Android Security and the Elusive HSM

Mobile Digital Wallet Security Summit

Jon Oberheide
Duo Security
jono@duosecurity.com

Introduction

- Jon Oberheide
- CTO, Duo Security

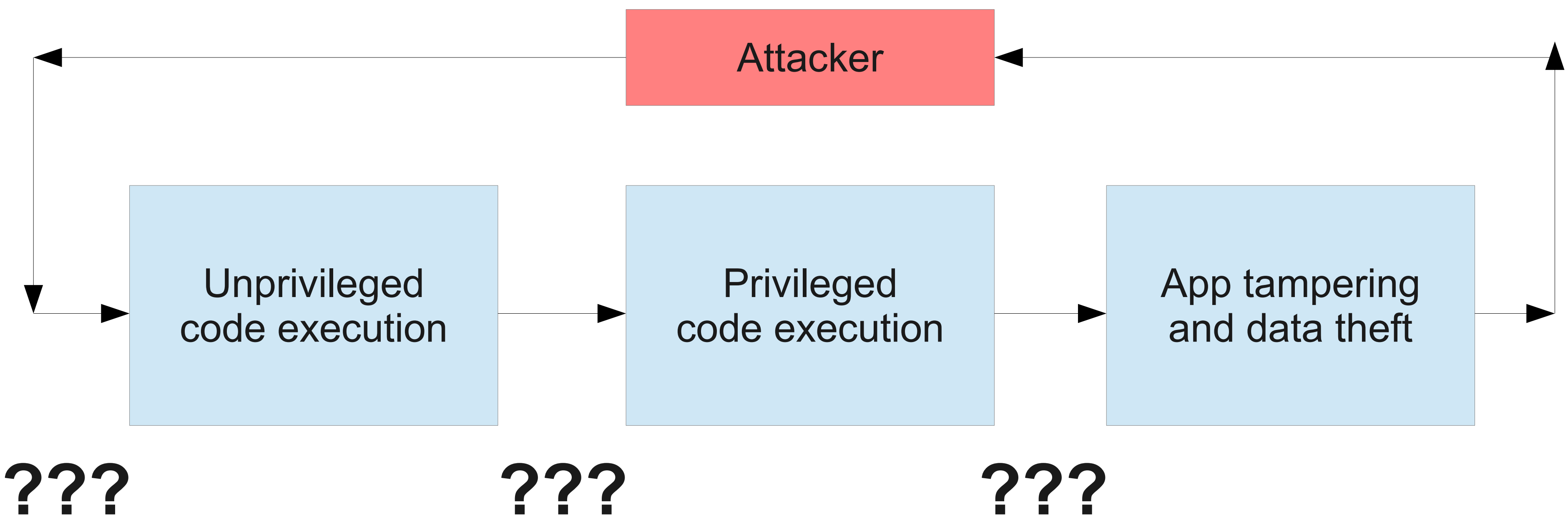
- Today
  - High level look at Android security and how HSMs can help
  - Only 30 minutes!
    Lots of external refs!

Use case

- Mobile platform security is important (surprise!)
- Apps have data to protect (confidentiality/integrity)
- Examples
  - Duo Mobile 2FA app
  - Visa Mobile app? V.me?

Threat model

- Attacker wants access to sensitive data
  - Stages of attack/capability
  - Points of attack disruption/mitigation



Agenda

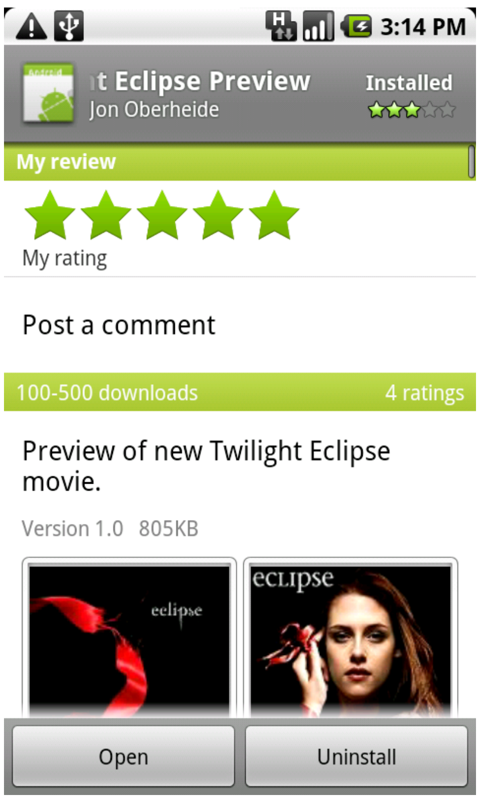
- Introduction
- **Gaining code execution**
- Escalating privileges
- Recent HSM developments
- Wrap-up

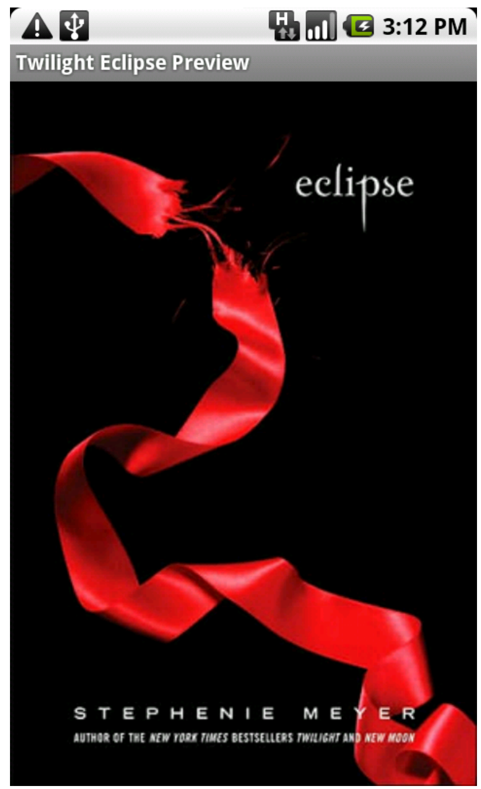
Code execution

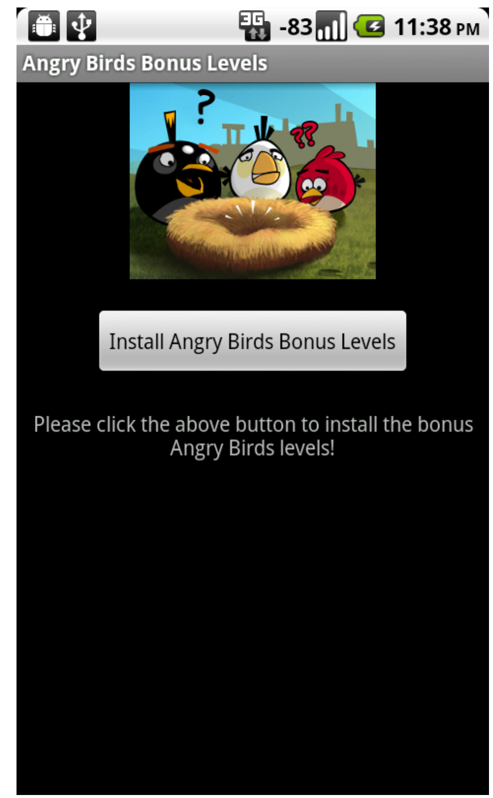
- First step is getting a foothold on the mobile device
  - Need code execution on victim's device
- A couple easy vectors for attackers:
  - Social vector: malicious applications
  - Technical vector: exploitation of existing applications

PoC malicious apps

- In the past, mostly researcher PoCs







Real-world malicious apps

- Nowadays, real-world malware is out there

Security on NBCNEWS.com

New DroidDream malware infects 24

DroidDreamLight Trojan has a

Android malware numbers explode to 25,000 in June 2012

IT administrators await mammoth M...

Rapper Soulja Boy blames Facebo...

malware threats found has hit 25,000. In Ju

10,000, easily the largest find for a month y

Plankton malware drifts into Android Market

Join thousands of others, and sign up for

Android S

FAST FEED

Android Malware Infects 500,000 Users

BY NEAL UNGERLEIDER | AUGUST 20, 2012

Stopping malicious apps

- Mobile antivirus
  - Reactive signature-based detection of malicious apps

Norton
by Symantec

- Mobile malware exists, but these folks tend to push the hype a bit beyond reality

Google's Bouncer

- Google's Bouncer
  - Guards the entry point to the Android Market
  - Dynamically analyze submitted apps and block malicious apps from being published

- Dynamic analysis is <u>hard</u>
- Fingerprinting Bouncer is <u>easy</u>
- Catches some malware, but easy to bypass

http://jon.oberheide.org/files/summercon12-bouncer.pdf

Google's Bouncer

BUSTICATI PRODUCTIONS PRESENTS

DISSECTING THE ANDROID BOUNCER

STARRING

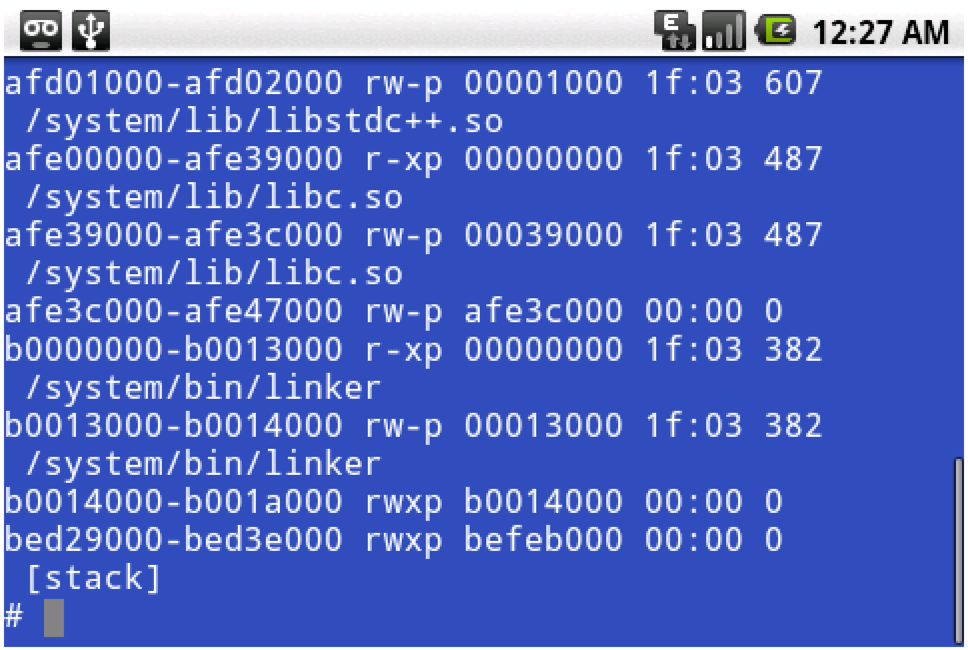
DR. OBERHEIDE and DR. MILLER

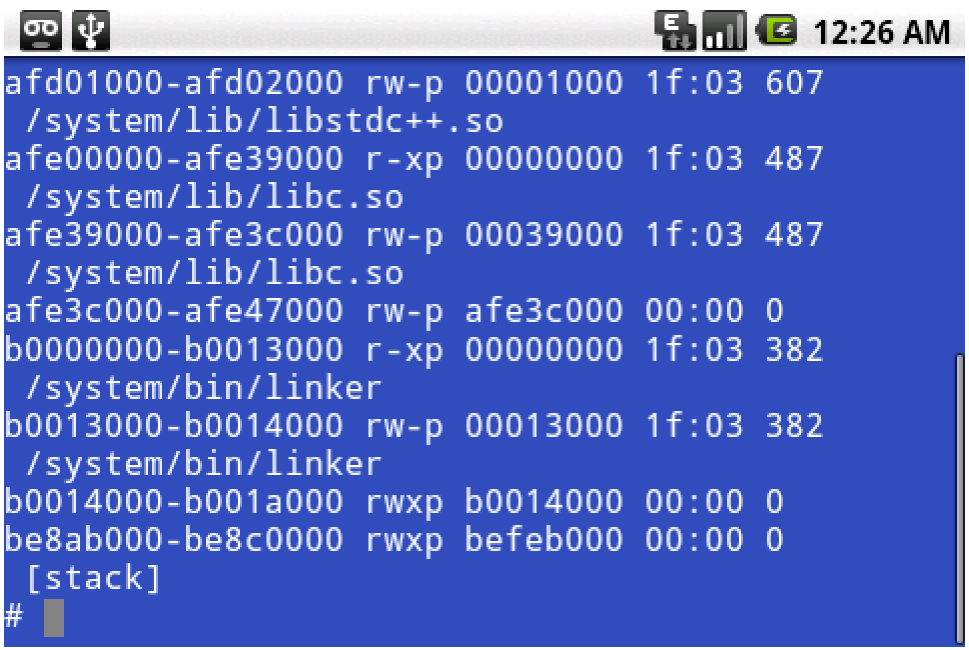
Client-side exploitation

- Threat of client-side applications
  - Large attack surface of native code
  - Traditional memory corruption vulns
  - Similar to desktop client-side threats
- Browser/PDF/Docs = huge attack surface
- Standard Linux hardening mechanisms
  - NX, ASLR, RELRO, BIND_NOW, etc
- Android exploit mitigations are getting better...

Exploit mitigation evolution

- Android exploit mitigations have <u>slowly</u> evolved over the years...
- Before Android 2.3.x, no NX/ASLR:





Exploit mitigation evolution

- Android 2.3.x – Gingerbread
  - Finally got NX support!
  - But still ineffective ASLR:

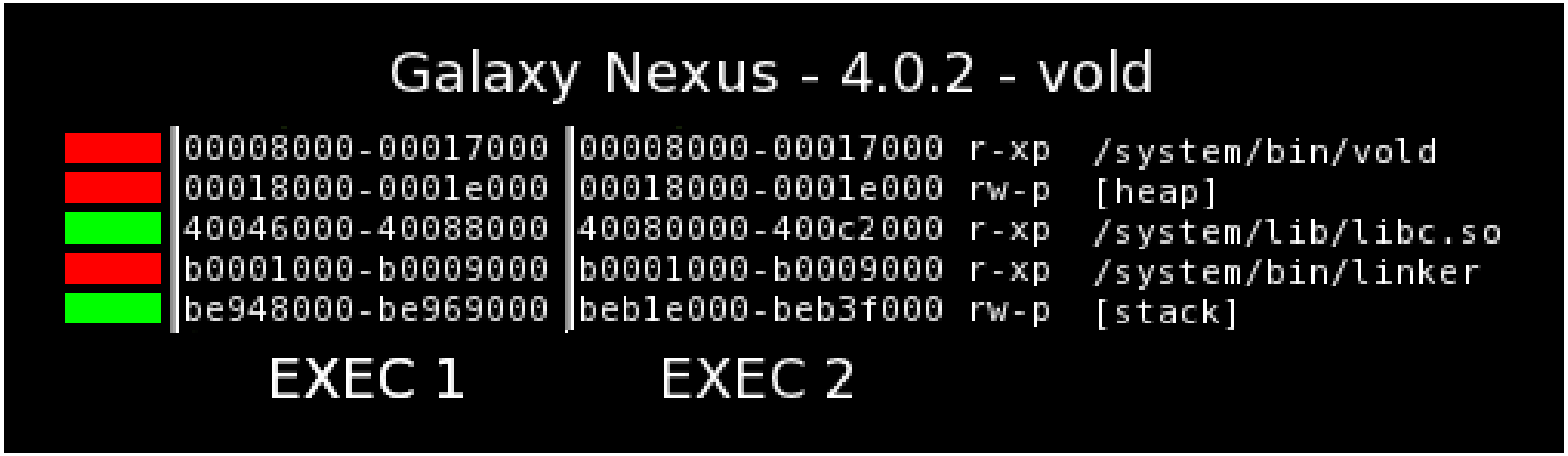
Nexus S - 2.3.4 - vold

```
00008000-00014000 | 00008000-00014000 r-xp  /system/bin/vold
00015000-0001b000 | 00015000-0001b000 rw-p  [heap]
afd00000-afd40000 | afd00000-afd40000 r-xp  /system/lib/libc.so
b0001000-b0009000 | b0001000-b0009000 r-xp  /system/bin/linker
bea31000-bea52000 | beccb000-becec000 rw-p  [stack]
```

EXEC 1 EXEC 2

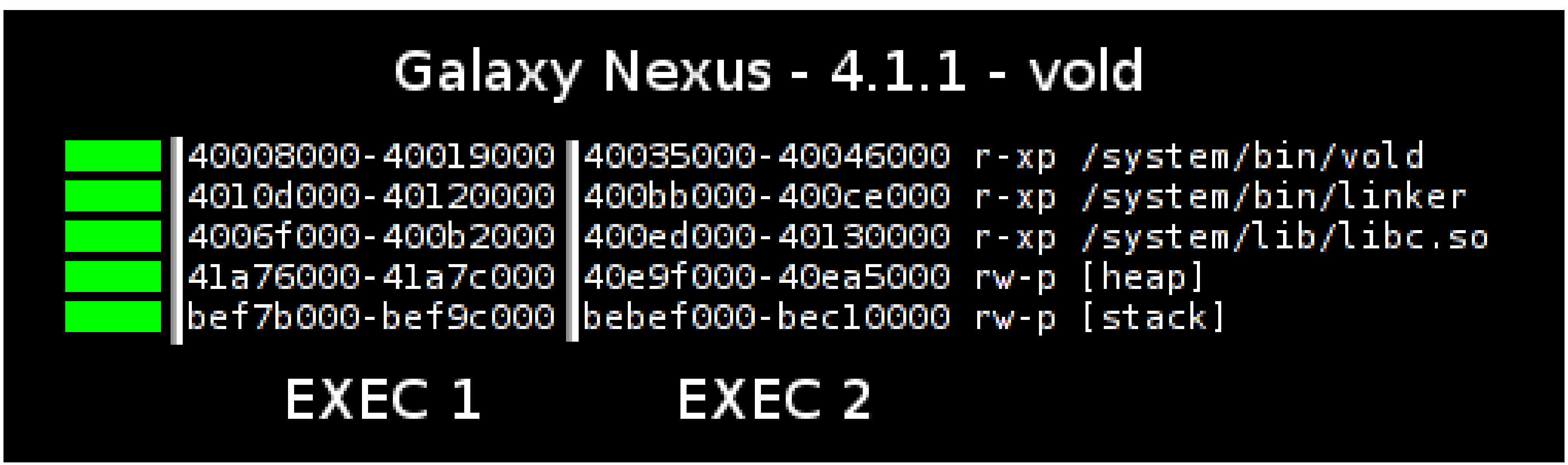
Exploit mitigation evolution

- Android 4.0 – ICS
  - ASLR listed in the release notes as a new security feature!
  - But upon deeper inspection...



https://blog.duosecurity.com/2012/02/a-look-at-aslr-in-android-ice-cream-sandwich-4-0/

- Android 4.1 – Jelly Bean
  - Ok, this time we have ASLR for real...



https://blog.duosecurity.com/2012/07/exploit-mitigations-in-android-jelly-bean-4-1/

Code execution wrap-up

- First goal of attacker is getting a foot hold on the device with code execution
- Either by compromising an existing app or tricking user into installing a malicious app
  - Some strides made in exploit mitigations
  - More general problem of malicious apps is hard

Best to assume that malicious code/apps will be present on the user's device!

Agenda

- Introduction
- Gaining code execution
- **Escalating privileges**
- Recent HSM developments
- Wrap-up

So what happens now?

- With code execution the attacker can:
  - Change the behavior of the exploited app
  - Steal data used by the exploited app
- The attacker can't:
  - Affected other apps on the device
  - Steal data from other apps
- Thanks to the Android "sandbox"

Android app “sandboxing”

- Calling it a sandbox is a stretch
- Each application gets a unique uid/gid upon install

```
drwxr-xr-x    1 10027    10027        2048 Nov
9 01:59 org.dyndns.devesh.flashlight
drwxr-xr-x    1 10046    10046        2048 Dec
8 07:18 org.freedictionary
drwxr-xr-x    1 10054    10054        2048 Feb
5 14:19 org.inodes.gus.scummvm
drwxr-xr-x    1 10039    10039        2048 Mar
8 12:32 org.oberheide.org.brickdroid
```

Escalating privileges

- What does the privileged attack surface look like on an Android device?
  - Entire vanilla Linux kernel
  - + custom kernel modifications by Google
  - + custom drivers by third-party devs
  - Privileged system daemons (vold, etc)
  - Poorly written setuid binaries

Bottom line: <u>Lots</u> of attack surface to exploit!

Overview of privesc vulns

- Some vulns affect nearly all Android devices:

Name	Component	Notes
ASHMEM	kernel	custom Google mod
Exploid	init daemon	netlink source check
Gingerbreak	vold daemon	netlink source check
Levitator	kernel device driver	third-party kernel mod
Mempodroid	kernel	affected vanilla kernel
RageAgainstTheCage	adb daemon	setuid(2) return value
Wunderbar	kernel	affected vanilla kernel
ZergRush	libsysutils	memory corruption
Zimperlich	zygote	setuid(2) return value

http://jon.oberheide.org/files/bsides11-dontrootrobots.pdf

Levitator exploit

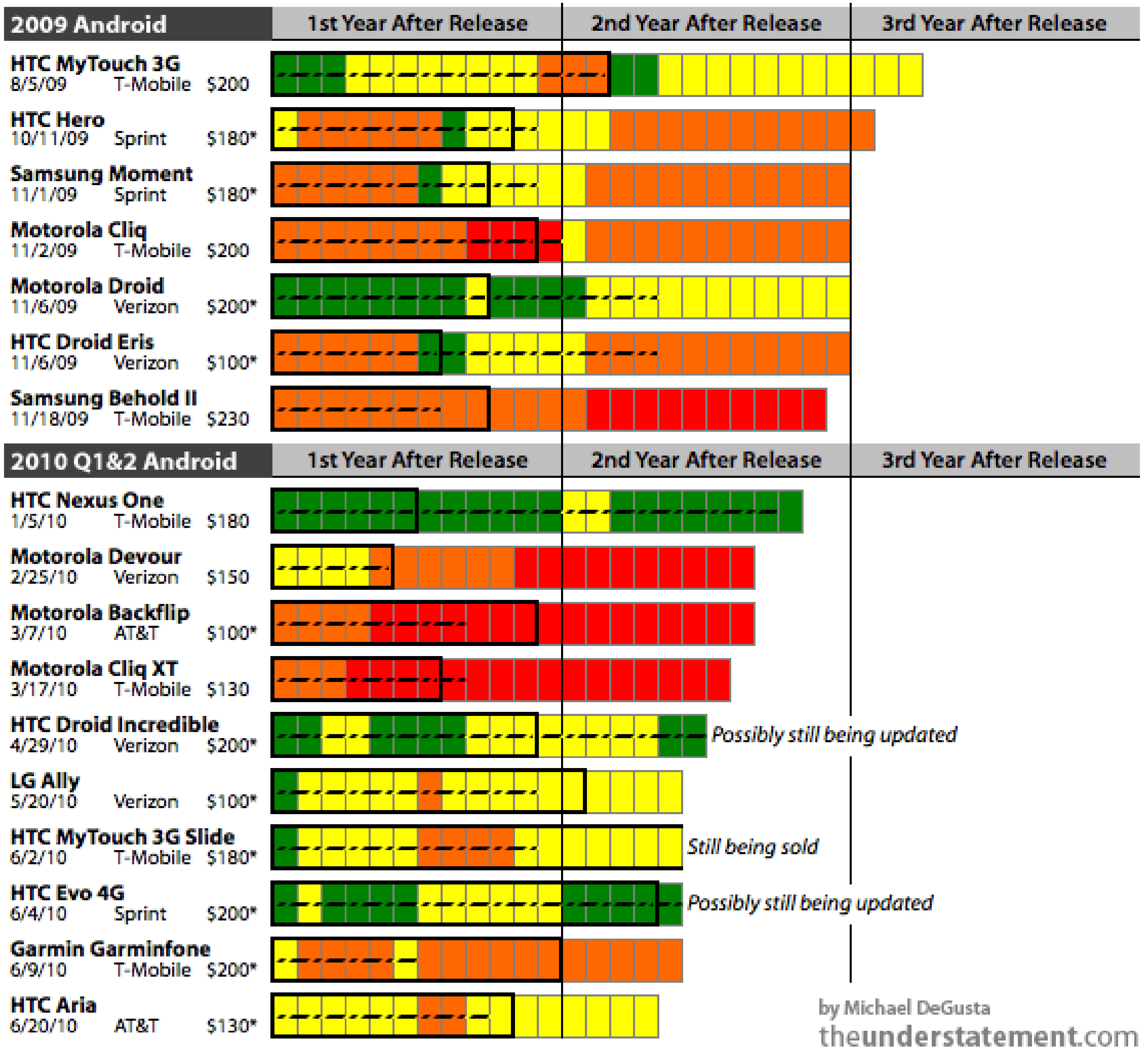
- Levitator exploit
  - Targeted PowerVR vulnerability: /dev/pvrsrvkm
  - Allowed arbitrary kmem read/write
  - Affected popular S series devices
  - Patched in 2.3.6 after 10+ months
- Chain of custody?
  - Researcher → Google → Samsung → Imagination Tech → Manufacturers → Carriers

http://jon.oberheide.org/files/levitator.c

Carrier patching problem

- Carriers are <u>terrible</u> at patching
  - Slow, conservative patch practices
  - Inverted user/economic incentives
  - 6+ months typical vulnerability window
- One of the biggest causes of mobile insecurity
  - Carrier's tight grip on control
  - Complex ecosystem of software responsibility
  - Third-parties have no opportunity to intervene

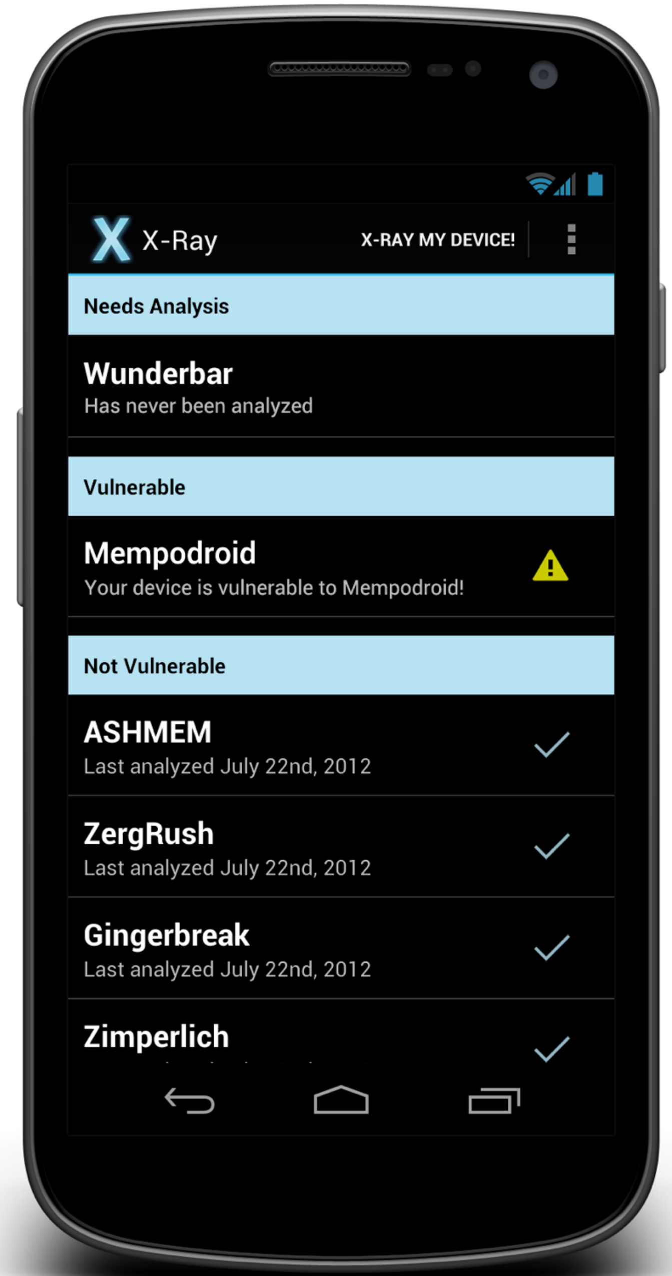
Carrier patching problem



Vulnerability assessment on mobile

- We can't patch the vulns*, can we at least enumerate?
- X-Ray app
  - Vulnerability assessment for Android devices
  - Launched just weeks ago
- Full stats coming next month at United Summit

http://xray.io



A privileged attacker

- If attacker escalates privileges, it's game over, right?
  - Can break out of “sandbox”
  - Tamper with applications
  - Sensitive data can be accessed/stolen
- How can we maintain security guarantees given this threat model?
  - Generally speaking, we can't!
  - But for some use cases...

Agenda

- Introduction
- Gaining code execution
- Escalating privileges
- **Recent HSM developments**
- Wrap-up

How a HSM helps

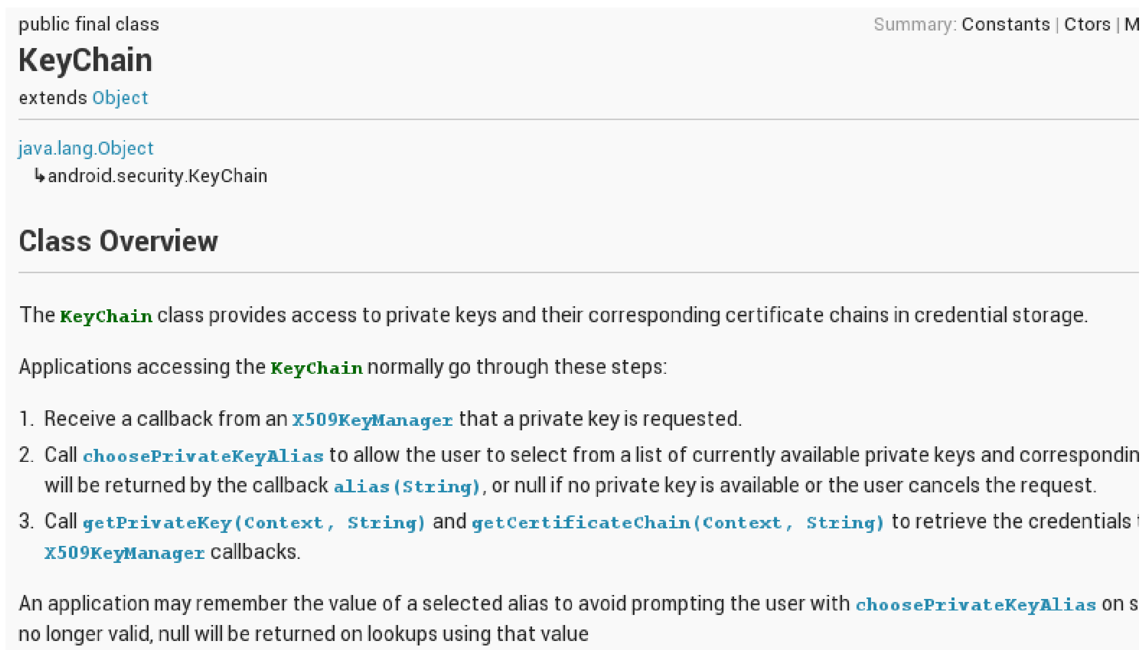
- If we're dealing with privileged attacker, assume that all system memory is compromised
  - So, key material must be kept out of memory
- HSM can provide:
  - Smartcard-style crypto engine
  - Hardware-backed tamper-proof credential storage
  - Key generation, signature computation

Past approaches

- SIM-based
  - mSign, etc
- SOC approahces
  - OMAP M-Shield, ARM TrustZones, etc
- Android-specific initatives
  - Google Wallet, SEEK, etc

Recent Android developments

- Keychain API released in Android 4.0
  - Primitives for credential storage



public final class
KeyChain
extends Object

java.lang.Object
↳android.security.KeyChain

Summary: Constants | Ctors | M

Class Overview

The `KeyChain` class provides access to private keys and their corresponding certificate chains in credential storage.

Applications accessing the `KeyChain` normally go through these steps:

1. Receive a callback from an `X509KeyManager` that a private key is requested.
2. Call `choosePrivateKeyAlias` to allow the user to select from a list of currently available private keys and correspondin will be returned by the callback `alias(String)`, or null if no private key is available or the user cancels the request.
3. Call `getPrivateKey(Context, String)` and `getCertificateChain(Context, String)` to retrieve the credentials `X509KeyManager` callbacks.

An application may remember the value of a selected alias to avoid prompting the user with `choosePrivateKeyAlias` on s no longer valid, null will be returned on lookups using that value

http://nelenkov.blogspot.com.es/2011/11/using-ics-keychain-api.html

Recent Android developments

- Further HSM support in Android 4.1
  - keymaster framework implementation
- Galaxy Nexus hardware support
  - OMAP 4, TI M-Shield platform
- Basic HSM crypto operations
  - generate_keypair, import_keypair, sign_data, verify_data, get_keypair_public, delete_keypair, delete_all

http://nelenkov.blogspot.com.es/2012/07/jelly-bean-hardware-backed-credential.html

Changes in threat model

- With a HSM, attacker can no longer extract sensitive data / key material post-exploitation
- Example:
  - Duo Push private key generated on-device
  - Signatures generated within the HSM
  - Key material never leaves the credential store

Not a silver bullet: attacker can still compute signatures.
Need trusted input and trusted display paths.

Why HSM approach?

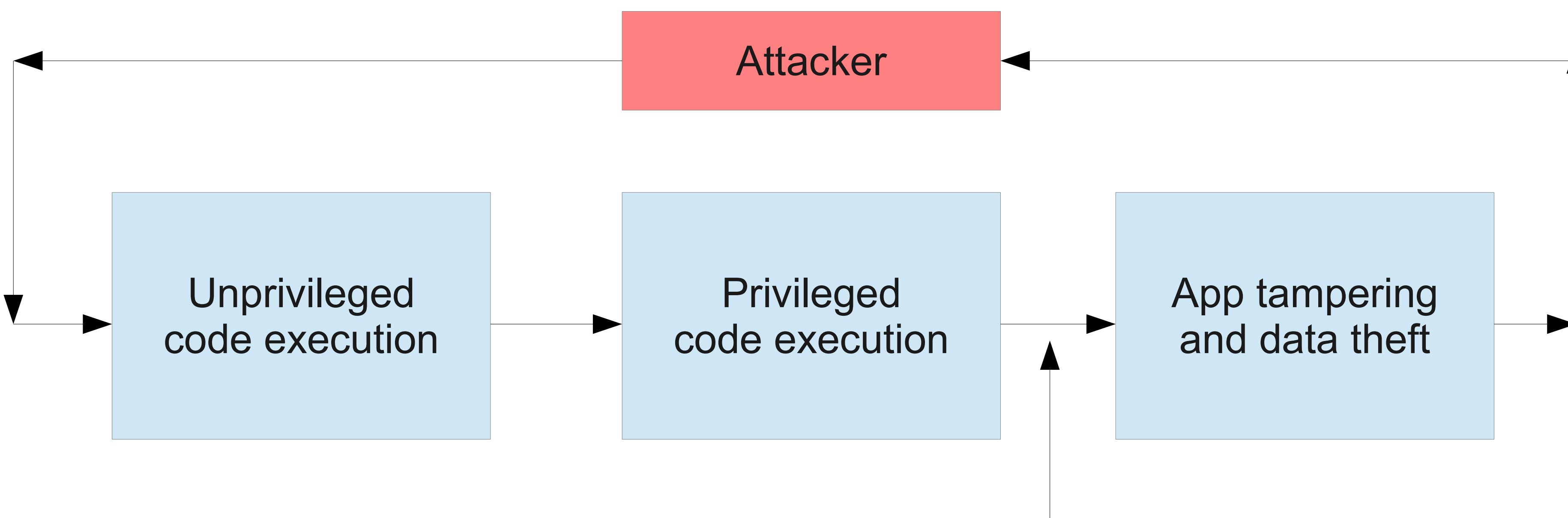
- Need HSM = failure of system security
  - Similar to some virtualization use cases
- Need for third-party availability
- Not the only feasible approach
  - Other avenues to contain/survive exploitation
  - On-device system containers
  - On-device hardware-level virtualization
  - Threshold crypto techniques?

Agenda

- Introduction
- Gaining code execution
- Escalating privileges
- Recent HSM developments
- **Wrap-up**

Wrap-up

- Multiple parties are attacking the mobile security problem at a variety of layers



HSM plays an important role in device security to disrupt the attack chain even if fully compromised.