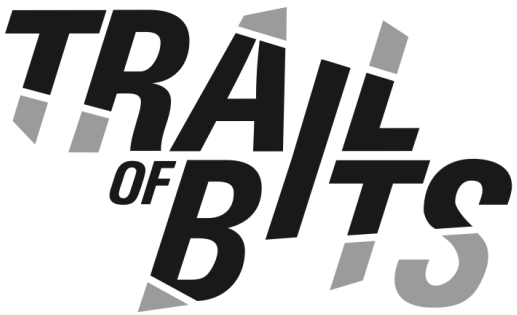
Mobile Exploit Intelligence Project

Dan Guido, Trail of Bits
Mike Arpaia, iSEC Partners

SOURCE Boston, 04/19/2012



Intro and Agenda

- Talk series discussing intelligence-driven security
  - Provide actual data on attacker characteristics
  - Provide analysis tradecraft to interpret it
    - Intrusion kill chains
    - Attacker characterization
    - Adversarial attack graphs

- Informed defense is more effective and less costly
  - Less hypothetical, more verifiable
  - Defenses supported by observation
  - "Technology doesn't beat determination"

- Secure Password Managers
- Heavy Metal that Poisoned the Droid
- Smartphone Apps are not that Smart

- Is Your Mobile Device Radiating Keys?
- Risk and Vuln Assessment of NFC
- Revisiting Baseband Attacks

- CrowdStrike Android Exploitation Demo
- iOS 5 – An Exploitation Nightmare?

Millions of Mobile Attacks

Attack Vector

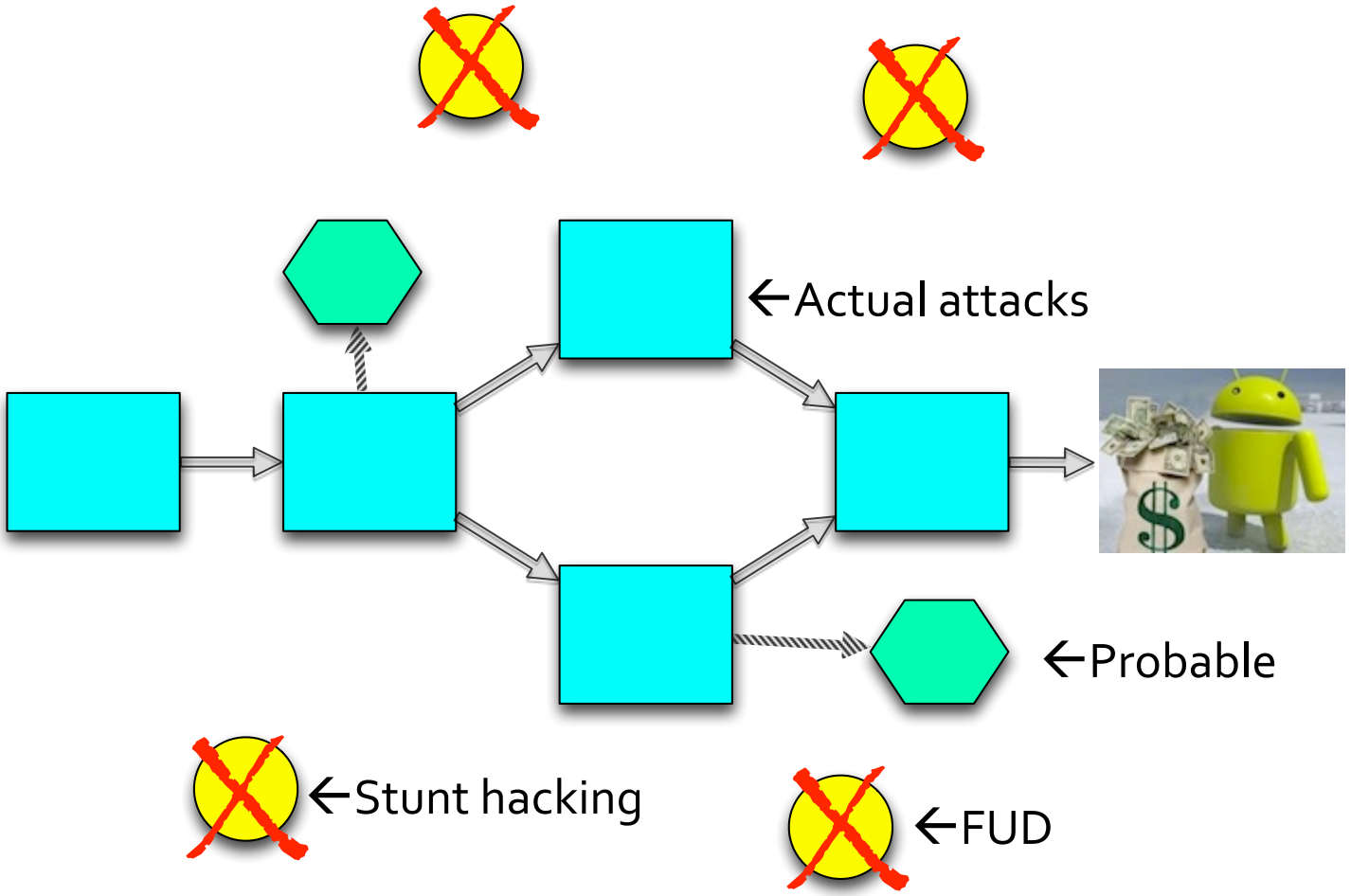
Exploits

Platform

* Android and iOS, 2011-2012

What are we doing wrong?

Your Defense Lacks Intelligence



Attackers choose the least cost path to their objective

Attacker Math 101

- Cost(Attack) < Potential Revenue
  - Attacks must be financially profitable
  - Attacks must scale according to resources

- Cost(Attack) = Cost(Vector) + Cost(Escalation)
  - What we know from Mobile OS architectures

Cost of Attack

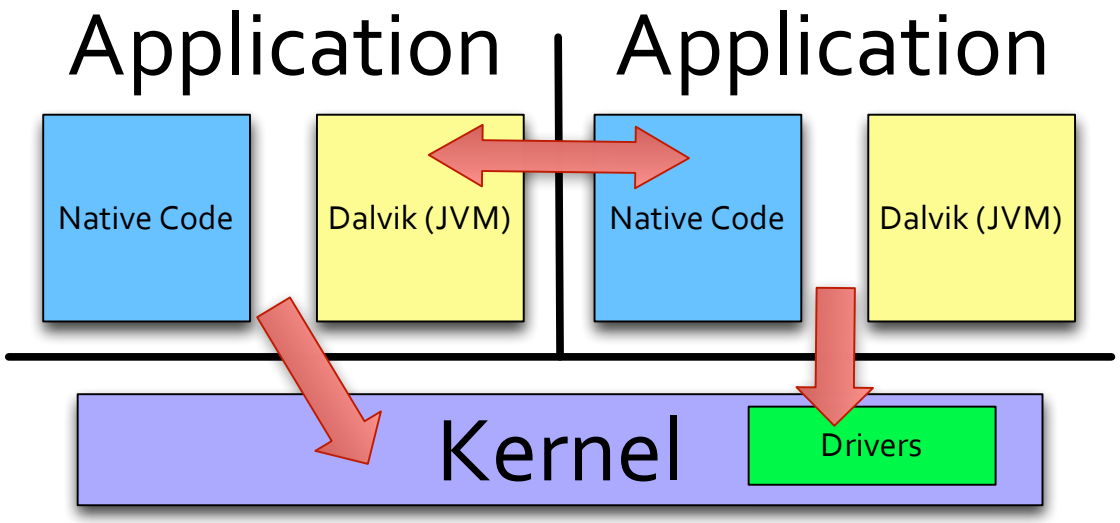
- Ease
- Enforcement
- Established Process

Potential Revenue

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

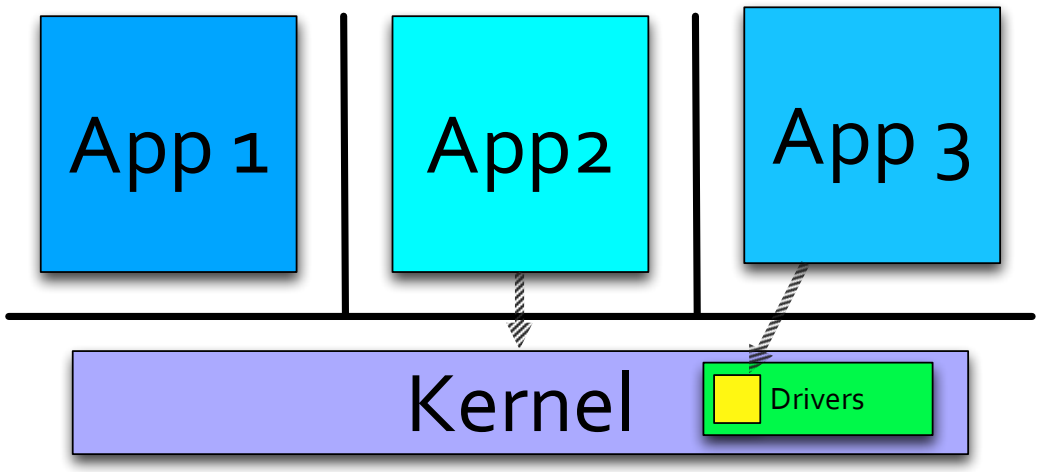
Android Security Model

- 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



Mobile Malware

How does it work?

Mobile Attack Data (2011-2012)

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"

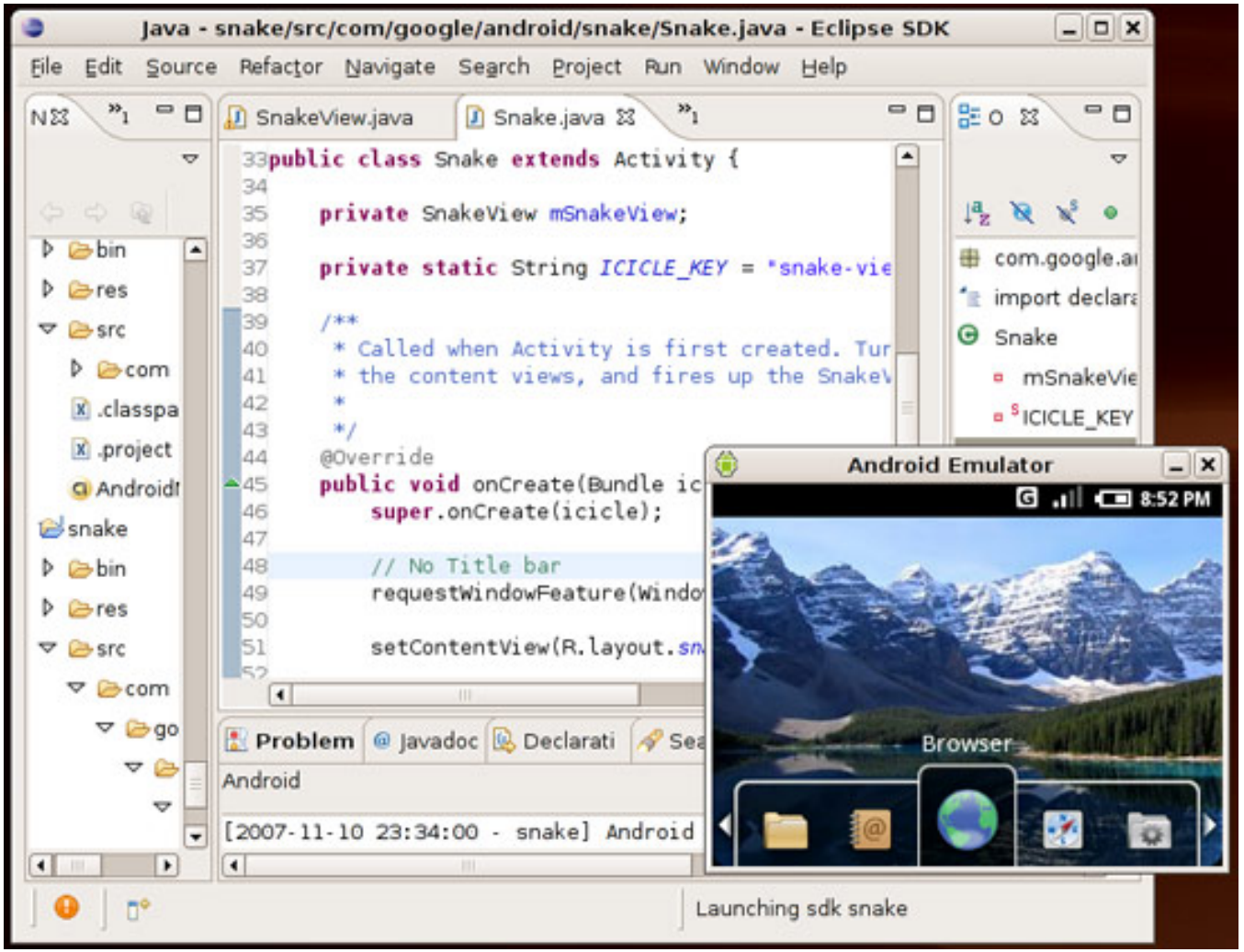
Target: Root-owned Android Userland (adbd)

Blame: Google

Technique: RLIMIT_NPROC

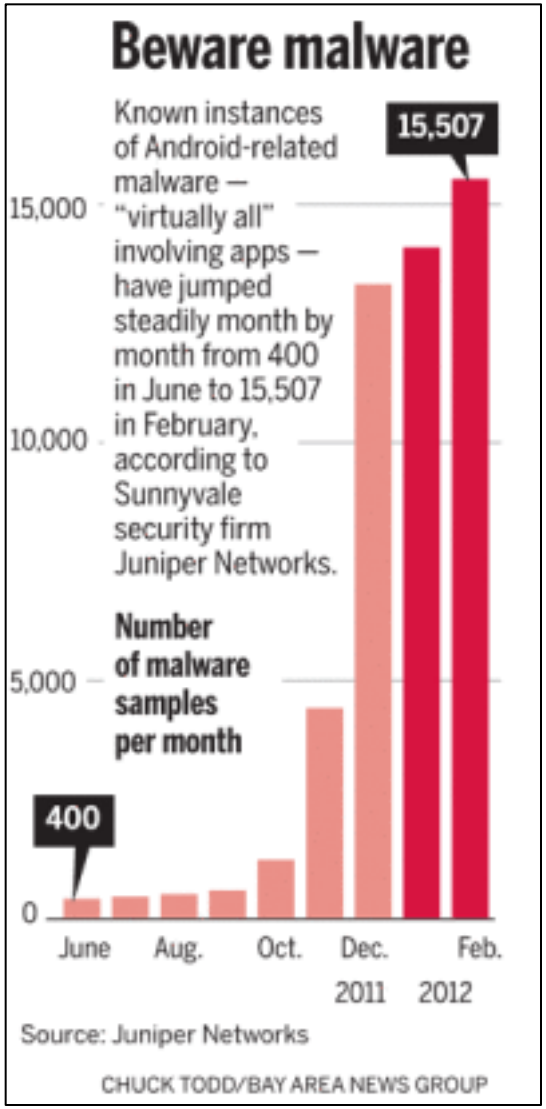
Affects: Android ??? - 2.2 (difficult to identify)

Develop Malware



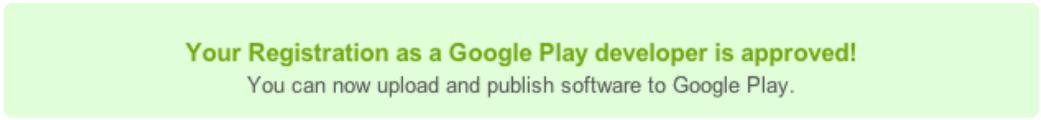
Find or develop malware to remotely interact w/ victim devices

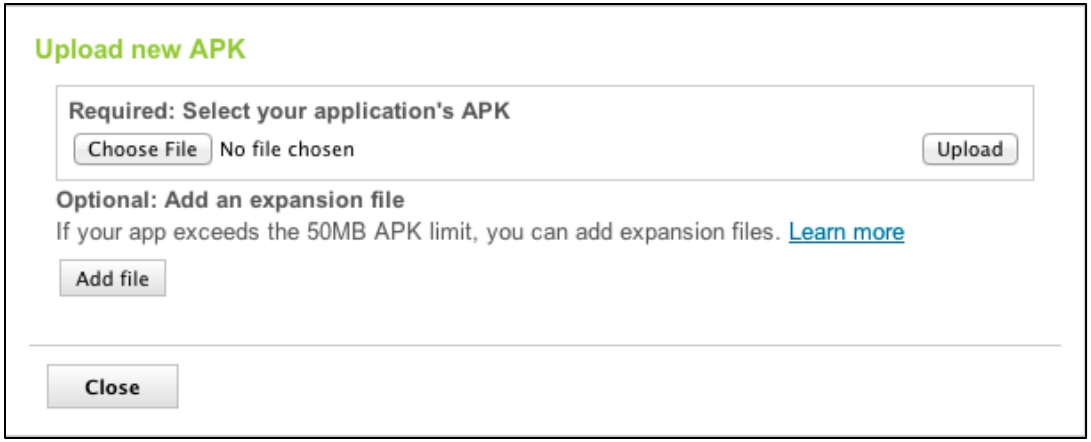
Add Malware to App

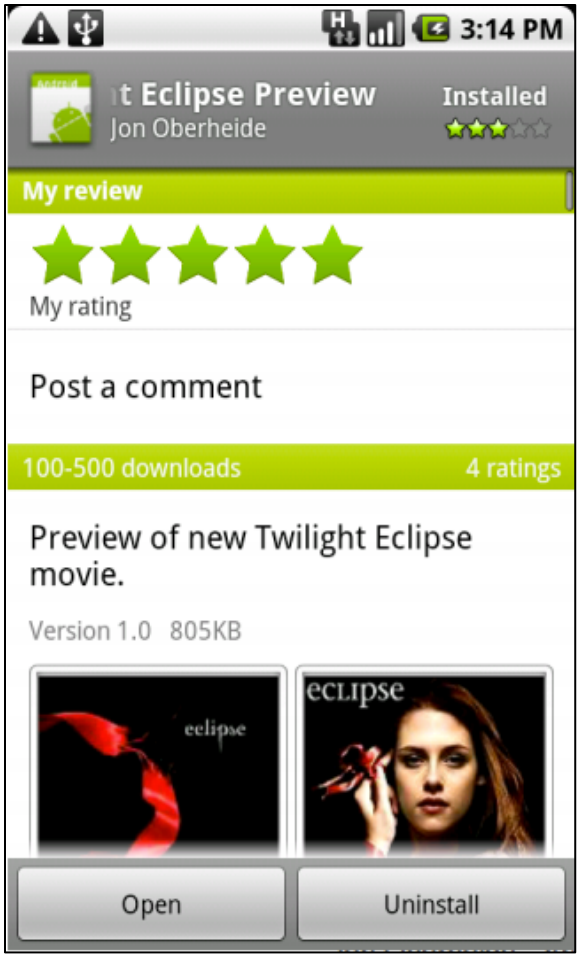


What would be likely for targets to install?

Gain Exposure

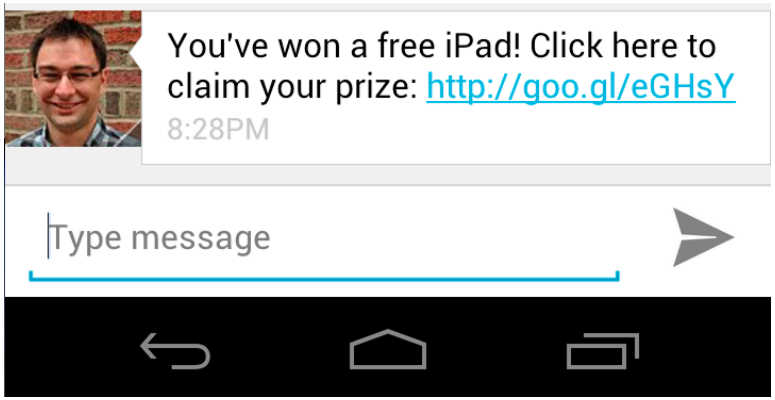


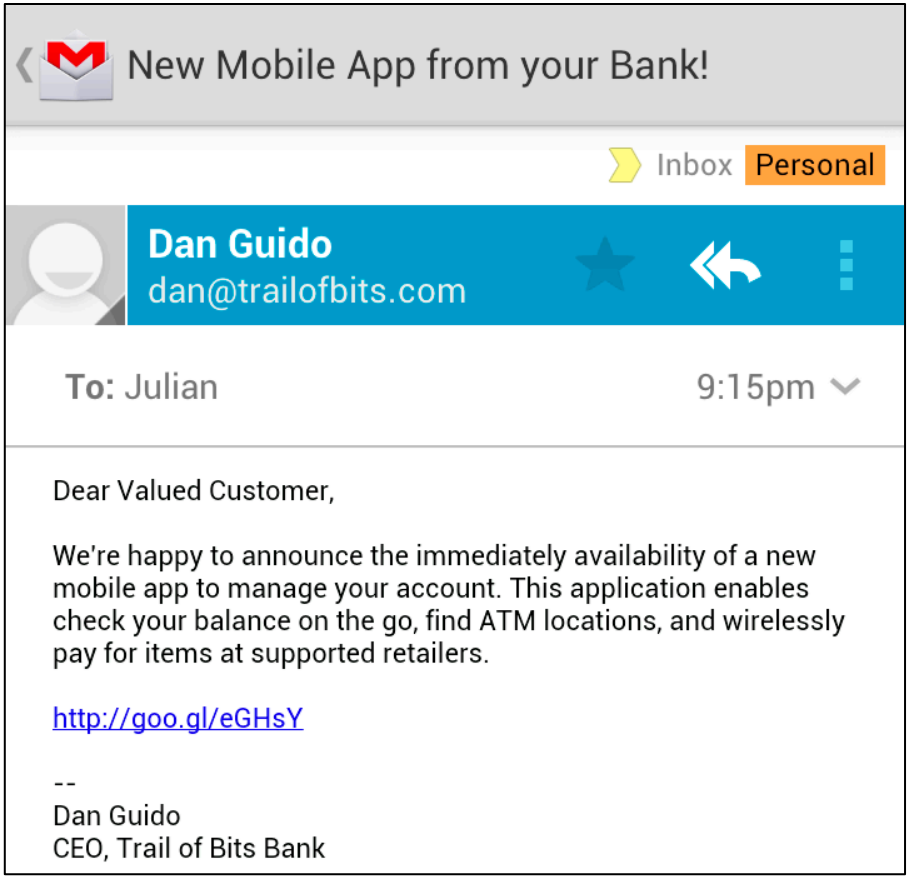




Upload the app somewhere victims can get it

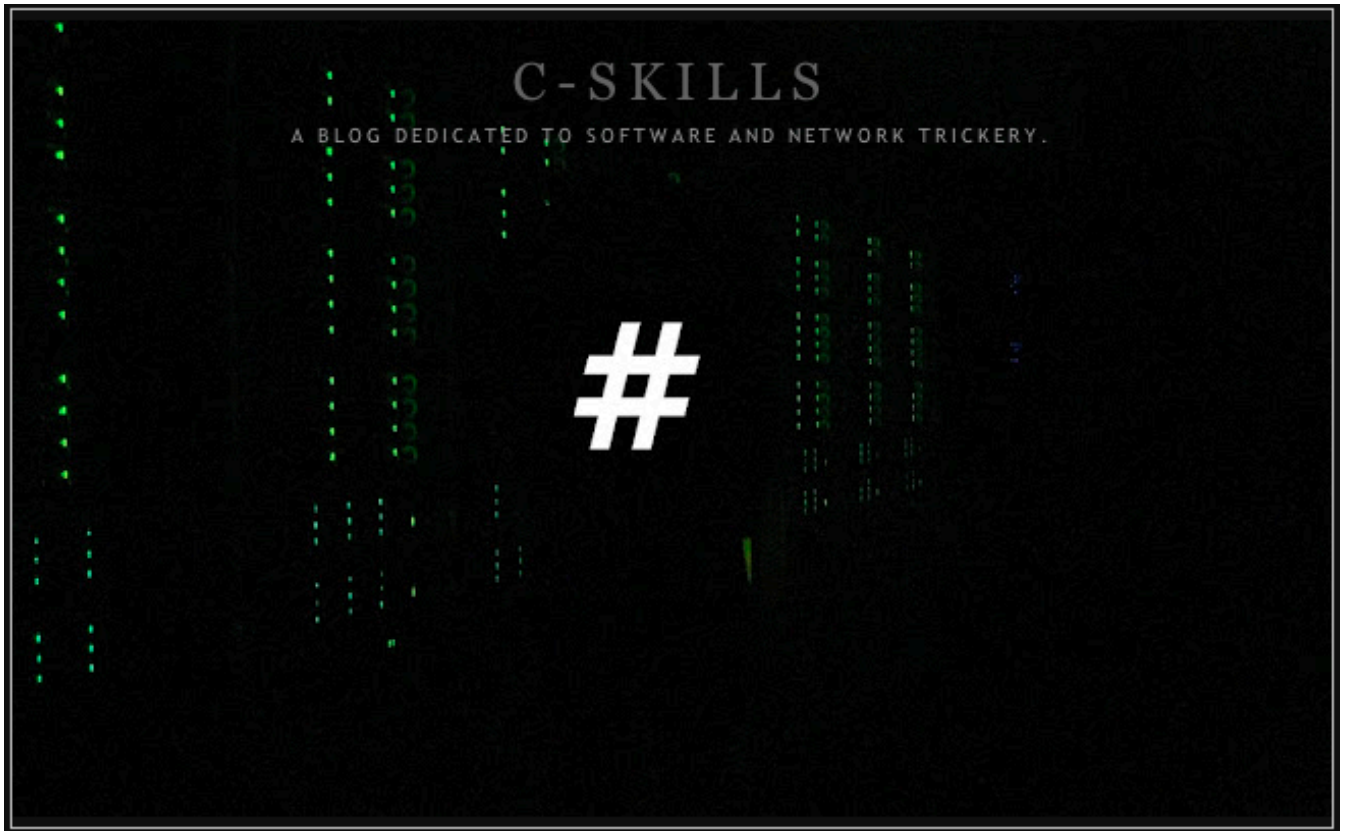
Drive Installs





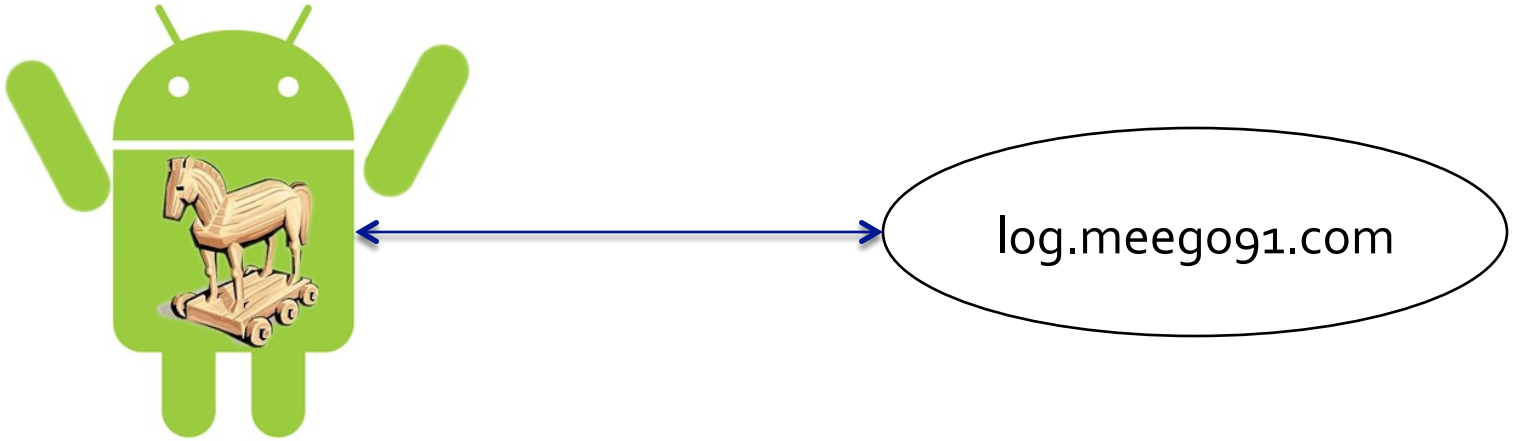
Use any method possible to increase # of installations

Escalate Privileges



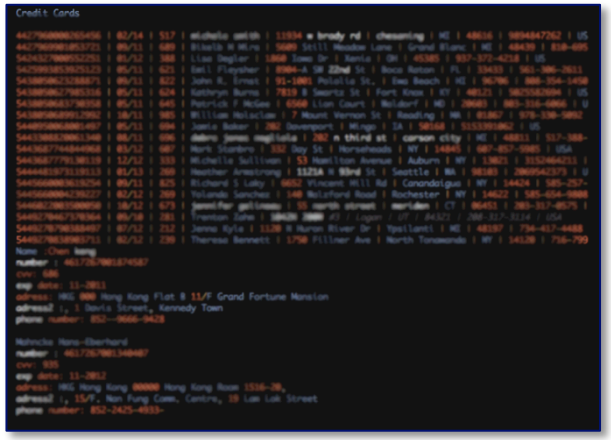
Obtain the ability to collect valuable information

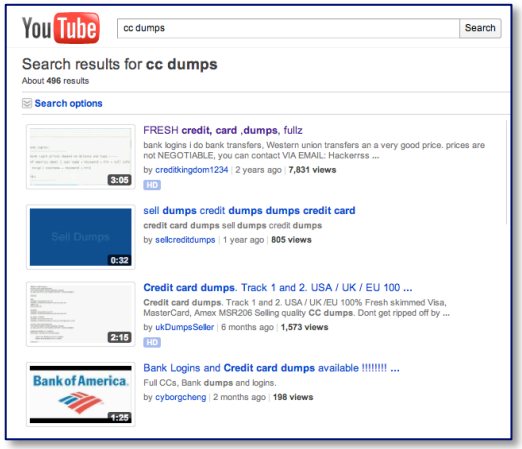
Establish Command & Control

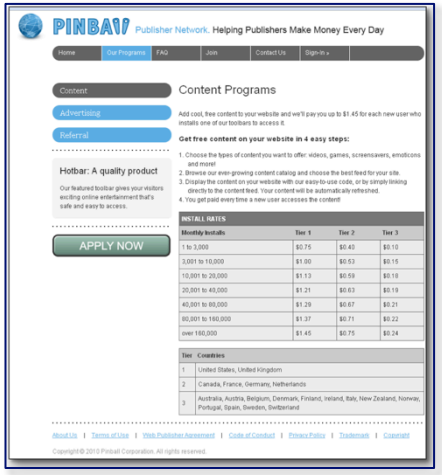


Simple C&C corresponds with simplistic options for network monitoring

Perform Actions on Objectives







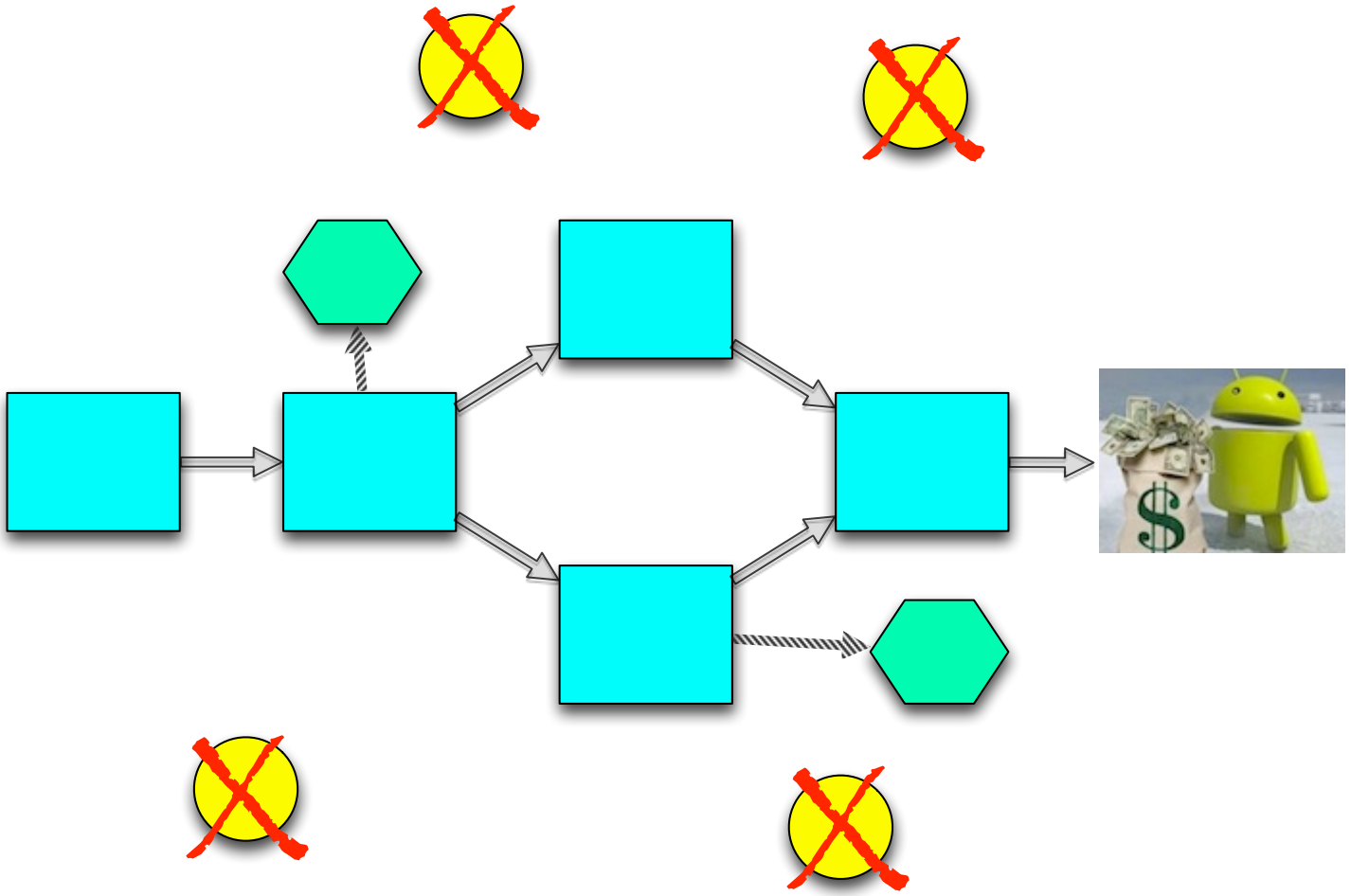
Abuse collected data

Leads to Cyber Pompeii

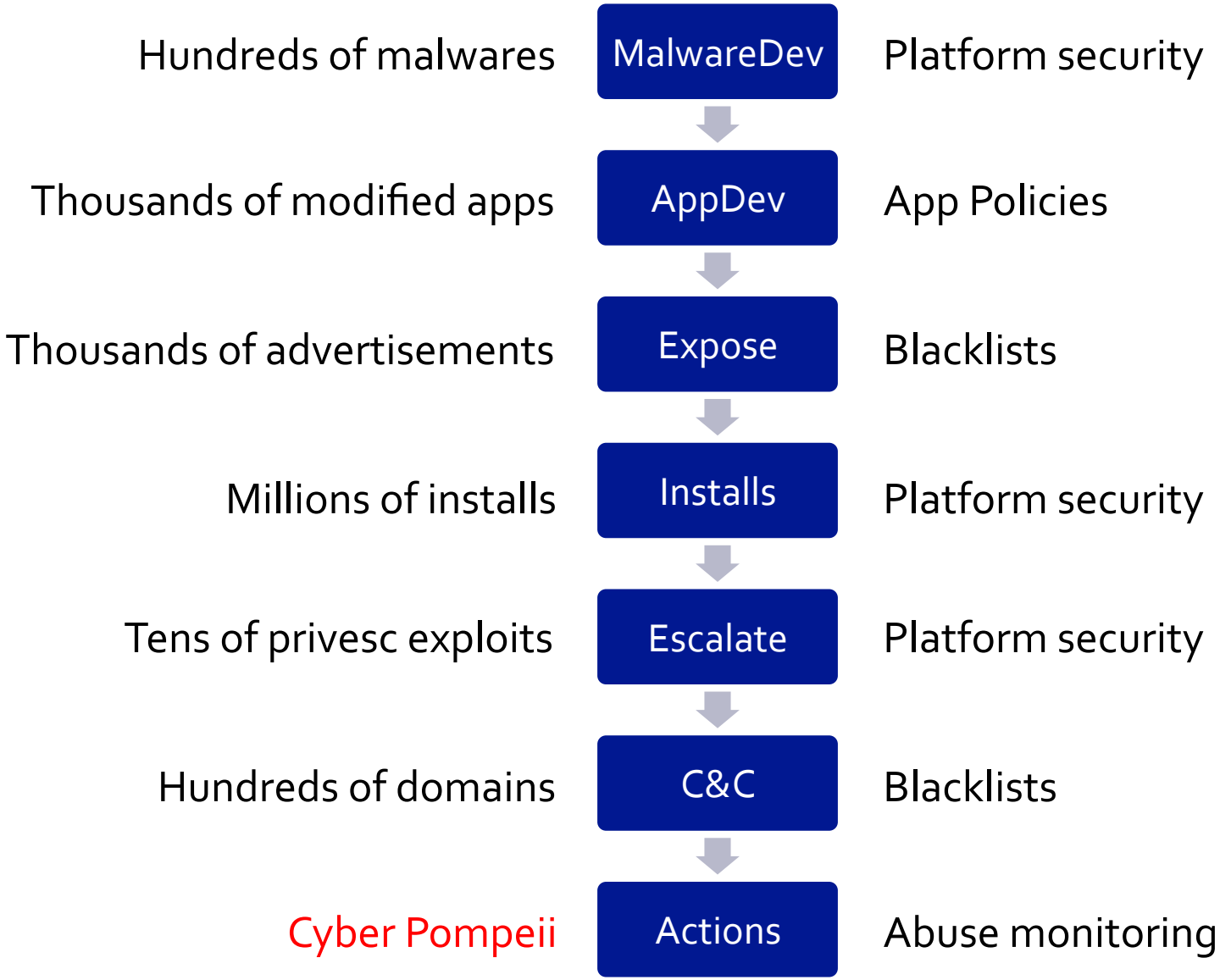
Intrusion Kill Chains

- Systematic process that an intrusion must follow
  - Deficiency in one step will disrupt the process
- Evolves response beyond point of compromise
  - Prevents myopic focus on vulnerabilities or malware
  - Identifies attacker reuse of tools and infrastructure
- Guides our analysis and implementation of defenses
  - Align defenses to specific processes an attacker takes
  - Force attackers to make difficult strategic adjustments

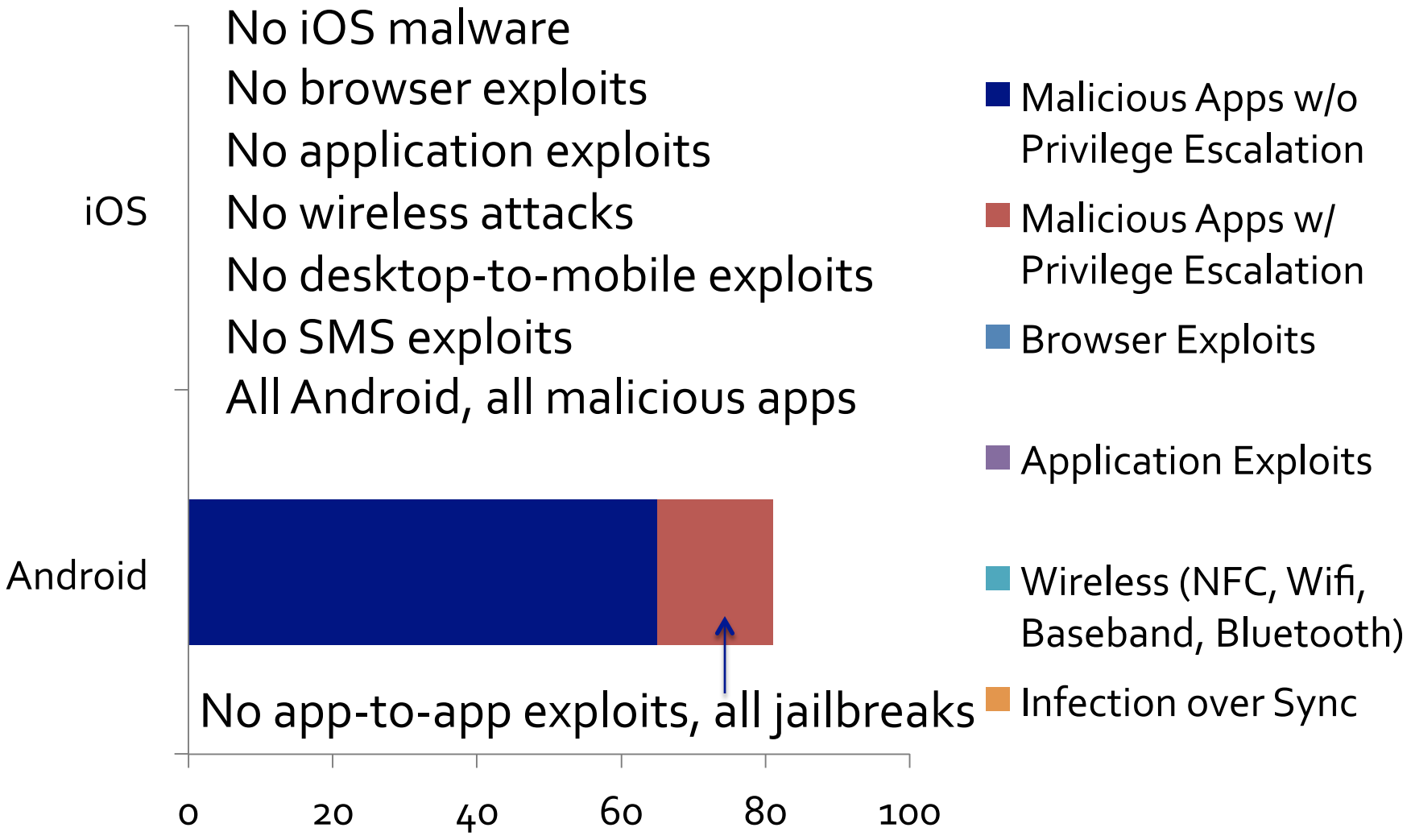
Kill Chains = Scalable Attack Strategies



Mobile Kill Chain Overview

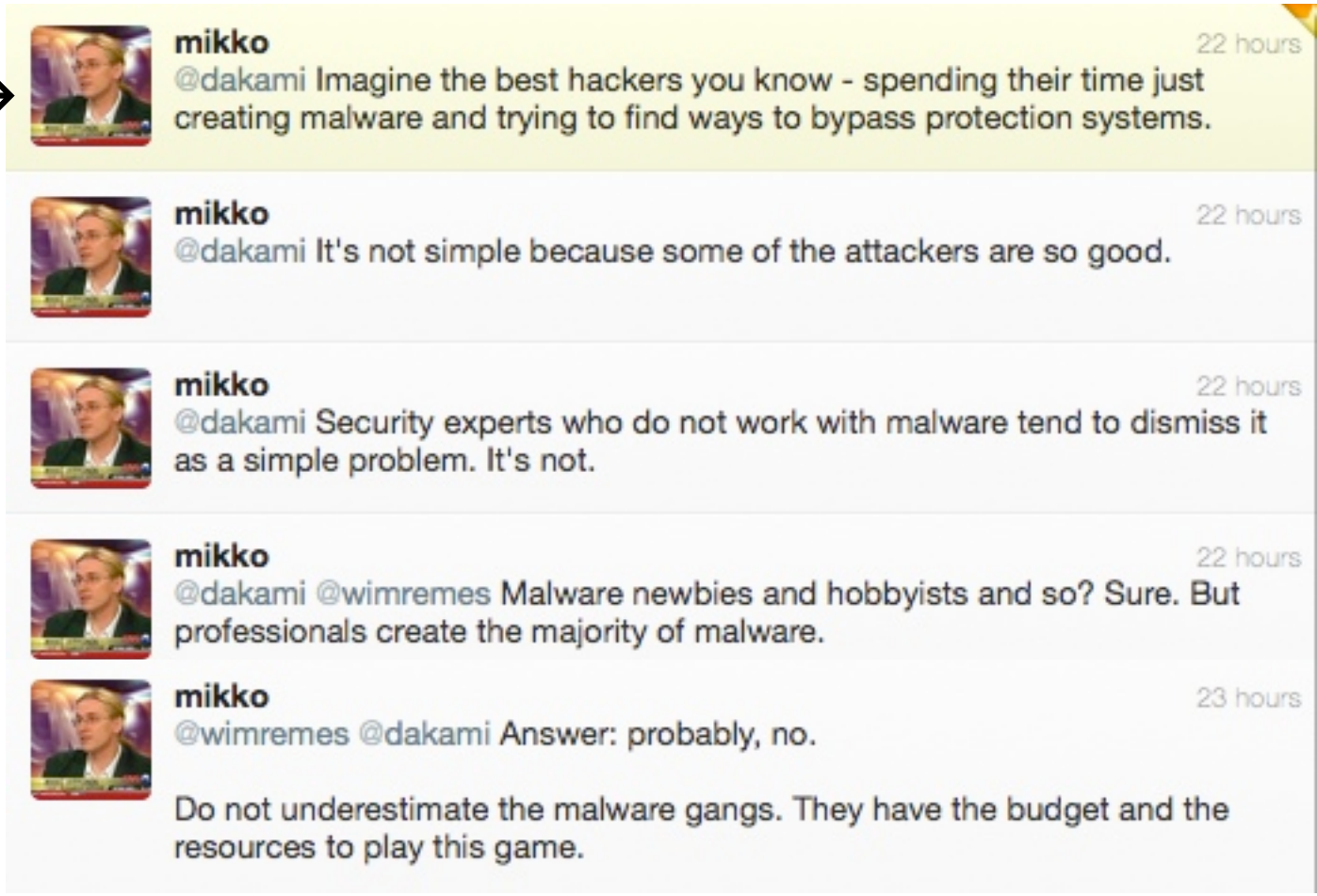


Malware per Platform & Vector



Economics in Practice

Not how it works →



Discrepancies

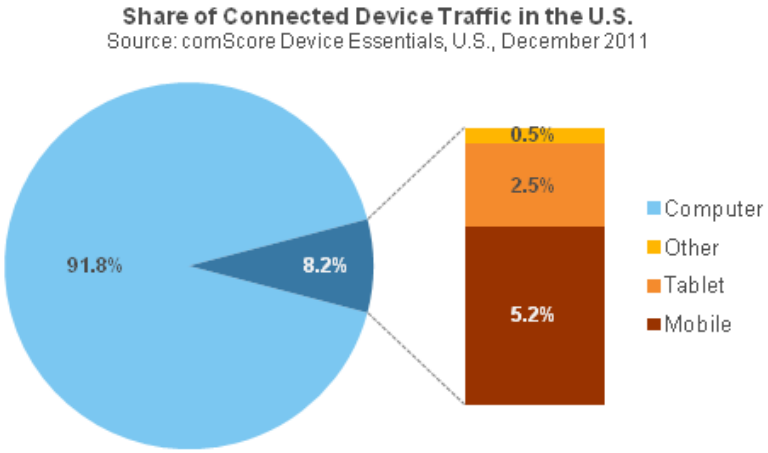
- Is the security industry lying to us?
  - Assumptions that mobile threat == desktop threat
  - Fascination with new attack vectors
  - Myopic focus on ease of attack and malware
- We have no idea how attackers actually work
  - Always more possibilities than probable attacks
  - Attacker economics are different on mobile
- Use economics and adversarial characterization!
  - Why don't we / why won't we see certain attacks?

Where are Mobile Drive-Bys?

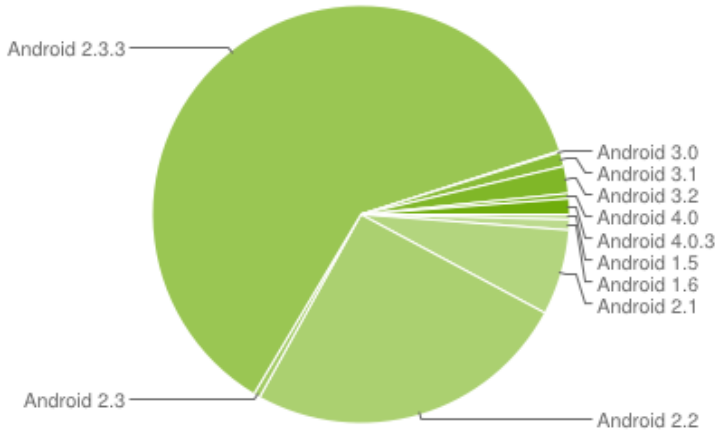
Mobile Town

Desktop City

Mobile Web Browsing



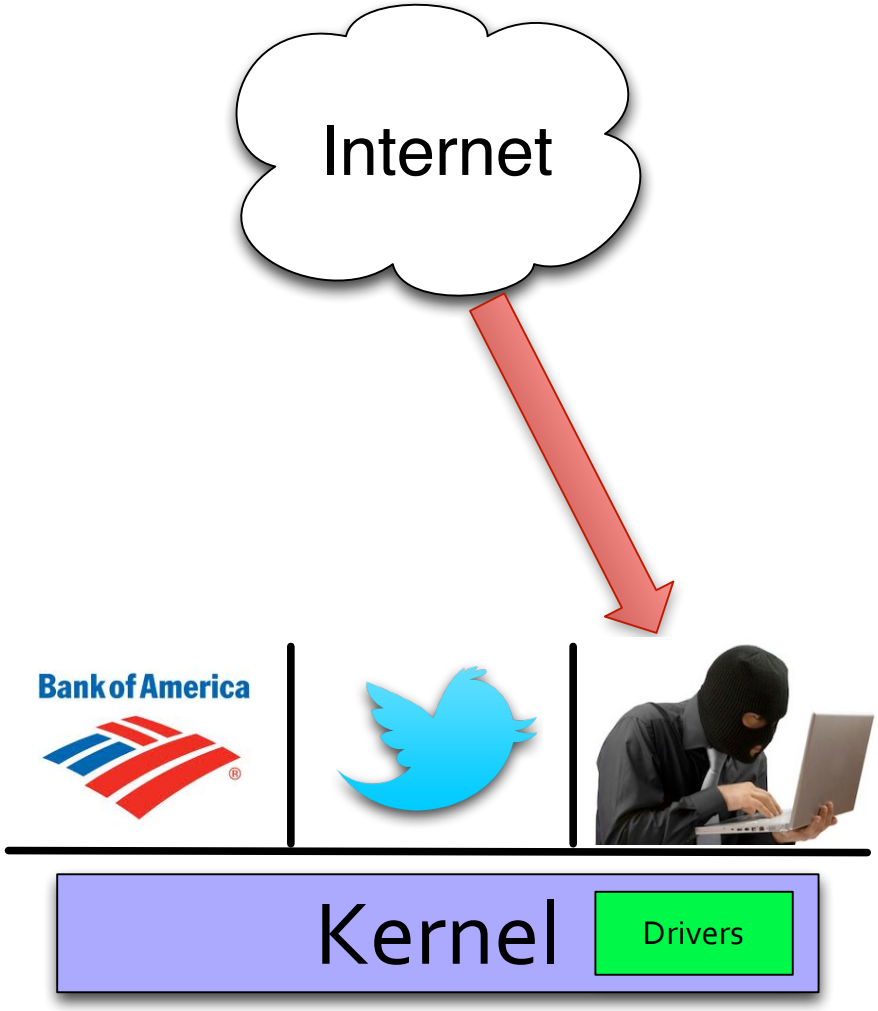
~8% of total web traffic comes from mobile devices



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

Mobile websites might not have any ads!

Browser Exploit Walkthrough



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

Mobile Drive-by Takeaway

- 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 potential revenue rises dramatically

Why attack individual Apps when even the browser isn't viable yet?

Vendor App Stores

Incentives	Browser Exploits	Malicious Apps
# of Targets	Minimal	All Devices (300 mil+)
Ability to Target	Ads	App Store SEO, Lures
Ease of Exploit	Multiple Exploits	Single Exploit
Enforcement	Anonymous	Anonymous?

App stores look like a great value proposition!

App Submission Process

Process	iOS App Store	Google Play
Sign Up	Minimal Cost	Minimal Cost
Identification	Identify Verified	Anonymity Acceptable
App Review	Static Analysis	Dynamic Analysis
Platform Characteristics	No runtime modification	Runtime modification

Apple knows who you are and has seen all your code

Malicious App Submission Process

- In order to submit a malicious iOS app:
  1. Create a believable false identity or risk arrest
  2. Pass a manual content review for originality
  3. Package your sophisticated exploit for Apple's review

- 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

Effectiveness of Bouncer is limited because of Android platform characteristics

Exceptions to Apple Review

- Self-modifying code is difficult, but possible on iOS
  - Since it's disallowed, it sticks out in the review process
  - No matter what: APPLE KNOWS WHO YOU ARE
- Let's look at what happened with Charlie Miller
  - App taken down, removed from phones, Charlie banned
  - If he did anything malicious, he could have gone to jail
- From another angle: Why wasn't it immediately abused?
  - Real-world verification trumps all technical attacks
  - Apple patched in *4 days*, reducing potential revenue
  - Charlie didn't discuss until after patch, no PoC code

Apple bans *identities*, not applications

Malicious App Campaigns

Apple App Store

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* (but still only on Android)

Privilege Escalation Exploits

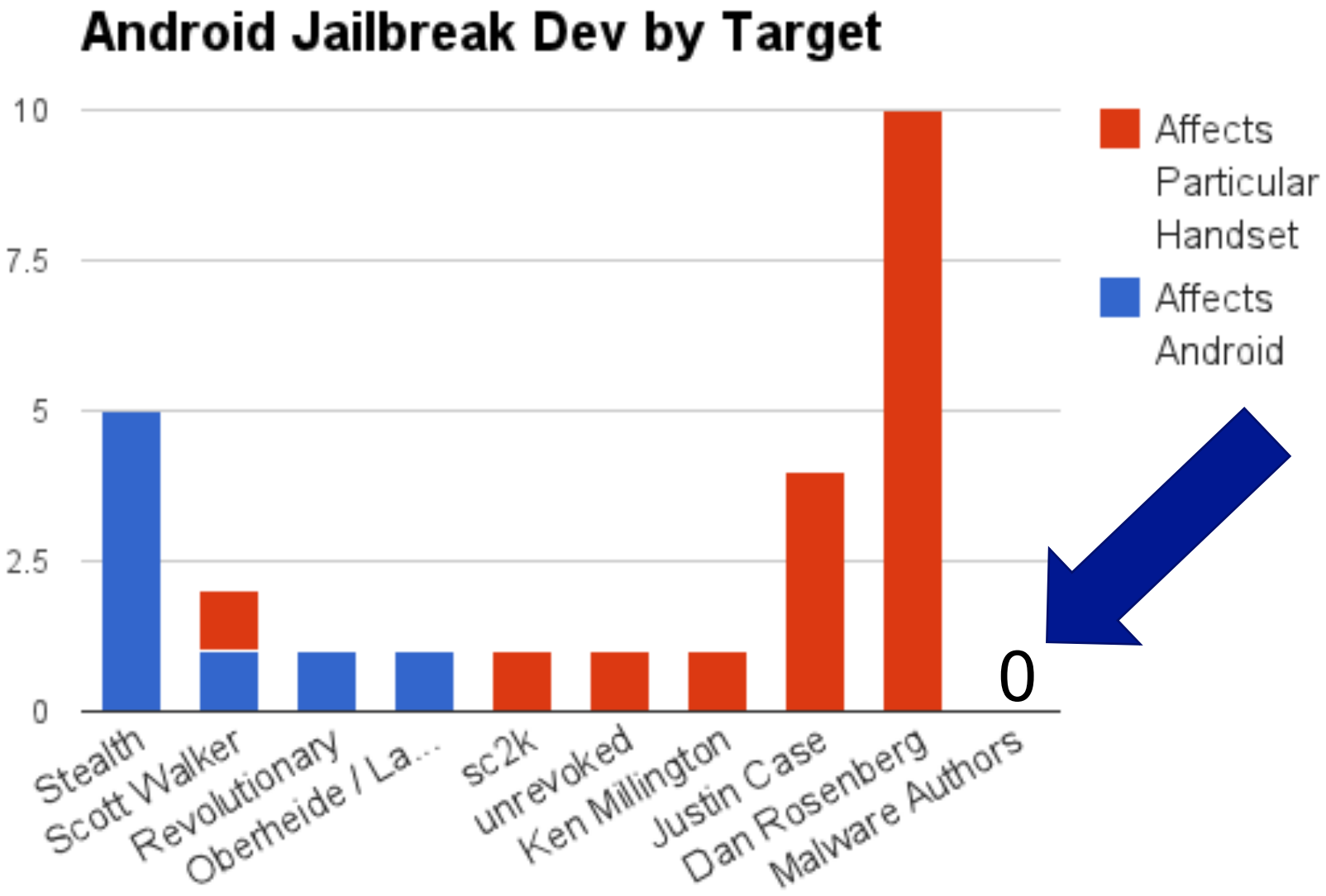
If I just had a jailbreak, then I could make money…

Jailbreak == Free Exploit

- Both platforms have active jailbreaker communities
  - Android: 26 jailbreaks from 10 different authors
  - iOS: 25 jailbreaks from ~4 main groups

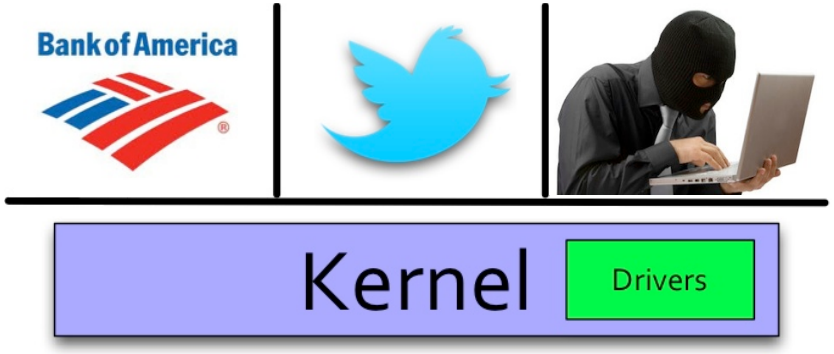
- Jailbreaker behavior mimics attacker behavior
  - Want cheapest possible jailbreak & most possible use
  - Choose target attack surfaces for maximum return
    - Boot ROM (unpatchable) vs. iOS (quickly patchable)

"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)
http://goo.gl/azzOh



What we want to know:
Which exploits get used by malware and why?

Android Escalation Scenarios



I got you to install my app
Now what?

Scenario	Cost of Attack	Value of Data	# of Targets
Universal JB	Free	High (all data)	High (all)
Request SMS	Free	High (2FA)	Medium (some)
Handset-specific	Limited Availability	High (all data)	Limited
App-to-App	Limited Availability	Low (limited data)	Limited

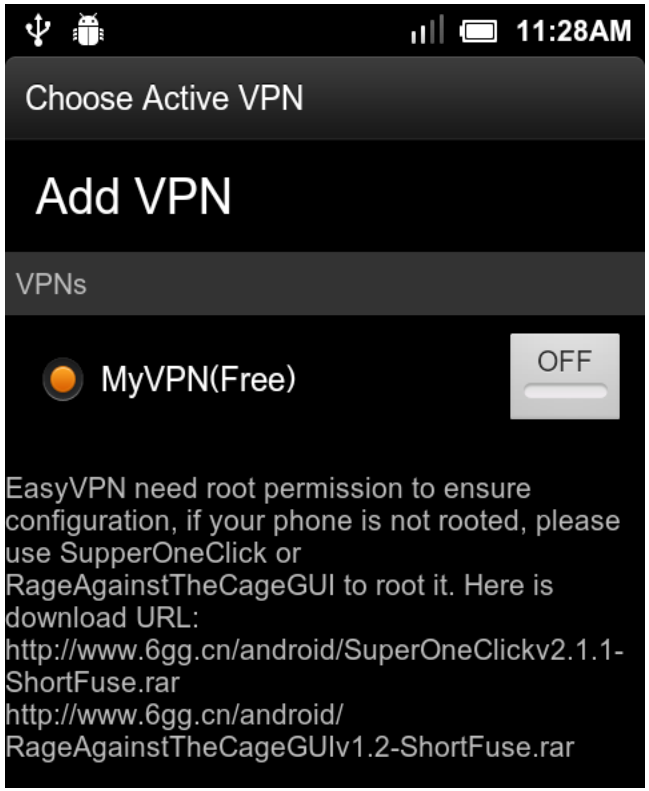
App-to-App and Handset-specific exploits have similar incentives: neither are used

Universal Android Exploits

Exploit Name	Last Affected Version	Abused?
Exploid	2.1 (Éclair)	Malware
RageAgainstTheCage	2.2.1 (Froyo)	Malware
Zimperlich	2.2.1 (Froyo)	No
KillingInTheNameOf	2.2.2 (Froyo)	No
Psneuter	2.2.2 (Froyo)	No
GingerBreak	2.3.4 (GingerBread)	Malware
zergRush	2.3.5 (GingerBread)	No
Levitator	2.3.5 (GingerBread)	No (Low # of Devices)
mempodroid	4.0.3 (ICS)	No

Android 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 these techniques observed in-use by actual malware
- They're less effective (user interaction), less used, but still work

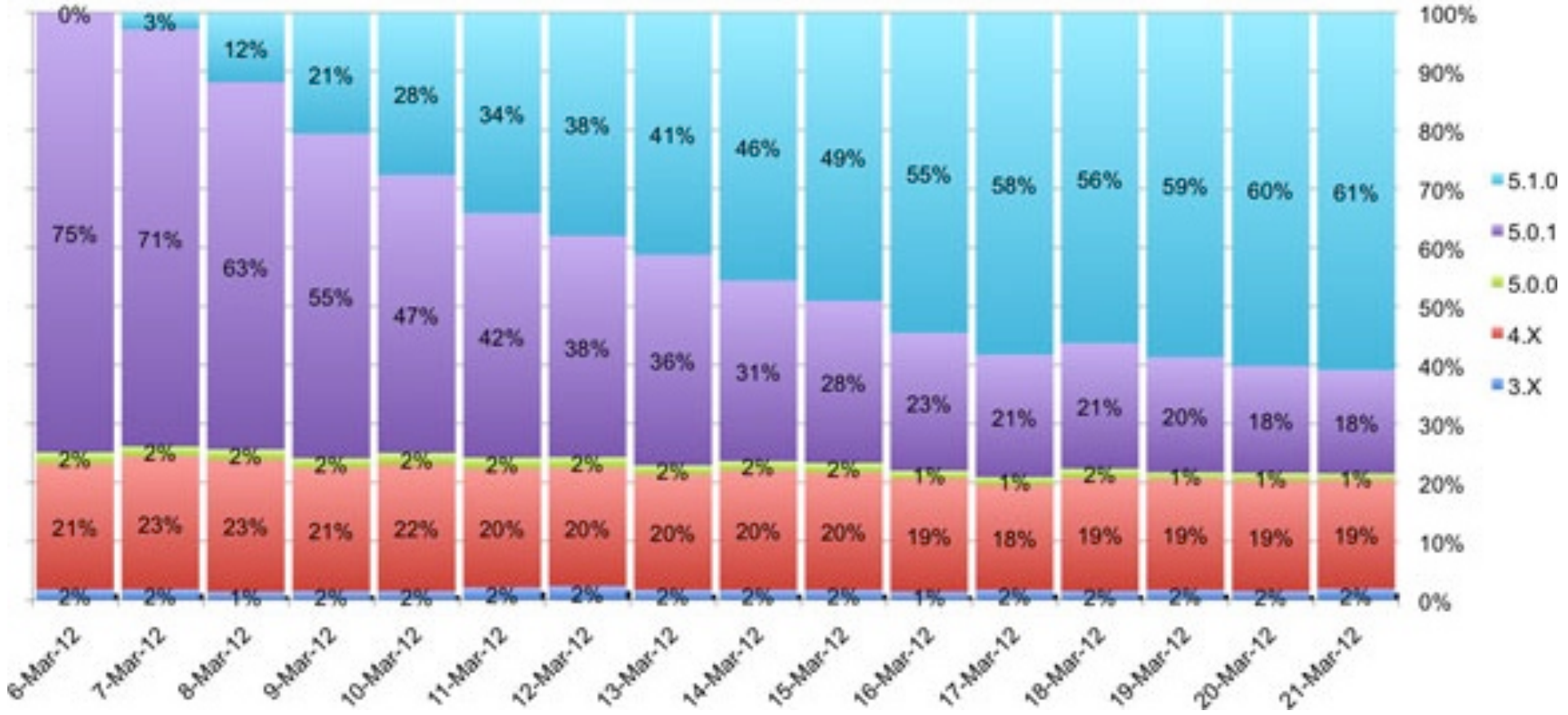
Android Maximizes Potential Revenue

Platform	Codename	03/12/2012	4/18/2012
1.x	Cupcake / Donut	1.2%	1.0%
2.1	Eclair	6.6%	6.0%
2.2	Froyo	25.3%	23.1%
2.3.0 - 2.3.2	Gingerbread	0.5%	0.5%
2.3.3 – 2.3.7	Gingerbread	61.5%	63.2%
3.x	Honeycomb	3.3%	3.3%
4.x	Ice Cream Sandwich	1.6%	2.9%

Android Exploit	Time to Patch 50%
Exploid (2.1)	294 days
RageAgainstTheCage (2.2.1)	> 240 days

http://blog.mylookout.com/blog/2011/08/04/inside-the-android-security-patch-lifecycle/

iOS Limits Potential Revenue



http://david-smith.org/blog/2012/03/10/ios-5-dot-1-upgrade-stats/index.html

iOS Limits Potential Revenue

Exploit	Jailbreak	Patch Availability
Malformed CFF	Star (JailbreakMe 2.0)	10 days
T1 Font Integer Overflow	Saffron (JailbreakMe 3.0)	9 days
mmap Logic Flaw	Charlie Miller	4 days (30d Apple headstart)

Apple quickly patches Jailbreaks that would be useful in malicious attacks

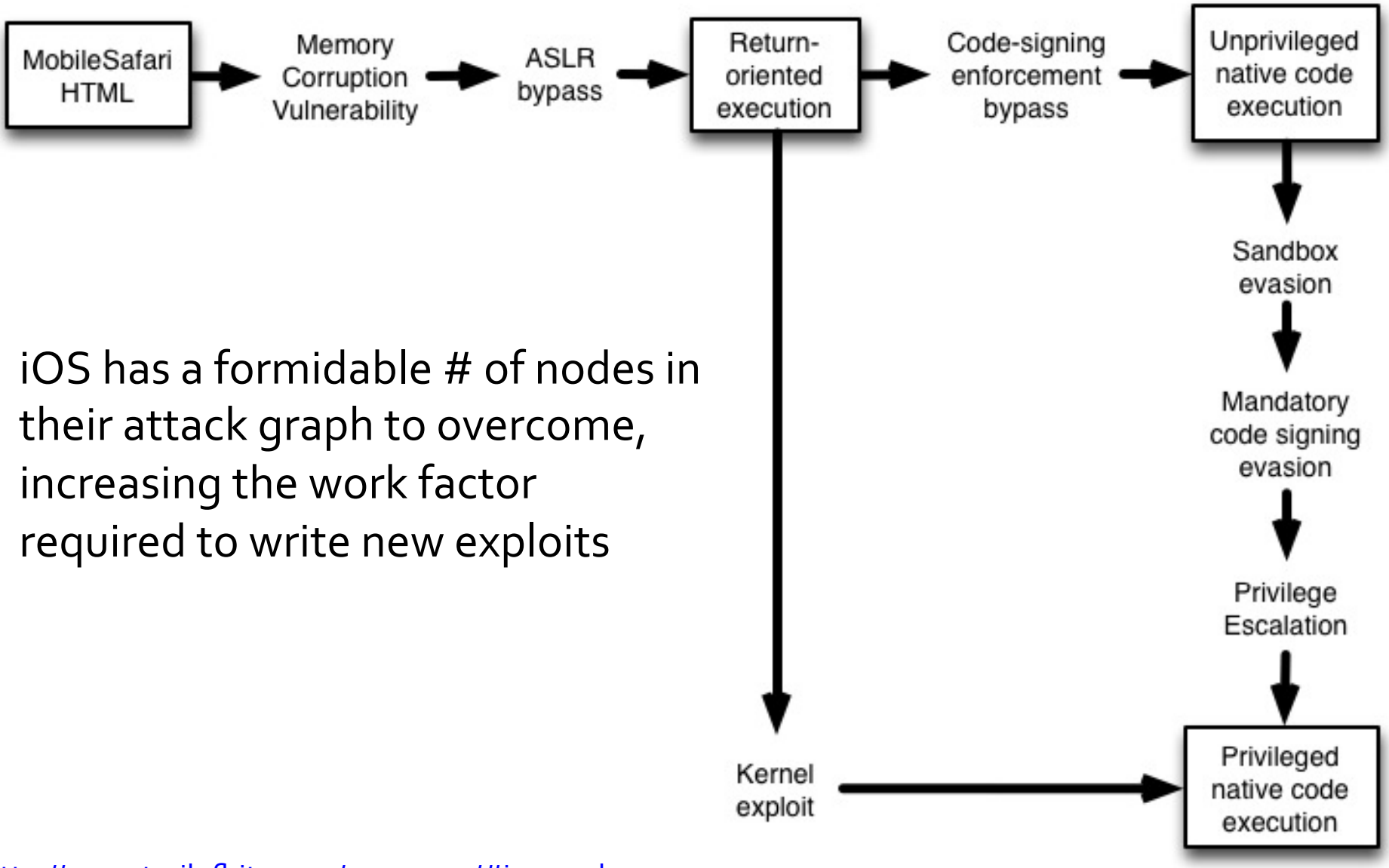
Android Minimizes Cost of Attack

Mitigation	iOS	Android
Code Injection	Code Signing	No-eXecute
Randomization	Strong ASLR	Incomplete ASLR
Sandbox	Seatbelt	None
Patch Information	Available	Not Available

- Code Signing is significantly stronger than NX (Partial vs Full ROP)
- Android app permissions have no effect on privilege escalation exploits
- Google does not track exploited vulnerabilities (CVEs) on Android

http://jon.oberheide.org/blog/2012/02/27/aslr-in-android-ice-cream-sandwich-4-0/

iOS Maximizes Cost of Attack



iOS has a formidable # of nodes in their attack graph to overcome, increasing the work factor required to write new exploits

http://www.trailofbits.com/resources/#ios-eval

Privilege Escalation Takeaways

- Malware authors have no ability to write exploits
  - The only exploits abused are public jailbreak exploits
- Google has not done anything to address jailbreaks
  - No attempt to mitigate them in the OS via sandbox
  - They don't track vulnerabilities that allow them
  - Platform is filled with alternate escalation scenarios
- Android patches have no effect on problem
  - Google has no ability to force carriers / OEMs to react
  - Even if they could, it's too easy to write new exploits

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

Effective Responses

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 handsets will use it?
- ASLR in Ice Cream Sandwich 4.x
  - Little to no effect on privilege escalations
  - Useful to make browser exploits difficult
  - Can't help 300+ million existing devices

Google is ahead of threats that don't exist yet, but far behind on ones that do

App Development Strategies

- Not all Keychain APIs are created equal
  - Keychain only in Android 4.0+ (2.9% of users)
  - Android only stores keys. No keygen, no data storage.
  - Successful jailbreaks means total exposure
  - Contrast with HW-backed iOS Data Protection API*

- Limit accessible data and implement a circuit breaker
  - Apps shouldn't request an entire DB of content
  - Circuit Breaker: cut off access after a threshold
  - Mobile users should only download data that mobile devices can actually read

* http://www.trailofbits.com/resources/#ios-eval

Enterprise BYOD Strategies

- Mobile groupware must follow app security strategy
  - Limit accessible data, implement a circuit breaker
  - Ask your vendor these questions!

- Assume that BYOD devices are compromised
  - Less likely on iOS, a certainty on Android
  - Existing jailbreak detection is fallible
    - Malicious attackers aren't connecting to Cydia

- If Android users can install their own apps, they will be compromised by accident
  - Restrict access to internal App Catalogue if possible

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 to be compromised
  - Bouncer, Chrome, ASLR have limited impact
  - No mitigations to slow jailbreaks, no ability to react
- iOS will steer clear of similar attacks for now
  - Real-world verification trumps all the technical attacks
  - Mitigations slow jailbreaking, reacting quick reduces value

References

- Attacker Math 101, Dino Dai Zovi
  - www.trailofbits.com/research/#attackermath
- iOS Security Evaluation, Dino Dai Zovi
  - www.trailofbits.com/research/#ios-eval
- Exploit Intelligence Project, Dan Guido
  - www.trailofbits.com/research/#eip
- 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/F8BjI
- 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/

Leftovers

Mobile Ads

Google admob

Anonymous	ID Verified	Anonymous
Low Cost	$300k min	$50 min
Scriptable	HTML5	Img / Text
High Traffic	Low interest from legit advertisers	

Close Access

- Insecure Storage, NFC, Bluetooth, Wifi, 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, Wifi 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 or fraudulent data collection through close access