Beating up on Android

[Practical Android Attacks]

Bas Alberts + Massimiliano Oldani
Immunity Inc.

Who are we?

- Massimiliano Oldani
  - Senior Security Researcher, Immunity, Inc.
  - max@immunityinc.com
- Bas Alberts
  - Senior Security Researcher, Immunity, Inc.
  - bas@immunityinc.com

Why pick on Android?

- Starting to out-sell the iPhone
- Decentralized maintenance
  - Carriers are responsible for updates (slooooow)
- Solid SDK and NDK
- Source Code available (for the most part)
- Free and usable bugs from public repositories
- Familiar (enough) architecture
- Prevent Skynet from getting online

Android Versions

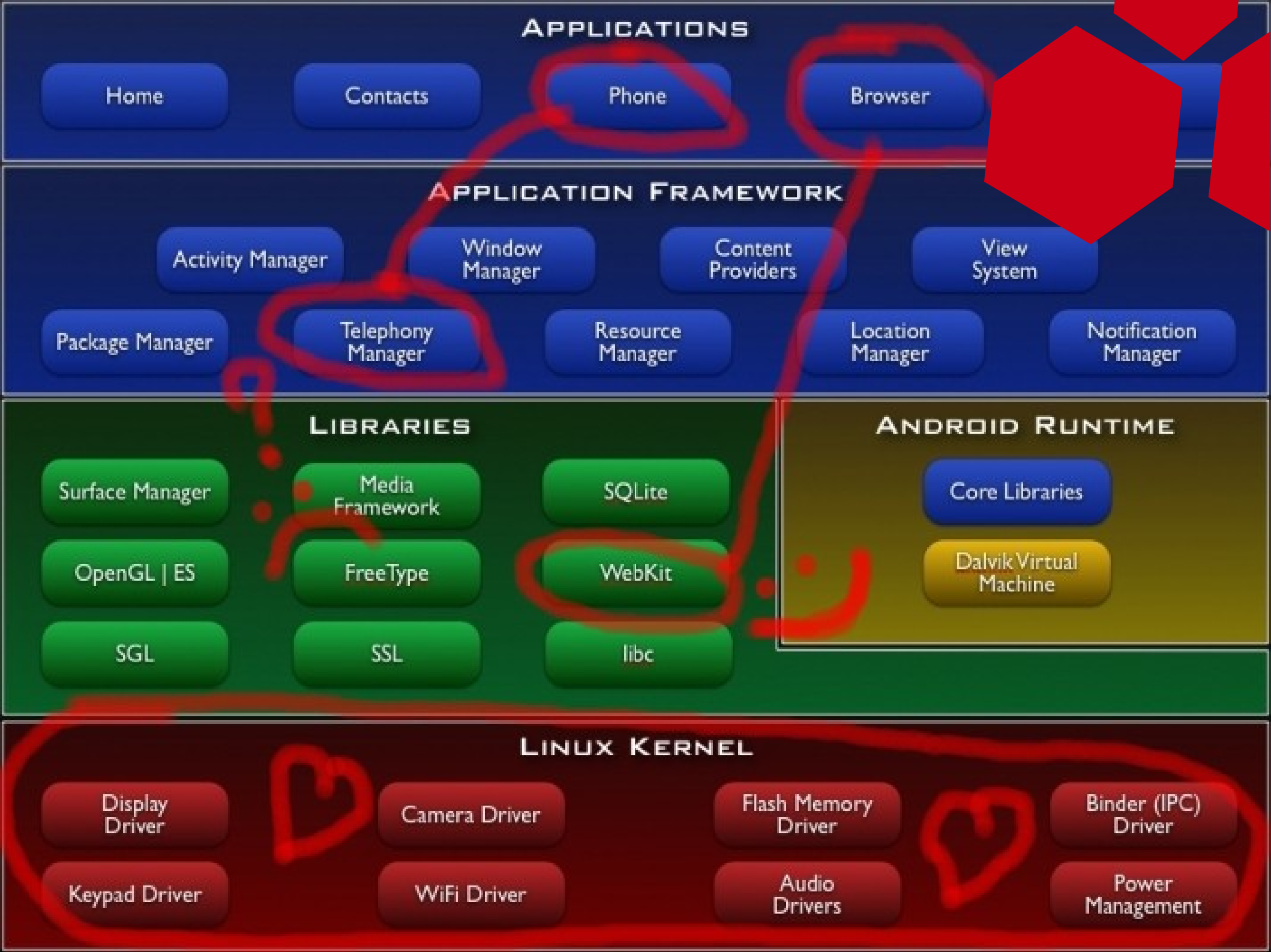
Platform	API Level	Distribution
Android 1.5	3	3.0%
Android 1.6	4	4.8%
Android 2.1	7	29.0%
Android 2.2	8	61.3%
Android 2.3	9	0.7%
Android 2.3.3	10	1.0%
Android 3.0	11	0.2%

http://developer.android.com/resources/dashboard/platform-versions.html

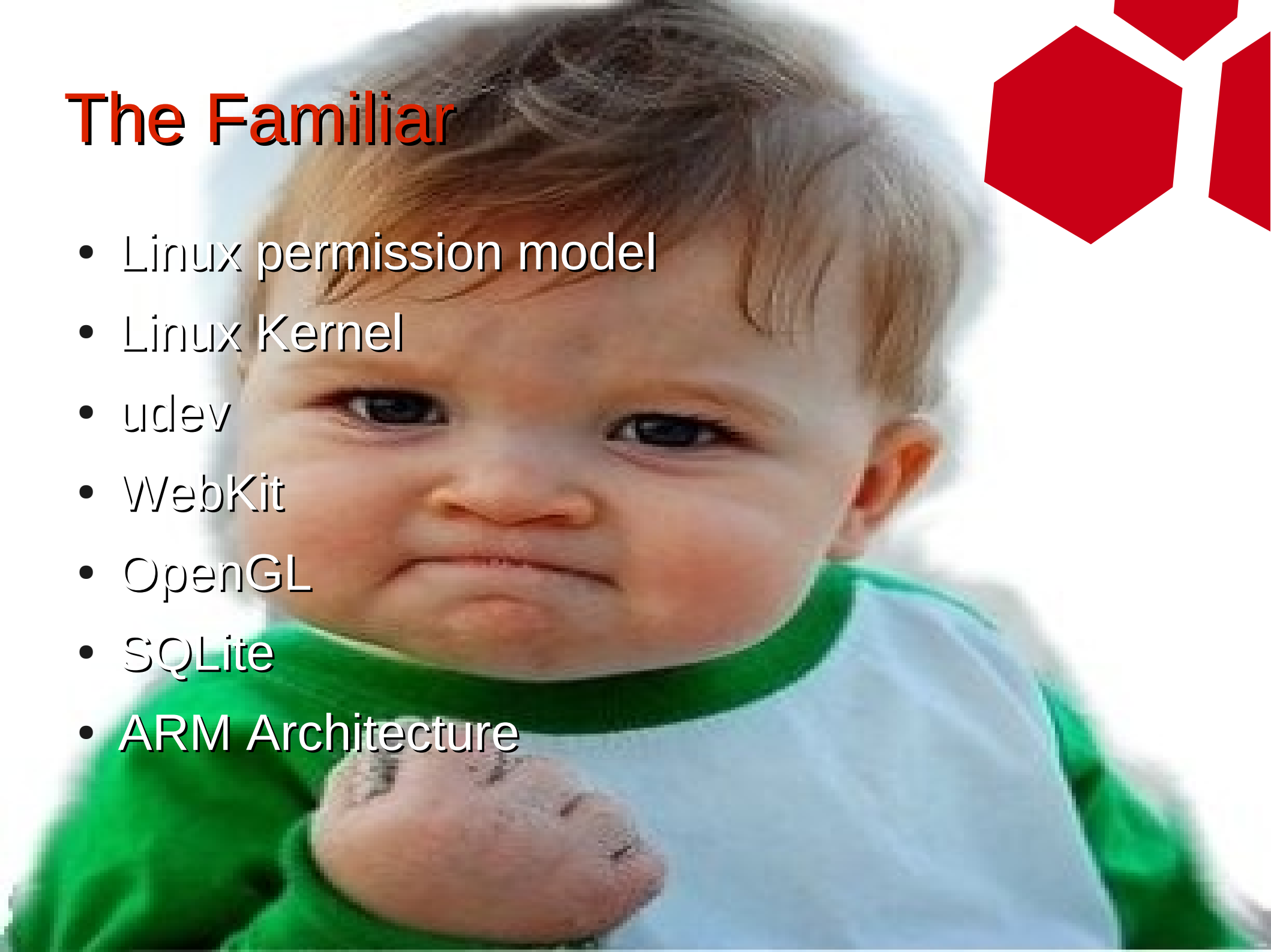

Applications
Home
Contacts
Phone
Browser
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Attack Surface

- Remote: gain access
  - Browser
    - WebKit
  - Phone
    - Telephony stack
- Local: elevate privileges
  - Kernel
    - Device drivers
  - Userland
    - Zygote, ADBd, udev, etc.
- ARM architecture (Linux EABI)


Applications
Home
Contacts
Phone
Browser
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

The Familiar



- Linux permission model
- Linux Kernel
- udev
- WebKit
- OpenGL
- SQLite
- ARM Architecture

The unfamiliar

- Binder IPC
- ADB (Android Debug Bridge)
- Ashmem (Anonymous Shared Memory)
- Vendor specific device drivers
- Android specific device drivers
- Telephony stack
- Bionic libc (not POSIX compliant)
- Custom dynamic linker
- Dalvik VM
- Zygote

It is a Linux, sort of

- Kinda
- Android specific kernel features
  - Binder IPC (drivers/misc/binder.c)
  - Ashmem (mm/ashmem.c)
  - Pmem (drivers/misc/pmem.c)
  - Logger (drivers/misc/logger.c)
  - And much more …
    - http://elinux.org/Android_Kernel_Features

Android Security Model

- Privilege separation
  - Every application has its own uid:gid
  - Distinct system components have their own uid:gid
- Privilege management
  - Zygote process parenting
  - No setuid files
- Application permissions
  - Application Manifest based whitelist
  - Manually accepted by user on install

Hardware protection

- ARM TrustZone
  - Used to provide tamper free data transactions
  - Not used by any Android vendor that we know of
- ARM eXecute-Never (NX bit)
  - Used to enforce memory executable permissions
  - **Not used** up until Android 2.3 (updated: 07/17/2011)
    - Executable stack
    - Executable heap
  - Since 2.3 noexec build flags enabled and enforced on hardware that supports it (e.g. Nexus S)
    - Thanks to Nick Kravelich @ Google for pointing this out

Software protection

- Android randomize_va_space is set to 1
  - 1: Conservative (stack, mmap base, VDSO, PIE) … no heap base (brk) randomization
    - Regardless: Applications are fork()'d from Zygote anyways, and inherit its ASL
  - 2: Full (stack, mmap base, VDSO, PIE, brk)
- Most .so are pre-linked with Apriori (hardcoded load address in an 8 byte "PRE " record at the end of .so) and can not be relocated
  - Ret2libc convenience
- Android's Dynamic Linker does not support runtime relocation
- ASLR: Android Speed Loathes Randomization
  - Google + Stanford: new protection schemes based around rebasing pre-linked libraries during Android device updates
    - http://bojinov.org/professional/wisec2011-mobileaslr-paper.pdf
- DLMalloc based heap with the associated pointer protection schemes
- ProPolice/SSP enabled GCC for native code

Application protection

- Applications can be self signed
  - No Certificate Authority in place to verify application publishers
- Jon Oberheide showed how Google can remotely pull/push apps from/to devices through the GTalkService
  - REMOVE_ASSET Intent
  - INSTALL_ASSET Intent
  - Recent examples include the 50 or so malicious apps that were pulled from the Android market

- http://jon.oberheide.org/blog/2010/06/25/remote-kill-and-install-on-google-android/

Android Sandboxing

- Based completely on privilege separation
  - Enforced by Linux Kernel
- Dalvik VM is NOT a sandbox in itself
  - Any application can run native code
  - That means any application can touch the Kernel directly (syscalls, ioctls, etc.)
- Fine grained Permission/Capability model
  - Per installed Application (Manifest)
  - Per URI (Intent permission flags)

Dalvik … it's not Java

- Applications are written in Java
- Applications are built as Dalvik Bytecode (.dex)
- You don't really care … buuut ...
  - Register based, not stack based
  - Designed specifically for Android architecture
  - Bla bla bla
  - Please don't sue Google for having an optimized JVM
- You do care when auditing Apps
  - dex2jar, smali, dedexer, ...

Android Properties

- The property service
  - Manages a system wide configuration registry not unlike the Windows registry
  - Property key:value string pairs are stored in a shared memory segment: the property space
  - Applications retrieve system critical properties through the property space
    - e.g. ro.secure property decides whether or not the ADB (Android Debug Bridge) daemon should run as root
      - If adbd runs as root (ro.secure == 0), an adb shell drops you to a root prompt and the device is now essentially jailbroken
- *NDK: property_get(), property_set() (libcutils)*
- *Shell: getprop, setprop*

Zygote Process Management

- Zygote is the **Dalvik VM master process** responsible for **starting** and **managing** all subsequent **Dalvik based Components** and their associated **privileges**
  - **Preloads** all the **commonly needed libraries + Dalvik VM** and **fork()'s itself** to instantiate new Application processes
- Listens on a socket for messages that indicate which applications to start and how to start them (frameworks/base/core/java/com/android/internal/os/ZygoteConnection.java)
- Because all Applications are fork()'d from Zygote, they **inherit the same ASL** as Zygote

Application Components

- Activities
  - Present a screen with a user interface
  - Can be shared/started across applications
- Services
  - Backgrounded capability with no user interface
  - Activities can bind to services to interact with them
- Content Providers
  - Manage (stores, retrieves, provides) Application data
  - Data can be on a file system, in an SQLite DB, on the Web, etc.
- Broadcast Receivers
  - Responds to system-wide broadcast announcements (Intents)
  - Screen turned off, Incoming SMS, etc.
- For more detail visit: http://http://developer.android.com/guide/topics/fundamentals.html

Android Jailbreaking

- Legal under the DMCA
  - Large community of people interested in jailbreaking phones (not just the usual suspects from the exploit dev scene)
- Some of the more interesting public attacks against Android come from Sebastian Krahmer of Team 743C (formerly of Team 7350)
- Focus mostly on Android specific components for obtaining local root privileges
  - Exception: Khramer's udev attacks

743C original attacks

- Let's examine some public exploits
  - *KillingInTheNameOf*
  - *Exploid*
  - *RageAgainstTheCage*
  - *Zimperlich*

743C

KillingInTheNameOf

- Affected Android <= 2.2
- Remapped Android property space to writable
  - Vulnerability in Ashmem implementation
  - Any user can remap shared memory belonging to init which contains the property space, with PROT_READ|PROT_WRITE permissions
- Toggled ro.secure property to 0
  - ADB Daemon now runs as root
  - Physical local root through ADB shell

Exploid

- Affected Android <= 2.1
  - As well as regular Linux installs with a vuln udev
- Udev < 1.4.1 did not verify origin of NETLINK udev event messages
  - Sent a NETLINK udev event message that tricked udev into running an arbitrary binary as root when triggering a hotplug event
  - On Android the udev code lives inside of the init daemon which runs as root
- Original bug (CVE-2009-1185) died in 2009 but resurfaced in very similar fashion in the Google udev implementation (updated: 17/07/2011)

RageAgainstTheCage

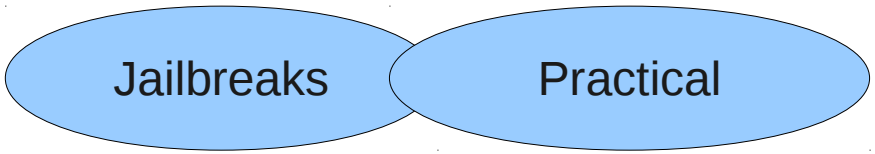
- Affected Android <= 2.2
- Setuid() return values not checked in ADBd
  - ADBd initially runs as root and setuid()'s to uid shell
  - If NPROC resource limit is reached for uid shell, setuid() from uid root to uid shell will fail
  - If setuid() fails, ADBd continues running as root
  - If ADBd runs as root, ADB shell also runs as root
- Fork()'s processes as shell user until NPROC is reached
  - Restart ADBd (bringing uid shell to NPROC-1) and fork() again (as uid shell) right before ADBd (as uid root) tries to setuid() back to uid shell, setuid() fails, Rage wins

ZimperLich

- Affected Android <= 2.2
- Pretty much the same exploit theory as RageAgainstTheCage only this time abusing missing setuid() return value checks in Zygote
- Triggered through Dalvik Application components who's privileges are managed by Zygote with setuid() calls
- More convenient because it doesn't require a uid shell prompt

Practicality of jailbreaks

- Practical jailbreak use for an attacker
  - Physical access required? Not that interesting for an attacker
  - Zygote vs. Adbd



- Our initial access to the device is generally established remotely
  - Through the browser
  - Through a malicious market Application
  - Through an attack against the Telephony Stack
  - Through an attack against the SMS/MMS handling

Establishing access

- The most interesting target by far is the Android Browser
  - Public vulnerabilities available in WebKit
  - Slow to non-existent carrier patch cycles
  - No effective ASLR + executable heap makes remote exploitation reliable
    - ARM: care about I/D cache syncing payload-wise
- Second most interesting target is the Android Market
  - Easy to publish malicious Applications
- Turn that order around if you can obtain INSTALL_ASSET capabilities :)

Elevating privileges

- What can we touch as the browser?
  - The Kernel
  - Privileged System Services through Binder IPC (RPC)
  - Zygote
- The entire **Android local security** model rises and falls with the **(in)security** of the Kernel
  - Audit Focus … from easy to hard
    - **Vendor specific** Linux Kernel components
    - **Android specific** Linux Kernel components
    - Mainline Linux Kernel
      - **No 0day needed** mostly … just porting efforts of **public vulns**
      - **Stale enough to bypass mmap_min_addr for NULL deref?**

Hacking Dave … demo …

- Using a public WebKit vulnerability (CVE-2010-1807)
- Using a private vendor specific bug that we can use from the context of the browser to pop root
  - Sorry we do not kill bugs ... but rest easy it only affects Android 2.1 with a very vendor specific configuration
  - Already plenty of jailbreaks out there for 2.1, but we needed one that worked practically from the context of the browser
- Target: up-to-date T-Mobile 3G Android phone

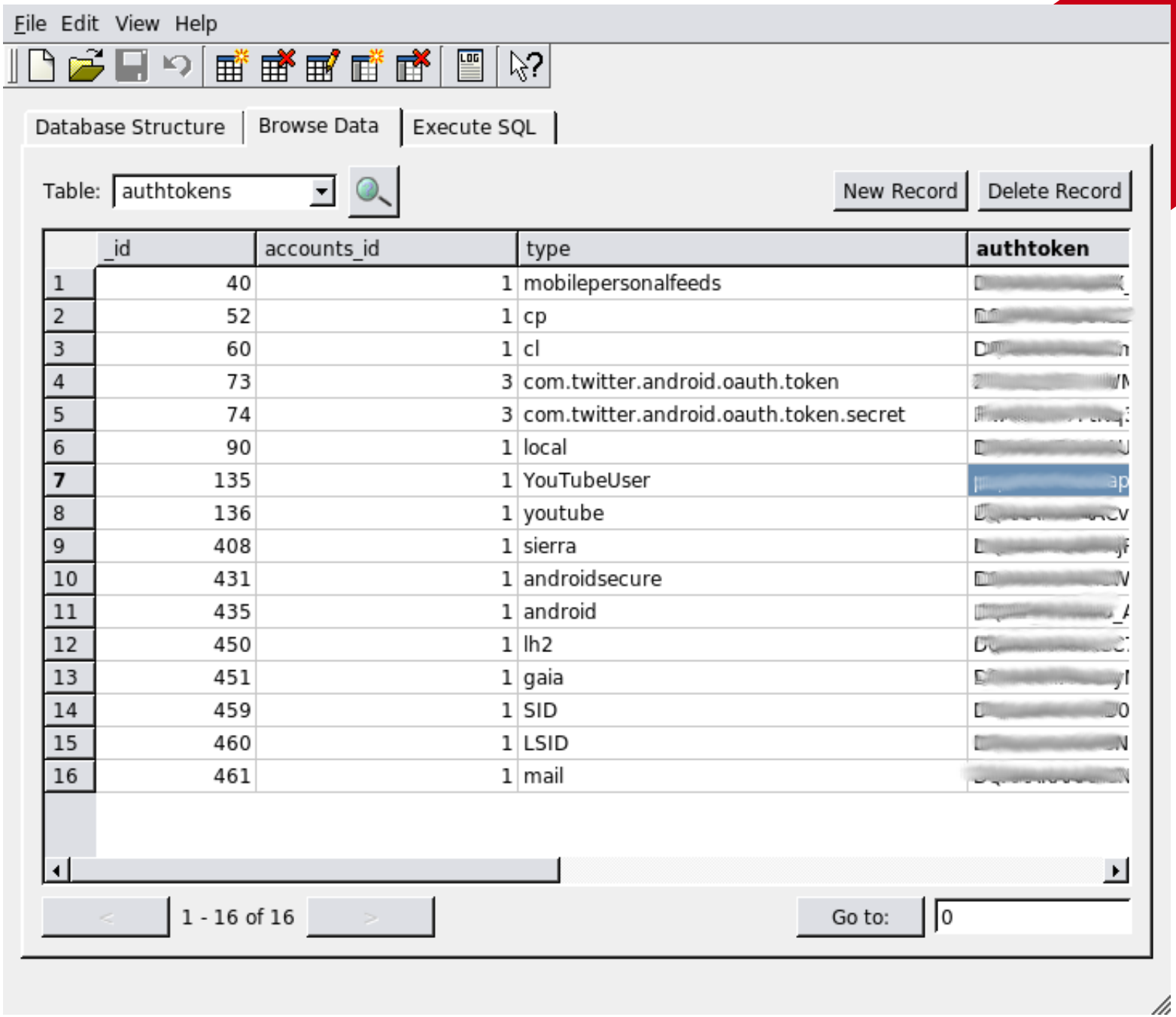
Hacking Dave … lessons

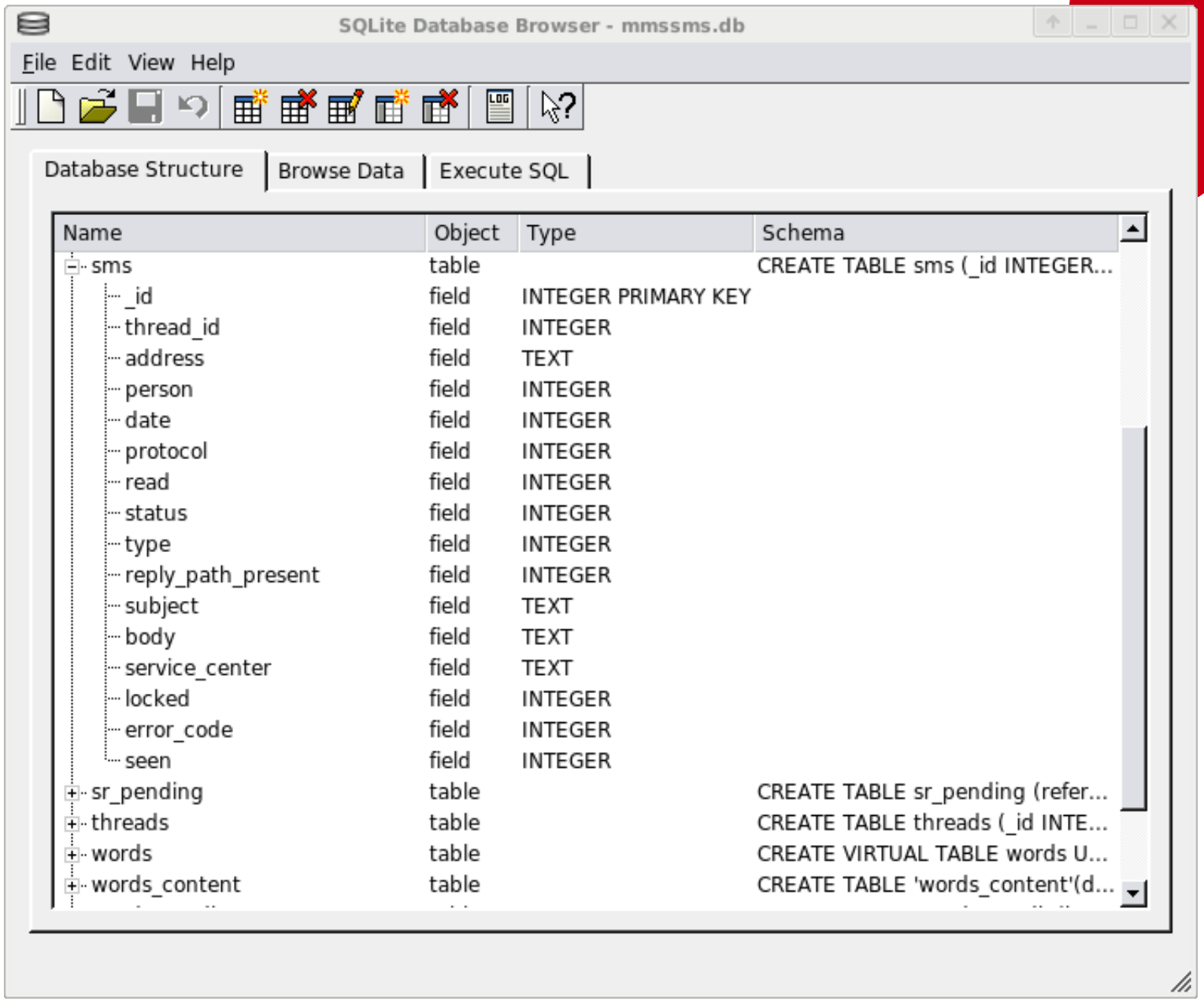
- Reasonably competent (ahum) attackers with no specific background in Android hacking can go from zero to owning Immunity's CEO in the span of about a week
- Things that helped speed up the process
  - Prior knowledge of Linux Kernel internals
  - Prior knowledge of use-after-free WebKit attacks
  - Prior knowledge of ARM architecture
  - Prior knowledge of @daveaitel's Twitter obsession
  - Team work (Thanks Matias + Agustin!)


uid=0

Got root, now what?

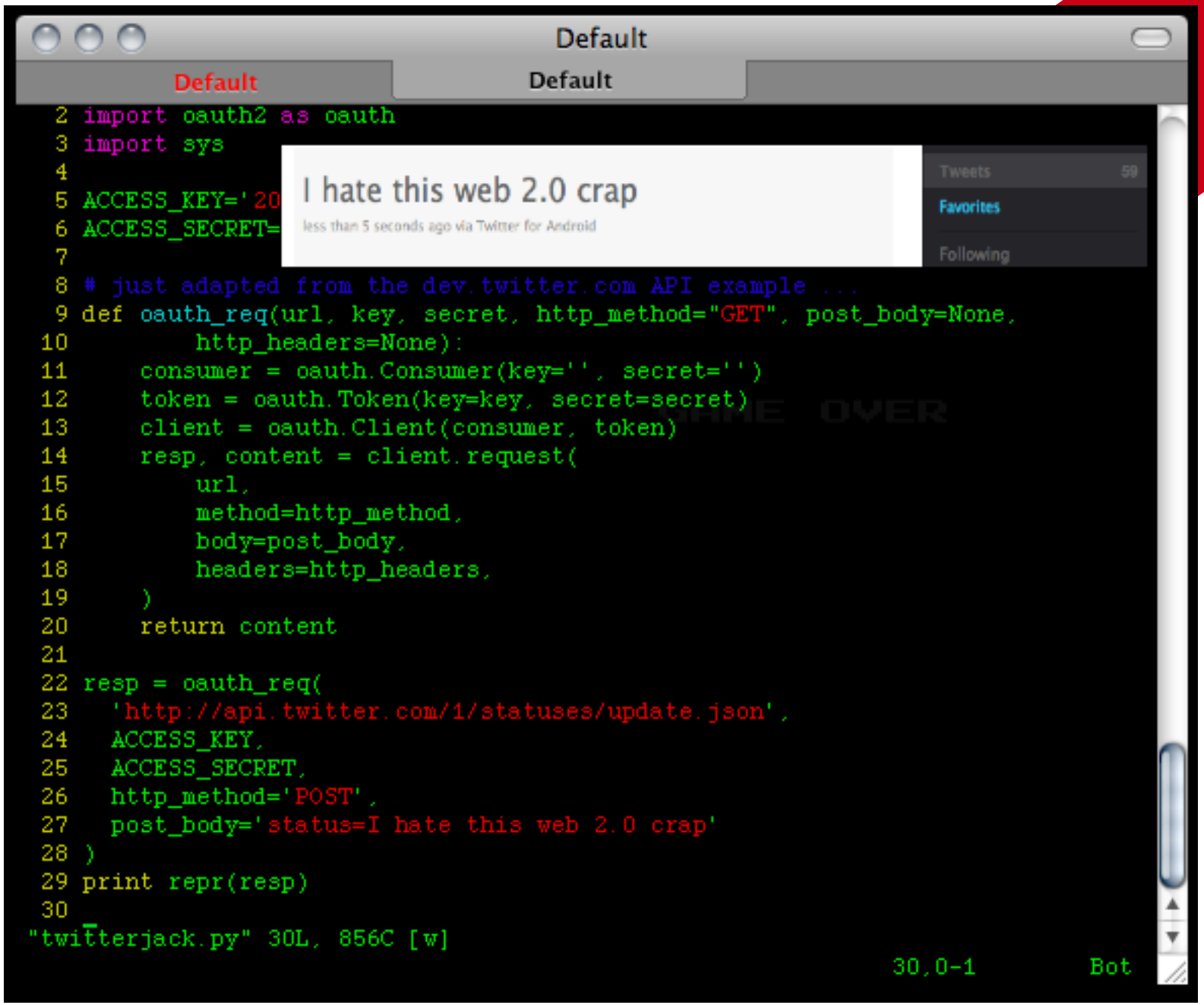
- /data/system/accounts.db
- /data/data/com.android.providers.telephony/databases/mmssms.db
- If you suck as much at SQL as we do, use SQLiteBrowser to grab auth tokens
  - http://sqlitebrowser.sourceforge.net
- To do useful things with oauth tokens use oauth2 for Python
  - https://github.com/simplegeo/python-oauth2
- You also get ClientLogin auth tokens for most google data services
  - Such as: cp, cl, l2h, mail, youtube
  - curl –header “Authorization: GoogleLogin auth=....”


File Edit View Help
Database Structure
Browse Data
Execute SQL
Table: authtokens
New Record
Delete Record
_id
accounts_id
type
authtoken
1 40 1 mobilepersonalfeeds
2 52 1 cp
3 60 1 cl
4 73 3 com.twitter.android.oauth.token
5 74 3 com.twitter.android.oauth.token.secret
6 90 1 local
7 135 1 YouTubeUser
8 136 1 youtube
9 408 1 sierra
10 431 1 androidsecure
11 435 1 android
12 450 1 lh2
13 451 1 gaia
14 459 1 SID
15 460 1 LSID
16 461 1 mail
1 - 16 of 16
Go to:
0


SQLite Database Browser - mmssms.db
File Edit View Help
Database Structure
Browse Data
Execute SQL
Name
Object
Type
Schema
sms
table
CREATE TABLE sms (_id INTEGER...
_id
field
INTEGER PRIMARY KEY
thread_id
field
INTEGER
address
field
TEXT
person
field
INTEGER
date
field
INTEGER
protocol
field
INTEGER
read
field
INTEGER
status
field
INTEGER
type
field
INTEGER
reply_path_present
field
INTEGER
subject
field
TEXT
body
field
TEXT
service_center
field
TEXT
locked
field
INTEGER
error_code
field
INTEGER
seen
field
INTEGER
sr_pending
table
CREATE TABLE sr_pending (refer...
threads
table
CREATE TABLE threads (_id INTE...
words
table
CREATE VIRTUAL TABLE words U...
words_content
table
CREATE TABLE 'words_content'(d...

Fun with oauth

- Twitter jacking is a crucial element of public humiliation, recycling oauth access keys and secrets for Twitter is easy
- Consumer key and consumer secret are unique to every Twitter application
- Because you are using an access key and access secret that was negotiated through the official Twitter Android App, your Tweets look like they came from “Twitter for Android”


Default
Default
Default
import oauth2 as oauth
import sys
ACCESS_KEY='20
ACCESS_SECRET=
I hate this web 2.0 crap
less than 5 seconds ago via Twitter for Android
Tweets 59
Favorites
Following
just adapted from the dev.twitter.com API example ...
def oauth_req(url, key, secret, http_method="GET", post_body=None,
http_headers=None):
consumer = oauth.Consumer(key='', secret='')
token = oauth.Token(key=key, secret=secret)
client = oauth.Client(consumer, token)
resp, content = client.request(
url,
method=http_method,
body=post_body,
headers=http_headers,
)
return content
resp = oauth_req(
'http://api.twitter.com/1/statuses/update.json',
ACCESS_KEY,
ACCESS_SECRET,
http_method='POST',
post_body='status=I hate this web 2.0 crap'
)
print repr(resp)
"twitterjack.py" 30L, 856C [w]
30,0-1
Bot

Backdooring options

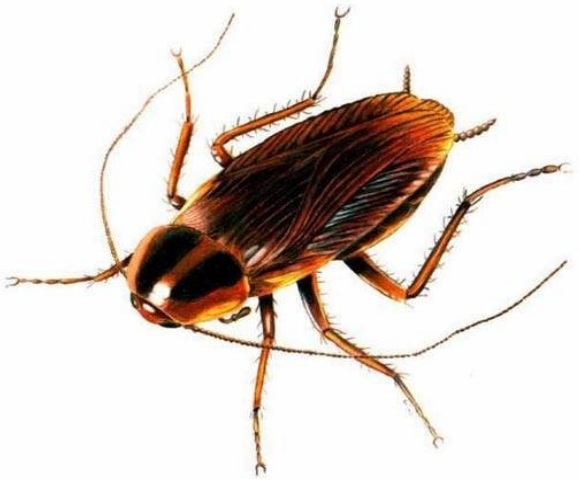
- Application level
  - With full Manifest permissions
    - Register broadcast receivers for Intents that do something interesting
      - Snoop SMS
      - Redirect calls
      - Make calls for $
- With remote root on the phone, just copy your APK to /system/app/ to have it installed (with any Manifest permissions you want)
  - Or just run 'pm install -t yourapp.apk'
- Remotely triggered Intents make it easy to communicate with your backdoor App even when it is not running, it just has to be installed
  - BroadcastReceivers for the win
- No Launch Activity in Manifest: no entry in the home Application list

Backdooring options (cont.)

- System level
  - Android Userland
    - Modify one of the userland daemons that run as root
    - Roll out a customized System Service manager that loads a backdoor service that you can talk to from apps through Binder IPC
      - Simplified: Roll out an app that registers a broadcast receiver for backdoor-app intents, and has a suid-root worker bin on the system to handle the requests (or a Kernel Trojan API available to it)
  - Linux Kernel
    - Runtime patching through /dev/mem or /dev/kmem
    - LKM
    - Downside: you lose all the convenience of the Application API hooks
    - Keep it simple, supports your userland trojan

Backdoor persistence

- Can be tricky on certain devices!
  - T-Mobile/HTC G2 eMMC storage write-through protection makes /system changes non-persistent
  - Root + Rootkits are lost on reboots
    - But Applications with full permissions are not lost
    - root not essential to persist with an interesting backdoor (SMS snooper, GoldDialer, etc.)
  - Radio settings S-OFF/S-ON secure flag controls whether or not the the eMMC is write-through protected

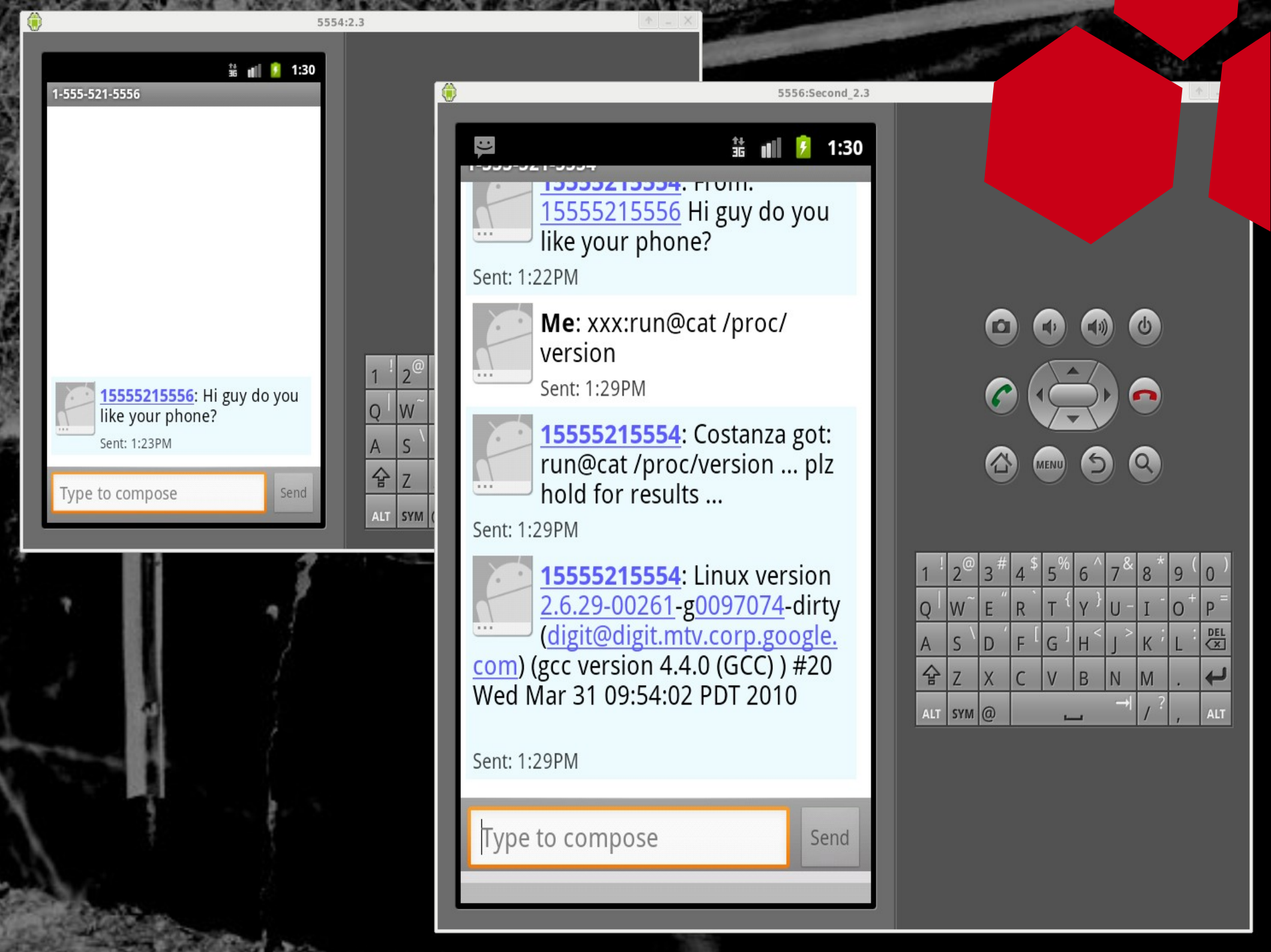


Backdoor persistence (cont.)

- Scott Walker's (scotty2) excellent gfree.c solution vs. G2's eMMC protection
  - Get root through whatever means
  - Powercycle eMMC to bring it back up in RW mode through LKM (wpthis.c)
  - Install a MMC block request filter in the kernel that removes the write protection on the hidden radio settings partition (/dev/block/mmcblk0p7 on the g2)
  - Patch the security flag to 0 (S-OFF) on the radio settings partition
  - https://github.com/tmzt/g2root-kmod/tree/master/scotty2

Backdooring … demo …

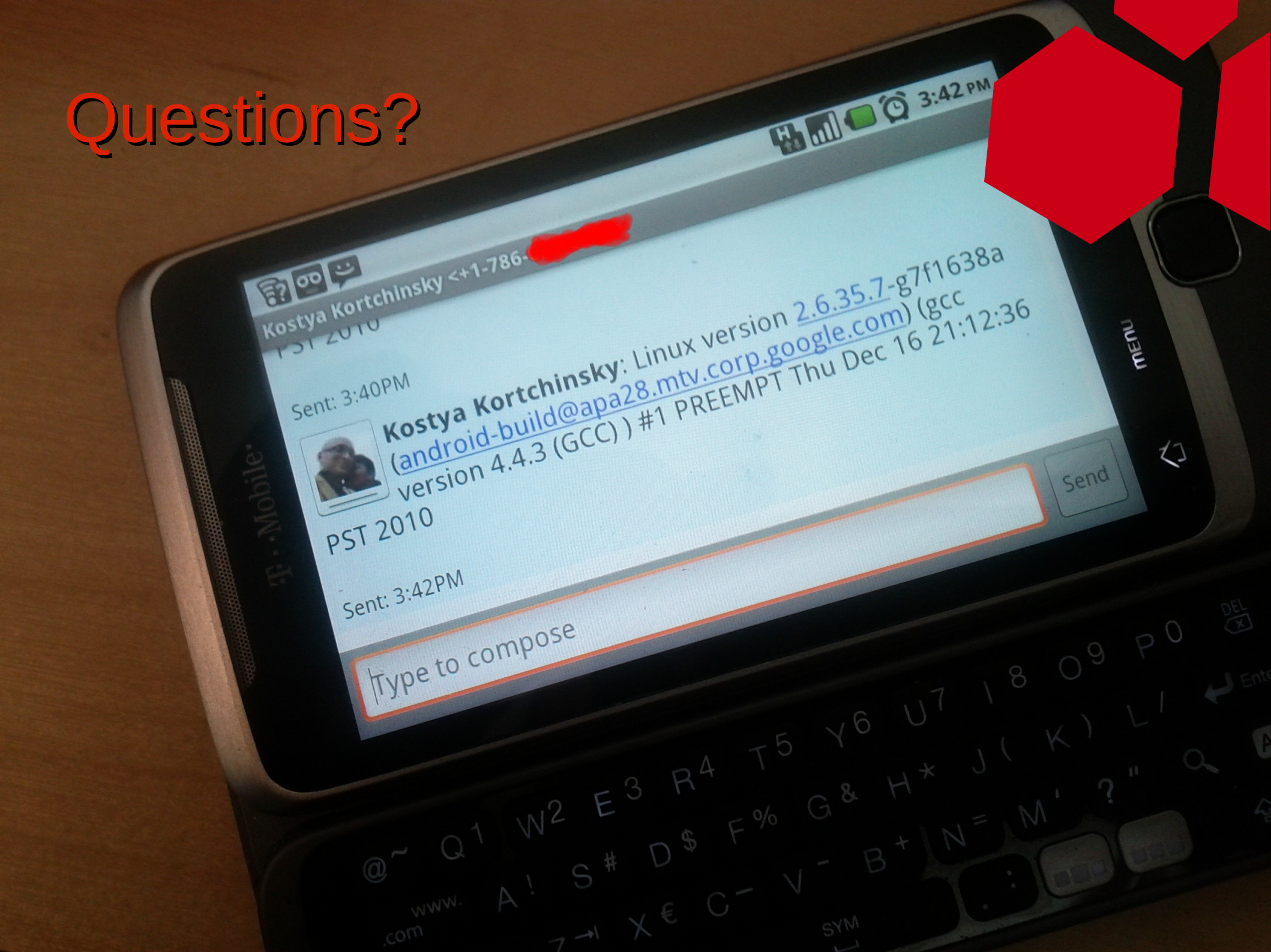
- The Costanza
  - A simple example of SMS driven Android backdoor
    - Shut up, it sounded cool on paper
    - Took a couple of hours to develop with no prior Android dev experience
  - Registers an SMS Intent Broadcast Receiver with a high priority
    - We get the SMS before the System App does
    - We can AbortBroadcast() to drop the SMS from the chain
      - C&C SMS won't show up on target phone, but WILL show up in their billing overview
  - Simple execute and HTTP POST capabilities
  - SMS snooping


5554:2.3
1:30
1-555-521-5556
15555215556: Hi guy do you like your phone?
Sent: 1:23PM
Type to compose
Send
5556:Second_2.3
1:30
15555215556 Hi guy do you like your phone?
Sent: 1:22PM
Me: xxx:run@cat /proc/version
Sent: 1:29PM
15555215554: Costanza got: run@cat /proc/version ... plz hold for results ...
Sent: 1:29PM
15555215554: Linux version 2.6.29-00261-g0097074-dirty (digit@digit.mtv.corp.google.com) (gcc version 4.4.0 (GCC)) #20 Wed Mar 31 09:54:02 PDT 2010
Sent: 1:29PM
Type to compose
Send
MENU

Conclusions

- Android is a Linux Frankenstein with an interesting attack surface
  - You should probably start auditing it
- Applications can have a lot of power
  - You don't always need root
- Developing Android Application level backdoors is easy thanks to a convenient API and a very solid SDK
- Get going whilst the getting is still good!

Questions?


3:42 PM
Kostya Kortchinsky <+1-786-
Sent: 3:40PM
Kostya Kortchinsky: Linux version 2.6.35.7-g7f1638a
(android-build@apa28.mtv.corp.google.com) (gcc
version 4.4.3 (GCC)) #1 PREEMPT Thu Dec 16 21:12:36
PST 2010
Sent: 3:42PM
Type to compose
Send
MENU
T-Mobile

References

- Jon Oberheide, “Android Hax”, Summercon 2010
  - http://jon.oberheide.org/files/summercon10-androidhax-jonoberheide.pdf
- Android Developer's Guide
  - http://developer.android.com/guide/index.html
- Sebastian Krahmer, 734C
  - http://c-skills.blogspot.com/
- Hristo Bojinov, “Mobile ASLR”, 2011
  - http://bojinov.org/professional/wisec2011-mobileaslr-paper.pdf
- Scott Walker, Gfree source
  - https://github.com/tmzt/g2root-kmod/tree/master/scotty2