - Both possible and easy under the right circumstances
 - But, no evidence that APT has done this at all
 - Not enough data to draw meaningful conclusions
-

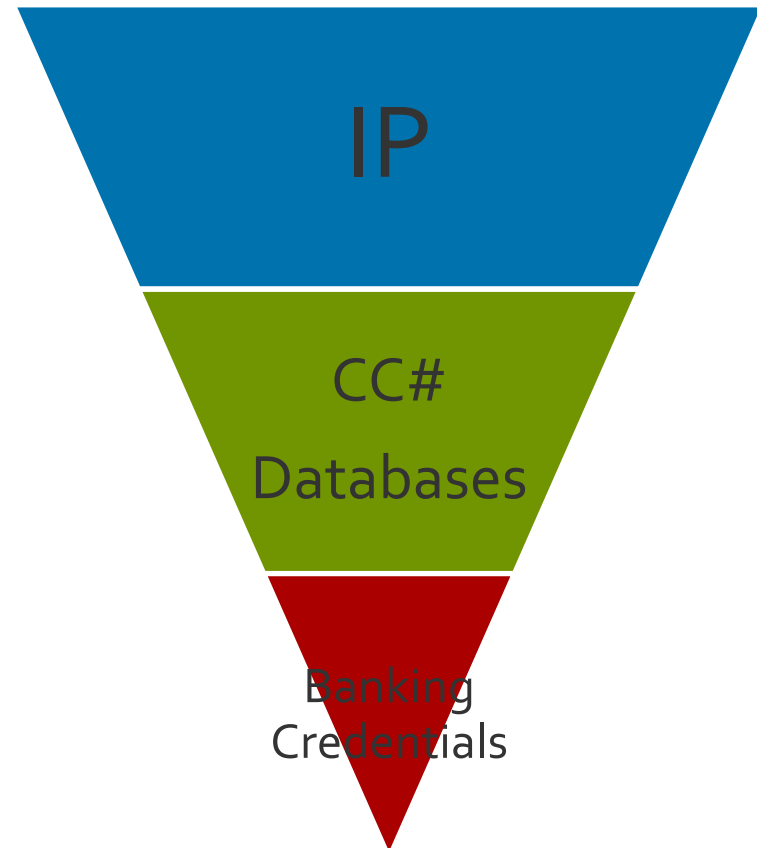
Modeling Attacker Behavior

Maslow's Hierarchy of Internet Threats

of Attacks



Value of Data Loss



Attacker Math 101

- What we know from Mobile OS architectures
 - $\text{Cost}(\text{Attack}) = \text{Cost}(\text{Vector}) + \text{Cost}(\text{Jailbreak})$
- What we know about attackers in general
 - $\text{Cost of Attack} < \text{Potential Revenue}$

Cost of Attack

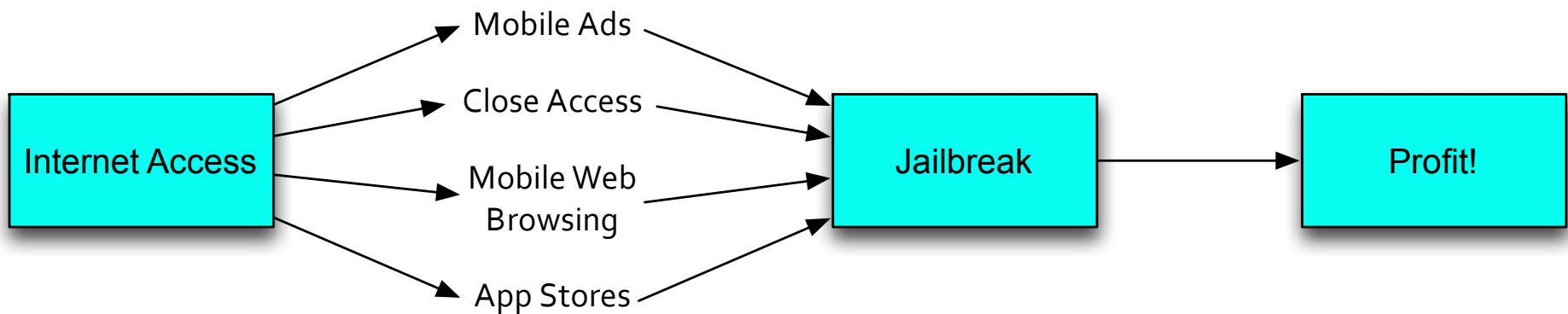
- Ease
- Enforcement
- Established Process

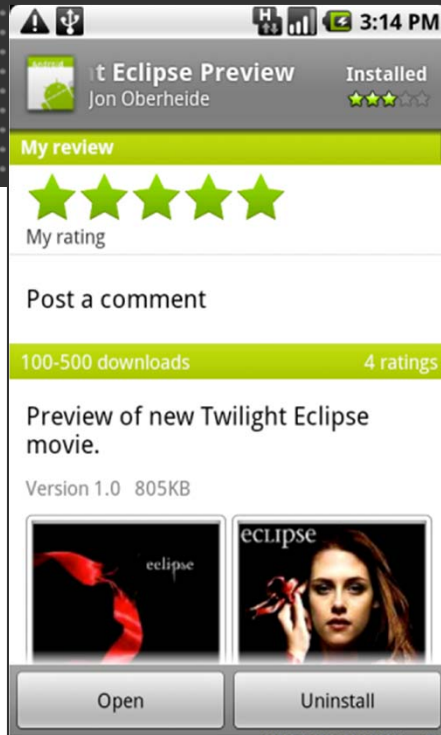
Potential Revenue

- # of Targets
- Value of Data
- Ability to Monetize

Attack Vectors

1. Mobile Ads
2. Close Access
3. Mobile Web Browsing
4. App Stores





Mobile Ads



Anonymous



ID Verified



Anonymous

Low Cost



\$300k min



\$50 min

Scriptable



HTML5



Img / Text

High Traffic



Low interest from legit advertisers

Mobile Ads

- Someone actually tried this – one reported case
 - No auto-exploit possible, link to download APK
 - Relied on social engineering to work
- GGTracker – SMS Fraud
 - Didn't escalate privileges
- Apparently didn't work well?
 - Mid-2011, no repeats since
- May be revisited in the future
 - If more apps use them
 - If scripting capability granted



Close Access

- Insecure Storage, NFC, Bluetooth, Wi-Fi, Baseband
 - All require proximity or possession of device
- Attacks that require close access don't easily scale
 - Can someone think of one that does?
- Credit card skimmers!
- Why are skimmers abused?
 - Magstripes are ubiquitous
 - Skimmers are dirt cheap
 - They have access to data I want

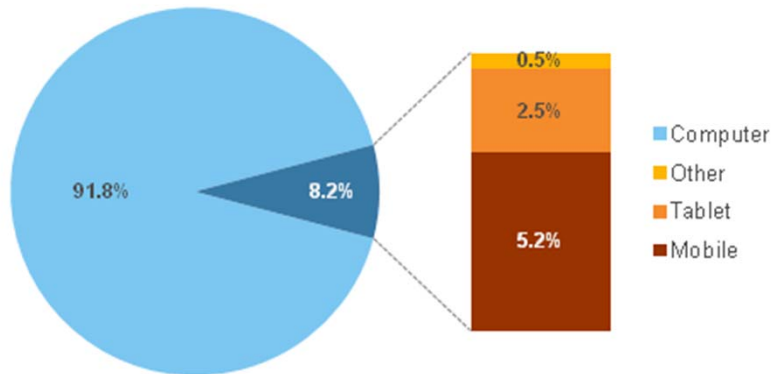


Close Access

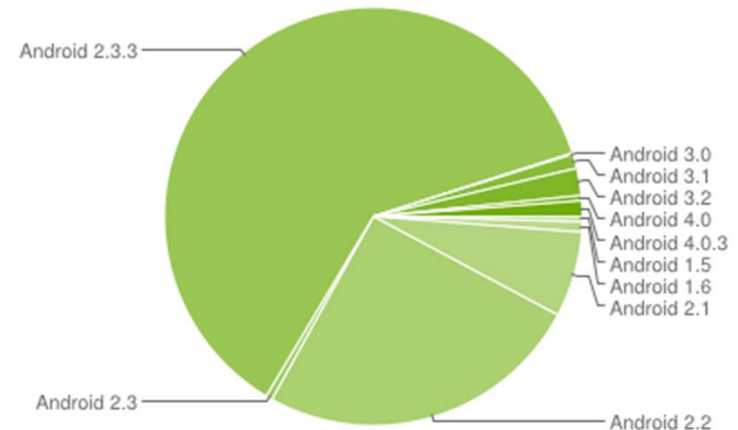
- Issues for abuse of mobile close access vector
 - NFC, Bluetooth, Wi-Fi not as ubiquitous as magstripes
 - May not allow collection of data or grant access that I want
 - Cost of exploitation not dirt cheap or commoditized yet
 - No ready-made close access tools available online yet
 - Baseband exploitation never likely to become cheap
 - Risk of arrest due to physical proximity is unchanged
 - *Zero* cases of mass malware through close access
-

Mobile Web Browsing

Share of Connected Device Traffic in the U.S.
Source: comScore Device Essentials, U.S., December 2011



~8% of total web traffic comes from mobile devices



Breakdown by version / features (+ varying rates of feature support)

Mobile websites might not have any ads!



Mobile Web Browsing

- 10-20x less potential targets than desktops
 - Not many mobile browsers, split between platforms
 - Mobile websites commonly won't have ads
 - Increased costs to exploit relative to desktops
 - Feature disparities, in particular flash support
 - Multiple exploits required for browser + jailbreak
 - However, may be able to achieve anonymity easily
 - Possible, but incentives are stacked against it
 - *Zero* identified cases in the data
 - Might change if # of targets rises dramatically
-

Vendor App Stores

- Principle difference in Mobile vs. Desktop OS's
 - Windows 8 Store & Mac App Store coming soon!
- Huge # of potential targets on App Stores
 - Every device uses the vendor app store
 - 300+ million devices on each of iOS and Android
- Reduced costs to exploit and escalate privileges
 - Apps run code locally. Who needs a browser exploit?
 - You can submit apps nearly for free, low upfront costs
 - Manipulation of SEO is simple and easy

App stores look like a great value proposition!

App Submission Process

- In order to reach 315 million iOS devices:
 1. Pay to join the iOS Developer Program (\$99)
 2. Identity verified by Apple: SSN/Phone Call/DUNS
 3. Apps reviewed for content and banned APIs
 4. No runtime modification – only reviewed code is run

 - In order to reach 300 million Android devices:
 1. Pay to enter Google Distribution Agreement (\$25)
 2. Fill out developer information in form online
 3. Apps reviewed through dynamic analysis (Bouncer)
 4. Runtime modification allowed – can load new code
-

Malicious App Submission Process

- In order to submit a malicious iOS app:
 1. Create a believable false identity or risk arrest
 2. Create a believable app that passes a content review
 3. Avoid banned APIs (might be useful) / known exploits
 - In order to submit a malicious Android app:
 1. Put fake developer information into a form online
 2. Avoid malicious activity until after Bouncer runs it
 - a. Package inside app -> wait two weeks to activate
 - b. Package outside app -> download code at runtime / update
-

Malicious App Campaigns

0

Apple App Store

30

Google Marketplace

“Say what you will about police states, but they have very little crime.”

3rd Party App Stores



- Are 3rd party app stores as attractive to abuse?
 - <10% of total devices*, use is split between markets
 - In strange reversal, 3rd parties may dominate in China
- The cost of exploitation needs to be very low
 - For iOS, access to 3rd party means device is jailbroken
 - Ability to review apps increases with size

* <http://www.wired.com/gadgetlab/2009/08/cydia-app-store/>

Malicious App Campaigns (3rd Parties)

20

US-based 3rd Party

32

Chinese 3rd Party

Abuse of 3rd party markets is happening *now* (only on Android)

Attack Vector Takeaways

- $\text{Cost}(\text{Attack}) = \text{Cost}(\text{Vector}) + \text{Cost}(\text{Jailbreak})$
 - Assume that $\text{Cost}(\text{Jailbreak}) = 0$ and exploits are free
 - Would app stores be abused in this scenario?
 - ID verification, app review, and code signing say NO on iOS
 - Google has none of these, also missing from AdMob
 - What if browser exploits were free too?
 - Was already the case on Android / JailbreakMe on iOS
 - Profit potential is not there, otherwise abuse would occur
 - # of vulnerable targets is too low for browser exploitation
 - High amount of patching or low amount of surfing
-

Mobile Exploits

- If I just had a jailbreak, then I could make money...

Android Exploits

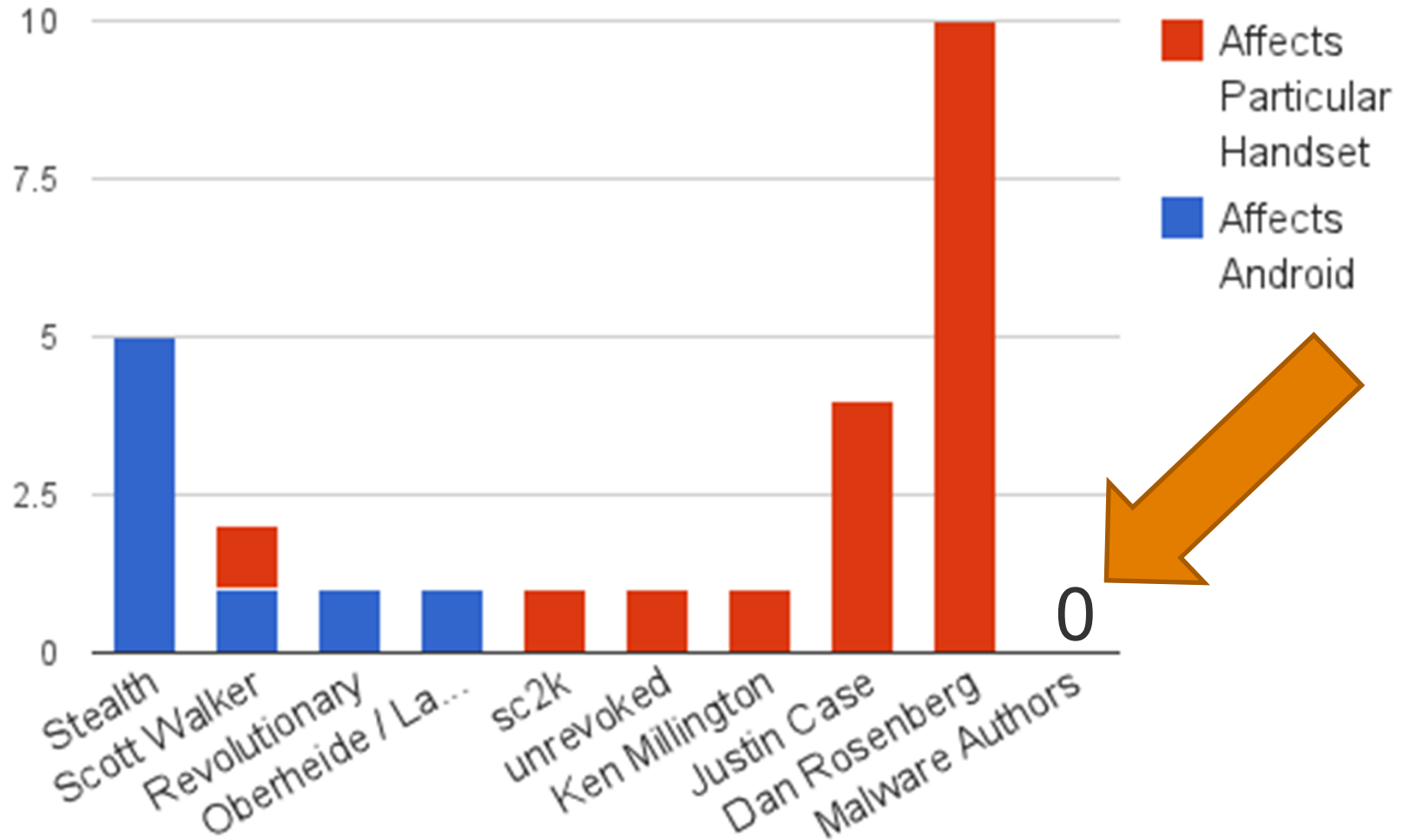
- Active Jailbreaker community, many free exploits
 - 26 separate jailbreaks from 10 different authors
 - As soon as a phone gets popular, it gets jailbroken
 - Previously noted “sandbox” design makes this easier
 - Google made no attempt to reduce attack surface here
 - Permissions have *nearly zero* effect on ability to exploit
 - No clearinghouse for Android vulnerability info (CVEs)
 - Even exploited vulns are untracked by Google!
 - Serious lack of info, this needs to change
-

Android Exploit Mitigations

- Must learn English to Google for exploits...
- Beg carriers to unlock bootloader...



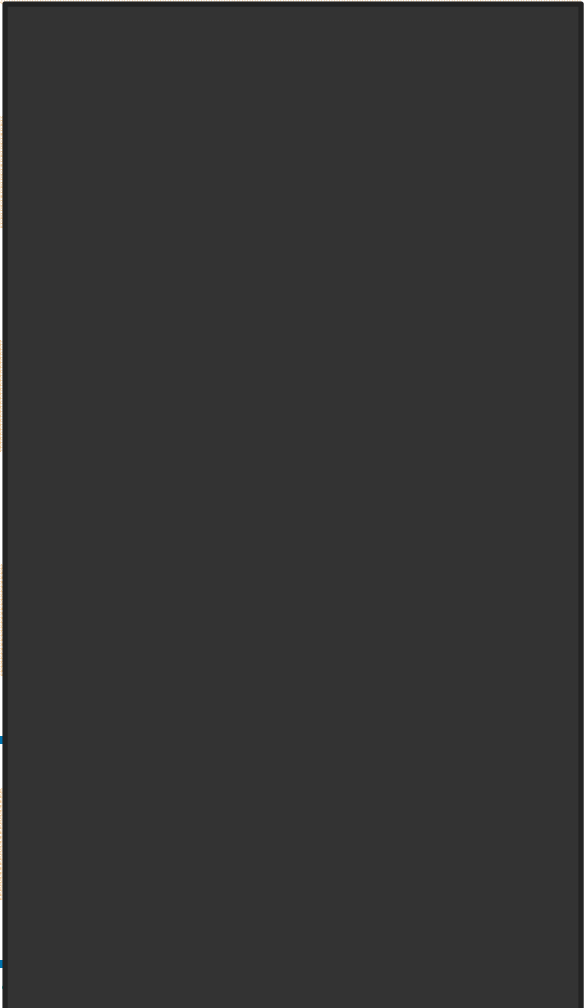
Android Jailbreak Dev by Target



What we want to know:

Which exploits get used by malware and why?

Universal Android Exploits

| Exploit Name | Last Affected Version | Abused? |
|-----------------------|--------------------------------|--|
| Exploid | 2.1 (Éclair) |  |
| RageAgainstTheCage | 2.2.1 (Froyo) | |
| Zimperlich | 2.2.1 (Froyo) | |
| KillingInTheNameOf | 2.2.2 (Froyo) | |
| Psneuter | 2.2.2 (Froyo) | |
| GingerBreak | 2.3.4 (GingerBread) | |
| zergRush | 2.3.5 (GingerBread) | |
| Levigator | 2.3.5 (GingerBread) | |
| mempodroid | 4.0.3 (ICS) | |

Android Patch Stats – 03/12/2012

| Platform | Codename | Distribution |
|---------------|-----------------------|--------------|
| 1.X | Cupcake / Donut | 1.2% |
| 2.1 | Eclair | 6.6% |
| 2.2 | Froyo | 25.3% |
| 2.3.0 - 2.3.2 | Gingerbread | 0.5% |
| 2.3.3 – 2.3.7 | Gingerbread | 61.5% |
| 3.X | Honeycomb | 3.3% |
| 4.X | Ice Cream Sandwich | 1.6% |

Reported by API level only. Difficult to determine accurate exposure.

<https://developer.android.com/resources/dashboard/platform-versions.html>

Android Patching Rates

- It doesn't matter when Google patches Android trunk
 - They need to hit AOSP repo first, then carrier gets it
 - OEMs/Carriers seem to treat handsets as disposable
- Lookout used their data to track vuln half-lives
 - # of days until 50% of Android Lookout users patched

| Exploit | Time to Patch 50% |
|--------------------|-------------------|
| Exploit | 294 days |
| RageAgainstTheCage | > 240 days |

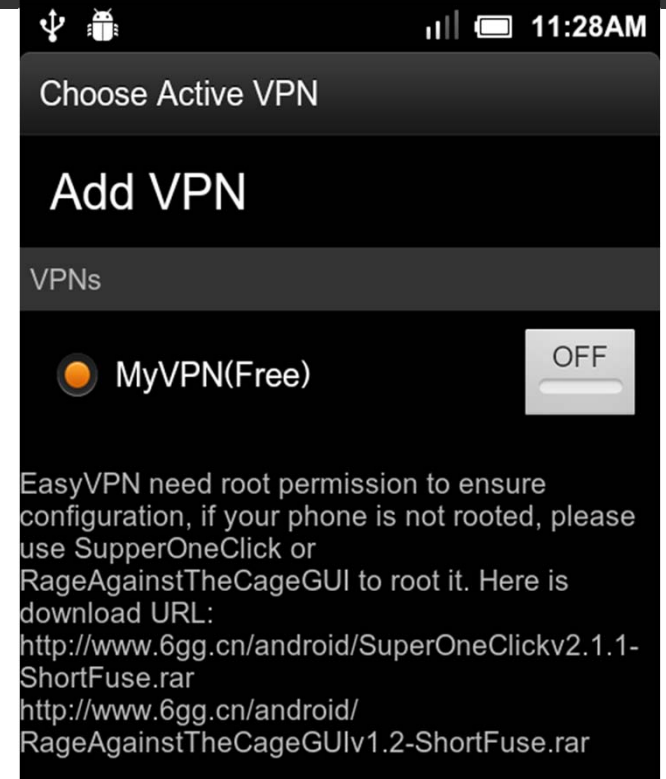
Exploits When You Want Them

“My Gingerbreak works, but I wont release it before a couple of devices are in the wild so the issue is not fixed before it can become useful.”

-- stealth (prior to releasing Gingerbreak)

Jailbreak Equivalents

- Android Private Signing Keys
 - jSMShider: <http://goo.gl/vPzjg>
 - Affects custom ROMs only
- Have the user do it (no joke) ----->
 - Lena: <http://goo.gl/eiTBA>
- Request Device Admin API Privs
 - DroidLive: <http://goo.gl/c3EET>
 - Android 2.2+
- All distributed as apps in App Markets (1st and 3rd party)
- Less effective (user interaction), less used, still works



What about other paths to data?

- App-to-App: not high enough potential revenue
 - Malware authors already don't use handset exploits
 - Are there more installs of one app than a Nexus S?
 - Most app data cannot be easily monetized
 - What can an attacker do with 250k Yelp credentials?
 - Remote-to-App has similar problems
 - What's the vector? How many apps use URL handlers?
 - How many # of targets? What's the value of the data?
 - No evidence of App-to-App, Remote-to-App
 - Why exploit apps when universal jailbreaks are free?
 - Prediction: App-to-App exploited when jailbreaks dry up
-

Android Mitigation Outlook



- Chrome for Android
 - Makes browser exploits hard
 - Not an exploited vector now
 - No effect on current Android malware



- SEAndroid
 - Kills userspace jailbreaks, but not kernel!
 - Jailbreakers delayed, will have to retool
 - What carrier will use it?



- ASLR in Ice Cream Sandwich 4.x
 - Little to no effect on jailbreaks
 - Useful to make browser exploits difficult
 - Can't help 300+ million existing devices

Android Exploitation Takeaways

- The only exploits abused are public jailbreak exploits
 - No demonstrated ability by mass malware to write exploits
- Google does not care about Android Jailbreaks
 - They make no attempt to mitigate them in the OS
 - They don't track vulnerabilities that allow them
 - None of their upcoming enhancements mitigate them
 - Platform is filled with alternate escalation scenarios
- Patching situation on Android is insane – jailbreak lifetime++
 - Google has no ability to force carriers / OEMs to react
 - Even if they could, it's too easy to write new exploits
 - Low amount of surfing likely the reason for no browser exploits

If you can install an app, you can take over Android

iOS Exploits

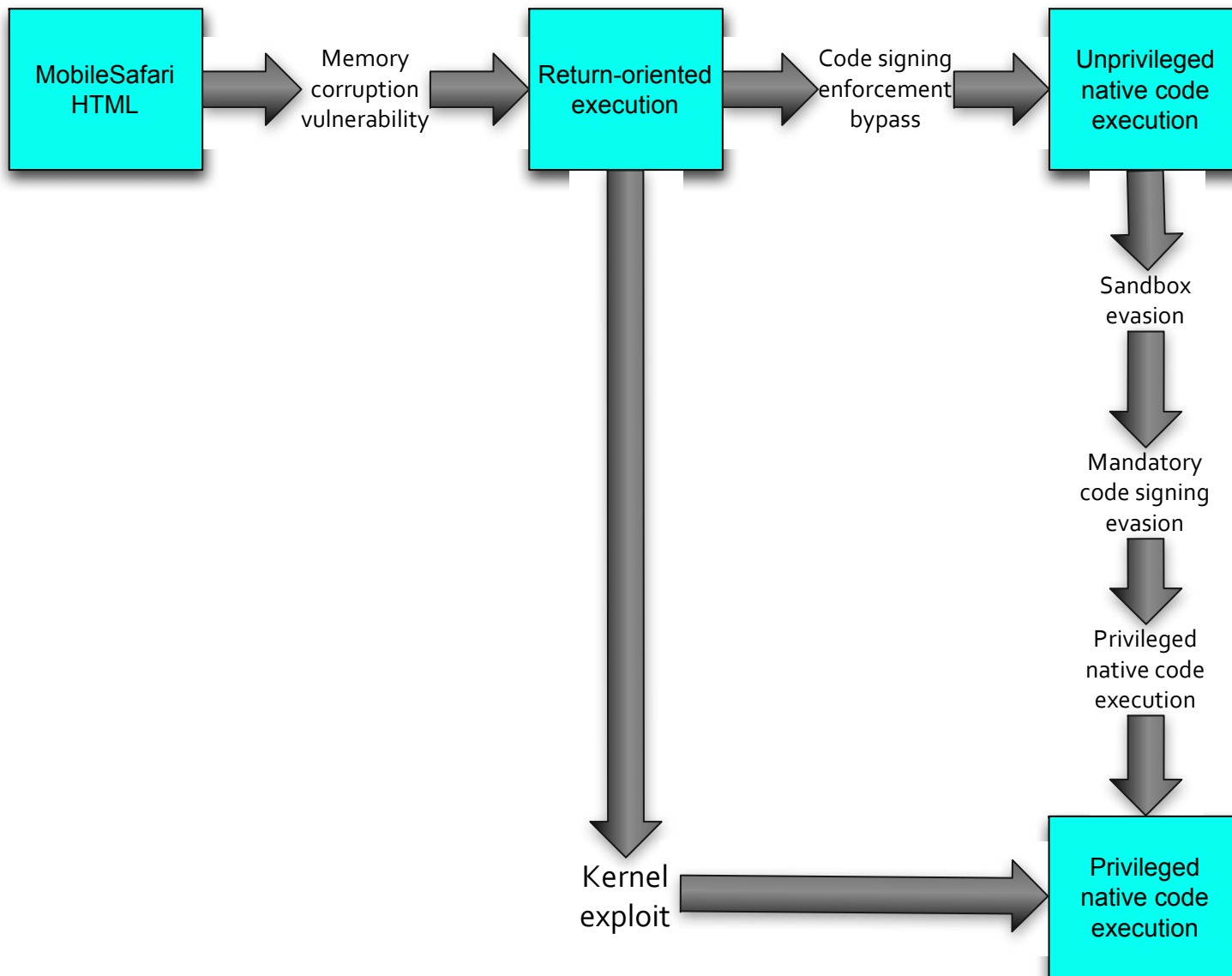
- All iOS exploits written by Jailbreakers
 - 25 separate jailbreaks from ~4 main groups
 - No evidence of abuse by malware authors at all!
 - Security researchers at conferences don't count
 - However, jailbreaker behavior mimics attackers
 - Want cheapest possible jailbreak & most possible use
 - Deploy exploits strategically (limer1n vs. SHAtter)
 - Choose target attack surfaces for maximum return
 - Boot ROM (unpatchable) vs. iOS (quickly patchable)
-

iOS Exploit Mitigations

- Code Signing introduced in iOS 2.0 (2008)
 - App must be signed to start execute
 - *Memory pages* must be signed at runtime
 - Prevents introduction of new code, like DEP
- ASLR introduced in iOS 4.3 (2011)
 - Apps need to turn on PIE and most 3rd party don't
 - However, all system apps are compiled with PIE
- The Seatbelt Sandbox
 - Restricts kernel attack surface somewhat
 - Increases difficulty of finding kernel vulnerabilities

Exploiting iOS is no weekend task

iOS Exploitation Requirements



Malicious iOS Apps

- Apple needs to review your app, unless...
- Need to find a flaw to allow code injection at runtime
 - Charlie Miller's mmap / Stock Trader exploit
 - Embedded mmap logic flaw into legit looking app
 - Submitted to App Store with real identity
 - Was able to inject new code at runtime
- This is the only known instance of potentially malicious content in the iOS App Store

Would you submit your sophisticated iOS kernel exploit to Apple for review?



ASK FOR FORGIVENESS, NOT PERMISSION

Differences b/t Research vs. Actual

- Charlie Miller signed up with his real identity
 - He had no perceived sense of risk: legal or criminal
 - $\text{Cost}(\text{Vector}) = 0$ for Charlie
 - His malicious app relied on a unique vulnerability
 - Attackers have no demonstrated discovery capabilities
 - This attack surface is likely small, unique bugs are rare
 - Why didn't this get immediately abused?
 - Apple patched in *4 days*, reducing potential revenue
 - Charlie didn't discuss until after patch, no PoC code
-

Apple Patch Response

- Apple patches fast and has control of platform
 - Discourages repeat research for jailbreaks
 - Potential profit is sharply limited after a few days

| Exploit | Jailbreak | Response Time |
|-----------------------------|------------------------------|---------------|
| Malformed CFF | Star (JailbreakMe 2.0) | 10 days |
| T1 Font Integer Overflow | Saffron (JailbreakMe 3.0) | 9 days |
| mmap Logic Flaw | Charlie | 4 days |



iOS 5 Patching Rates

| Time Since iOS 5 Release | % of iOS users on 5.x |
|--------------------------|-----------------------|
| 5 days | 20% |
| 3 weeks | 40% |
| 3 months | 66% |

- Updates generally paired with new features
- iPhone 3G / iPod Touch 2G unsupported (2008)
- Latest version of Android on 1.6% of devices today

5 days: <http://goo.gl/P2hkx> 3 weeks: <http://goo.gl/KhR9p> 3 months: <http://goo.gl/yGiKg>

Apple Patch Response

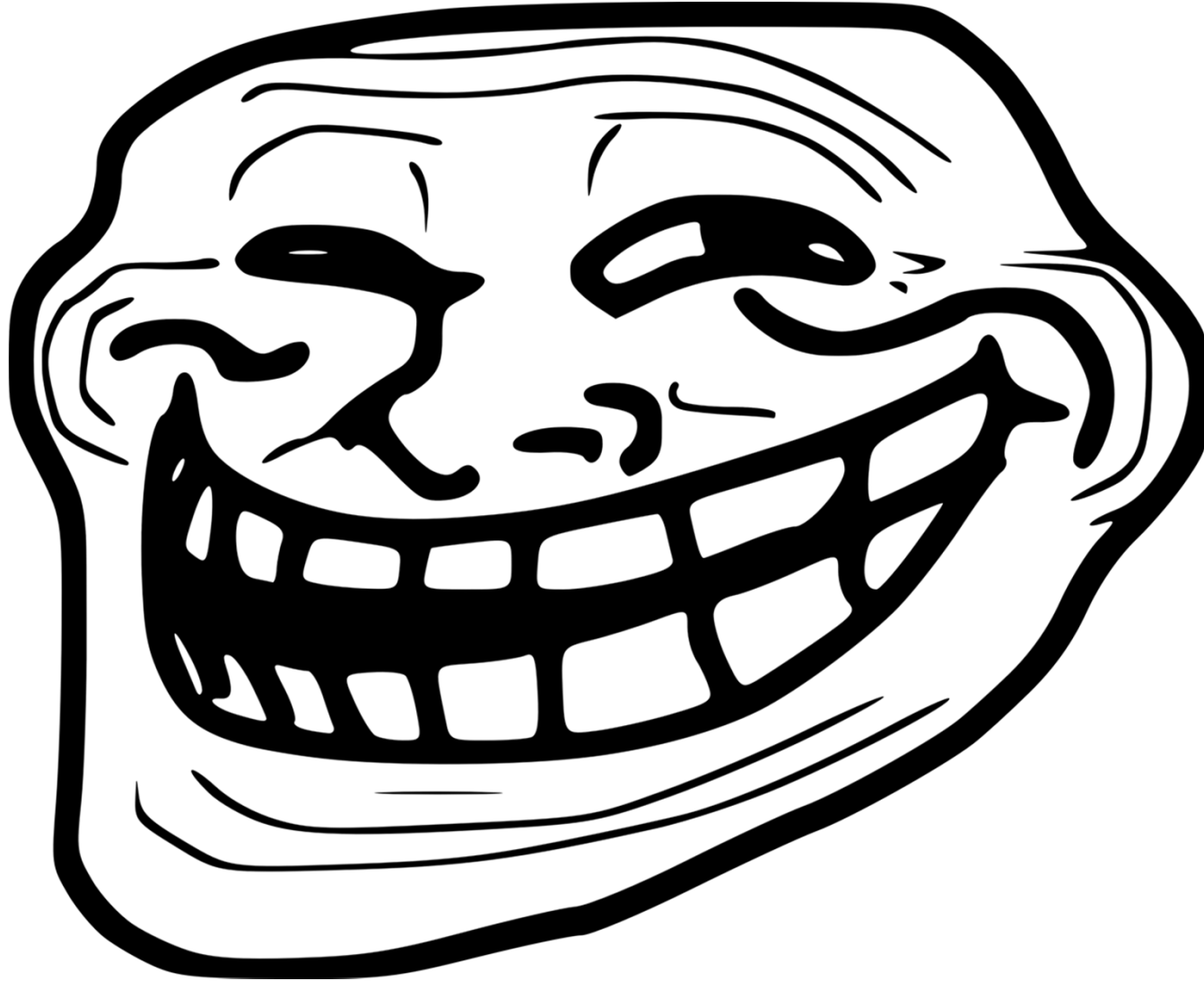
- Many users don't sync to iTunes for updates*
 - Apple has shifted to OTA updates in iOS 5



- Patching incentivizes unpatchable bugs in Boot ROM
 - No Boot ROM exploits for some time despite value
 - Remaining bugs must be extremely difficult to find

* <https://www.mylookout.com/mobile-threat-report#vulnerabilities-patching>

What about iOS App-to-App?



iOS Exploitation Takeaways

- No malicious use of iOS jailbreaks by mass malware
 - No demonstrated ability by mass malware to write exploits
 - Despite availability of exploits, limited potential revenue
 - Cost(Jailbreak) slows rate of discovery -> have to react less
 - iOS OTA updates and platform control allow quick reaction
 - Malicious app distribution is harder than just submission
 - Code signing is hostile to malicious apps
 - Nearly entire App-to-App attack surface is absent
-

Conclusions

- Long talk is long



Conclusions

- Attackers carefully balance incentives w/ strategy
 - Not all attack vectors will be explored maliciously
 - Intel-driven approach: concrete results from concrete data
 - Android will continue being compromised by malicious apps
 - Bouncer, Chrome Browser, ASLR have limited impact
 - No mitigations to slow jailbreaks, no ability to react w/ patches
 - iOS will steer clear of similar attacks
 - Real-world verification trumps all the technical attacks
 - Mitigations slow jail breaking, reacting quick reduces value
 - Future presentations will explore available defenses
 - Keep up to date at www.trailofbits.com
-

References

- Attacker Math 101, Dino Dai Zovi
 - <http://goo.gl/hVGjK>
 - iOS Security Evaluation, Dino Dai Zovi
 - <http://goo.gl/PcJ2U>
 - Exploit Intelligence Project, Dan Guido
 - <http://goo.gl/xzYVN>
 - Lookout Security Mobile Threat Report
 - <https://www.mylookout.com/mobile-threat-report>
 - Contagio Mini Dump
 - <http://contagiominidump.blogspot.com/>
-

References

- Don't Root Robots, Jon Oberheide
 - <http://goo.gl/A5XmR>
 - A look at ASLR in Android ICS, Jon Oberheide
 - <http://goo.gl/F8Bjl>
 - The Case for SEAndroid, Stephen Smalley
 - <http://goo.gl/KlQm6>
 - Practical Android Attacks, Bas Alberts and M. Oldani
 - <http://goo.gl/BwkLA>
 - Android Malware from Xuxian Jiang @ NC State
 - <http://www.csc.ncsu.edu/faculty/jiang/>
 - Androguard, Anthony Desnos
 - <https://code.google.com/p/androguard/>
-

Thank you!

Dan Guido, Trail of Bits



Mike Arpaia, iSEC Partners



Featured Member of the
TechTarget Editorial
Speaker Bureau