DEX EDUCATION 201
ANTI-EMULATION

TIM STRAZZERE - HITCON 2013
19.07.2013

WHOAMI

- Lead Research & Response Engineer @ Lookout Mobile Security (yes.. an av)
- Reversed the Android Market/Google Play Protocol
- Always enjoyed reversing “exotic” platforms, writing tools to automate the mundane
- Junkie for reversing mobile malware, creating write ups and teaching other to help raise the bar

AGENDA

- Recap of Dex Education 101
- Who is evading emulators / How is it done
- Who is hiding emulators / How is it done
- Easy detection - Hard time hiding
- Basic emulator detection / QEMU-FOO
- “Advanced” detection of emulator systems
- TLDR; Conceptualizing detection

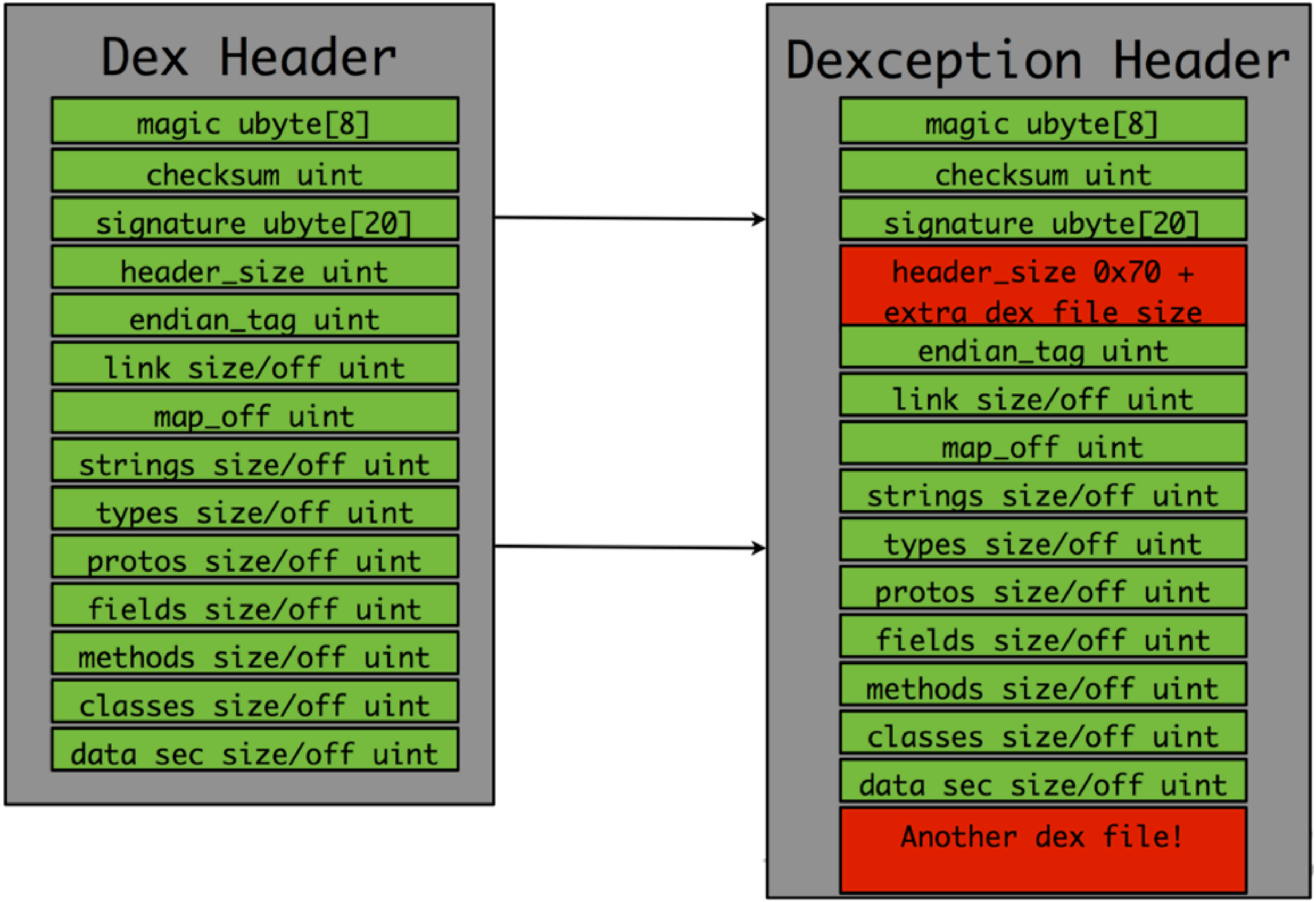
RECAP OF DEX EDUCATION 101

- Decompilers and disassemblers are easy to break but no one was doing it!
- Predicting these breakages can help up prevent and detect attacks
- Advances since then;
  - DexGuard
    ("Most sophisticated Android Malware")
  - HoseDex2Jar
- Slides available:

  www.strazzere.com/papers/DexEducation-PracticingSafeDex.pdf

RECAP OF DEX EDUCATION 101

- HoseDex2Jar used the “big ego” tactic of injecting files into the header
- Author even said hiTim in one of his method names (HOW NICE)
- github.com/strazzere/dehoser
- Latest example uses some simple crypto inside of native code - go play!


Dex Header
magic ubyte[8]
checksum uint
signature ubyte[20]
header_size uint
endian_tag uint
link size/off uint
map_off uint
strings size/off uint
types size/off uint
protos size/off uint
fields size/off uint
methods size/off uint
classes size/off uint
data sec size/off uint
Dexception Header
magic ubyte[8]
checksum uint
signature ubyte[20]
header_size 0x70 +
extra dex file size
endian_tag uint
link size/off uint
map_off uint
strings size/off uint
types size/off uint
protos size/off uint
fields size/off uint
methods size/off uint
classes size/off uint
data sec size/off uint
Another dex file!

WHO IS EVADING EMULATORS?

- Security Researchers
- Game cheaters / “hackers”
- Devs against competition
- Malware authors (?)

MOTIVE FOR EVASION?

- Games detect emulators to prevent cheating/abuse
    - Must uniquely identify devices to prevent referral abuse/easy cheating
    - Attempt to stop farming/cheats
- App devs want to “protect” secrets
- Security researchers want to break stuff, get famous and pwn people
    - duh!
- Malware authors want to avoid detection of their “products”

DETECTION IMPORTANT?

- Games / App / Research / Malware
    - All have different use cases and trying to detect different segments
- Games want to raise bar for cheating without FP
- Security researchers... Prevent getting pwned and hacing a talk at BH
- AV's want broad detection without FP

EVASION DETECTION IS DIFFERENT

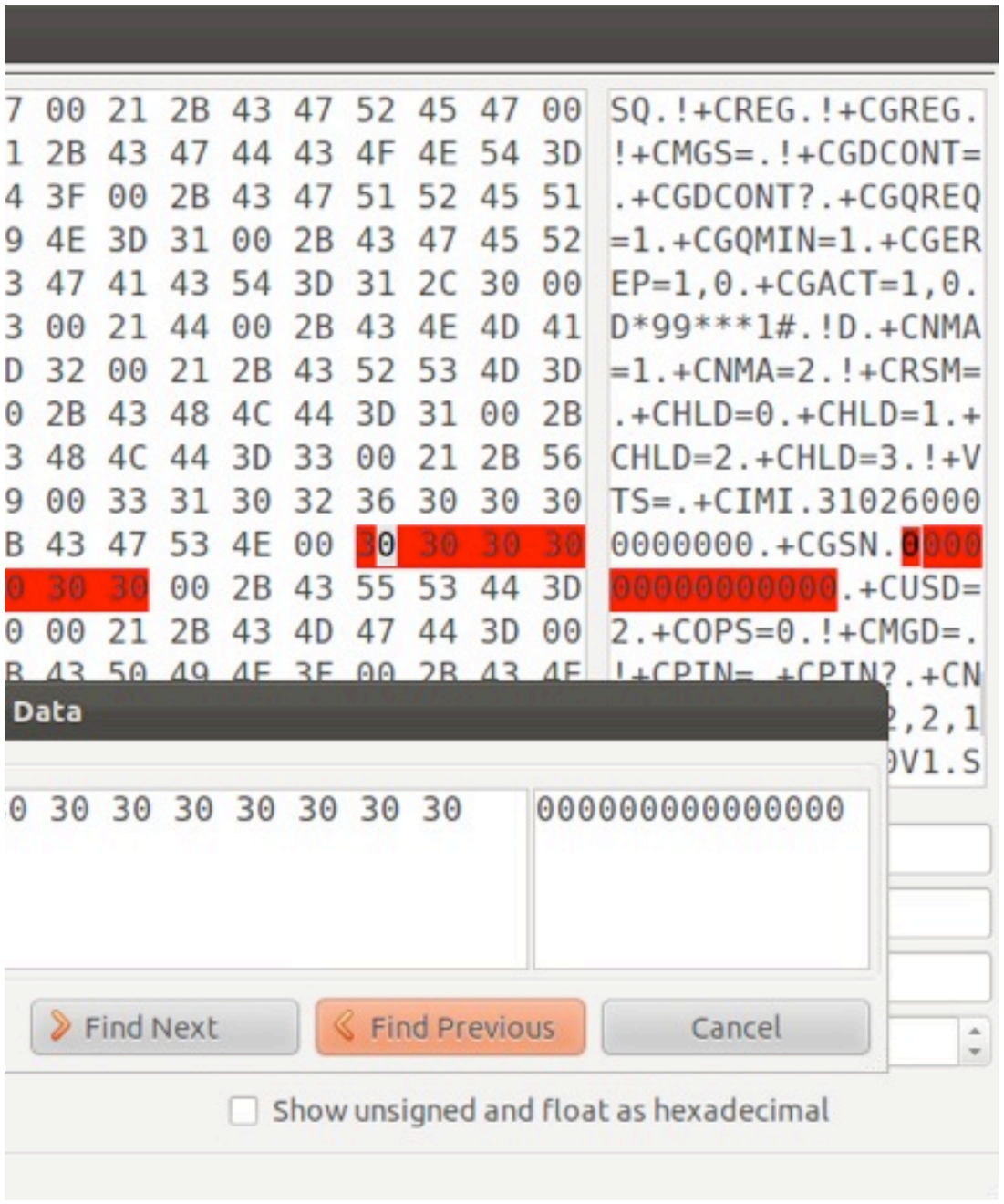
- Games / Apps want to detect and phone home
    - Either alert devs of abuse, or just fail to work properly
- Security researchers might just want to probe your infrastructure...
    - Has anyone ever probed Bouncer? (oh wait... hi Charlie/JonO!)
- AV's want malware to perform their payloads while they watch
    - If detected, malware won't perform detected/wanted behaviors

HOW DO PEOPLE EVADE EMUS?

- is phone number == 15555215554
- is imei == 012345678912345
- is device id == 000000000000000
- is build.MODEL == sdk || generic
- Traditional “red pills”
  - Bluebox/Dexterlabs produced some timing issue based red pills
  - Super interesting to develop/detect
  - Maybe not “practical”

HOW ARE PEOPLE HIDING EMUS?

- Hex edit img(unsuprisingly)
  - Change the phone number
  - Change the IMEI
  - Change device id
  - Change the fingerprints
- http://vrt-blog.snort.org/2013/04/changing-imei-provider-model-and-phone.html



EASY EMULATION DETECTION

- QEmu wasn't made for hiding
- REAL HARDWARE <-> Host <-> Android Client
- The "hardware" is inside the Host, so radio/gps/camera are communicated through QEmu pipes
- Pipes are _not_ hidden
- QEmu relies on lots of values initialized in system properties and a decently heavy infrastructure

GETPROPS == GET EMU

```
# getprop
[ARGH]: [ARGH]
[dalvik.vm.heapsize]: [48m]
[dalvik.vm.stack-trace-file]: [/data/anr/traces.txt]
[dev.bootcomplete]: [1]
[gsm.current.phone-type]: [1]
[gsm.defaultpdpcontext.active]: [true]
[gsm.network.type]: [UMTS:3]
[gsm.nitz.time]: [1342654156339]
[gsm.operator.alpha]: [Android]
[gsm.operator.iso-country]: [us]
[gsm.operator.isroaming]: [false]
[gsm.operator.numeric]: [310260]
[gsm.sim.operator.alpha]: [Android]
[gsm.sim.operator.iso-country]: [us]
[gsm.sim.operator.numeric]: [310260]
[gsm.sim.state]: [READY]
[gsm.version.ril-impl]: [android reference-ril 1.0]
[init.svc.adbd]: [running]
[init.svc.bootanim]: [stopped]
[init.svc.console]: [running]
[init.svc.debuggerd]: [running]
[init.svc.goldfish-logcat]: [stopped]
[init.svc.goldfish-setup]: [stopped]
[init.svc.installd]: [running]
[init.svc.keystore]: [running]
[init.svc.media]: [running]
[init.svc.netd]: [running]
[init.svc.qemu-props]: [stopped]
[init.svc.qemud]: [running]
[init.svc.ril-daemon]: [running]
[init.svc.servicemanager]: [running]
[init.svc.surfaceflinger]: [running]
[init.svc.vold]: [running]
[init.svc.zygote]: [running]
```

```
[net.bt.name]: [Android]
[net.change]: [net.dnschange]
[net.dns1]: [10.0.2.3]
[net.dns2]: [10.0.2.4]
[net.dnschange]: [1]
[net.eth0.dns1]: [10.0.2.3]
[net.eth0.dns2]: [10.0.2.4]
[net.eth0.gw]: [10.0.2.2]
[net.gprs.local-ip]: [10.0.2.15]
[net.hostname]: [android-e9bfcfdf35fbdff7]
[net.qtaguid_enabled]: [0]
[net.tcp.buffersize.default]: [4096,87380,110208,4096,16384,110208]
[net.tcp.buffersize.edge]: [4093,26280,35040,4096,16384,35040]
[net.tcp.buffersize.gprs]: [4092,8760,11680,4096,8760,11680]
[net.tcp.buffersize.hspa]: [4094,87380,262144,4096,16384,262144]
[net.tcp.buffersize.lte]: [524288,1048576,2097152,262144,524288,1048576]
[net.tcp.buffersize.umts]: [4094,87380,110208,4096,16384,110208]
[net.tcp.buffersize.wifi]: [524288,1048576,2097152,262144,524288,1048576]
[persist.sys.country]: [US]
[persist.sys.language]: [en]
[persist.sys.localevar]: []
[persist.sys.profiler_ms]: [0]
[persist.sys.timezone]: [America/Los_Angeles]
[persist.sys.usb.config]: [adb]
[qemu.hw.mainkeys]: [1]
[qemu.sf.fake_camera]: [back]
[qemu.sf.lcd_density]: [240]
[rild.libargs]: [-d /dev/ttyS0]
[rild.libpath]: [/system/lib/libreference-ril.so]
[ro.allow.mock.location]: [1]
[ro.baseband]: [unknown]
[ro.board.platform]: []
[ro.bootloader]: [unknown]
[ro.bootmode]: [unknown]
[ro.build.characteristics]: [default]
```

GETPROPS == GET EMU

```
[ro.build.date.utc]: [1332889705]
[ro.build.date]: [Tue Mar 27 23:08:25 UTC 2012]
[ro.build.description]: [sdk-eng 4.0.4 MR1 302030 test-keys]
[ro.build.display.id]: [sdk-eng 4.0.4 MR1 302030 test-keys]
[ro.build.fingerprint]: [generic/sdk/generic:4.0.4/MR1/302030:eng/test-keys]
[ro.build.host]: [vpba16.mtv.corp.google.com]
[ro.build.id]: [MR1]
[ro.build.product]: [generic]
[ro.build.tags]: [test-keys]
[ro.build.type]: [eng]
[ro.build.user]: [android-build]
[ro.build.version.codename]: [REL]
[ro.build.version.incremental]: [302030]
[ro.build.version.release]: [4.0.4]
[ro.build.version.sdk]: [15]
[ro.carrier]: [unknown]
[ro.com.google.locationfeatures]: [1]
[ro.config.alarm_alert]: [Alarm_Classic.ogg]
[ro.config.nocheckin]: [yes]
[ro.config.notification_sound]: [OnTheHunt.ogg]
[ro.crypto.state]: [unencrypted]
[ro.debuggable]: [1]
[ro.factorytest]: [0]
[ro.hardware]: [goldfish]
[ro.kernel.android.checkjni]: [1]
[ro.kernel.android.qemud]: [ttyS1]
[ro.kernel.console]: [ttyS0]
[ro.kernel.ndns]: [2]
[ro.kernel.qemu.gles]: [0]
[ro.kernel.qemu]: [1]
[ro.product.board]: []
[ro.product.brand]: [generic]
[ro.product.cpu.abi2]: [armeabi]
[ro.product.cpu.abi]: [armeabi-v7a]
[ro.product.device]: [generic]
```

```
[ro.factorytest]: [0]
[ro.hardware]: [goldfish]
[ro.kernel.android.checkjni]: [1]
[ro.kernel.android.qemud]: [ttyS1]
[ro.kernel.console]: [ttyS0]
[ro.kernel.ndns]: [2]
[ro.kernel.qemu.gles]: [0]
[ro.kernel.qemu]: [1]
[ro.product.board]: []
[ro.product.brand]: [generic]
[ro.product.cpu.abi2]: [armeabi]
[ro.product.cpu.abi]: [armeabi-v7a]
[ro.product.device]: [generic]
[ro.product.locale.language]: [en]
[ro.product.locale.region]: [US]
[ro.product.manufacturer]: [unknown]
[ro.product.model]: [sdk]
[ro.product.name]: [sdk]
[ro.radio.use-ppp]: [no]
[ro.revision]: [0]
[ro.runtime.firstboot]: [1342654168744]
[ro.secure]: [0]
[ro.serialno]: []
[ro.setupwizard.mode]: [OPTIONAL]
[ro.wifi.channels]: []
[status.battery.level]: [5]
[status.battery.level_raw]: [50]
[status.battery.level_scale]: [9]
[status.battery.state]: [Slow]
[sys.boot_completed]: [1]
[sys.usb.config]: [adb]
[sys.usb.state]: [adb]
[system_init.startsurfaceflinger]: [0]
[xmpp.auto-presence]: [true]
```

GETPROPS == GET EMU

- There are many things that appear “odd” in emulators
- Just masking “odd” ones might not be enough
- Smart attackers know their targets / could selectively attack
- Could use geographical context to help emulator applications predictable contexts
- a (well done) APT evasion scenario might be impossible to predict

GETPROPS == GET EMU

- How are they getting these properties?

GETPROPS == GET EMU

- How are they getting these properties?

```
public static String getProp(Context context, String property) {
    try {
        ClassLoader cl = context.getClassLoader();
        @SuppressWarnings("rawtypes")
        Class SystemProperties = cl.loadClass("android.os.SystemProperties");

        Method get = SystemProperties.getMethod("get", String.class);

        Object[] params = new Object[1];
        params[0] = new String(property);

        return (String) get.invoke(SystemProperties, params);

    } catch (IllegalArgumentException iAE) {
        throw iAE;
    } catch (Exception e) {
        return null;
    }
}
```

- Reflection! Trap it for a hopeful win
- Hook the getprop command ? (https://github.com/poliva/ldpreloadhook)

I'M A POWER USER, I USE TAINT

- Taintdroid is powerful!
  - It is however, _not_ stealth
- Taintdroid != emulation
  - Can run on "real" devices
- Talks on exfiltration of data on taintdroid
  - Why bother unless targeting that system?
  - Why not just detect and remain silent!

TAINTED?

- Detection of taint is relatively easy

TAINTED?

- Detection of taint is relatively easy
- Is package name “org.appanalysis” available

TAINTED?

- Detection of taint is relatively easy
- Is package name “org.appanalysis” available

```
public static boolean isTainted() {
    try {
        Class.forName("dalvik.system.Taint");
        return true;
    }
    catch (ClassNotFoundException exception) {
        return false;
    }
}
```

TAINTED?

- Detection of taint is relatively easy
- Is package name “org.appanalysis” available

```
public static boolean isTainted() {
    try {
        Class.forName("dalvik.system.Taint");
        return true;
    }
    catch (ClassNotFoundException exception) {
        return false;
    }
}
public static boolean isFileDescriptorTainted() {
    Class  fDClass = FileDescriptor.class;
    try {
        Field field = fDClass.getField("name");
        return true;
    } catch (NoSuchFieldException nsfe) {
        return false;
    }
}
```

CHECK THE PLUMBING

- Like previously said
  QEmu wasn't made for hiding

- "Pipes" talk to the host environment

- Publicly exposed and available if you have an internet permission

CHECK THE PLUMBING

- Easily found pipes;
  - /dev/qemu_pipe
  - /dev/socket/qemud
- Simple file check can suffice

```
public static boolean isEmulator() {
    File qemu_socket = new File("/dev/socket/qemud");
    if (qemu_socket.exists()) {
        Log.v("derp", "emulator!");
        return true;
    } else {
        Log.v("derp", "not emulator!");
        return false;
    }
}
```

- Connect for extra fun :D

CHECK THE PLUMBING

- Hiding these pipes is non-trivial
- Hardcoded / used plenty across codebase
- Used in many files, most of which stand out themselves;
  - /system/lib/ libc_malloc_debug_qemu.so
  - /sys/qemu_trace
  - /system/bin/qemu-props

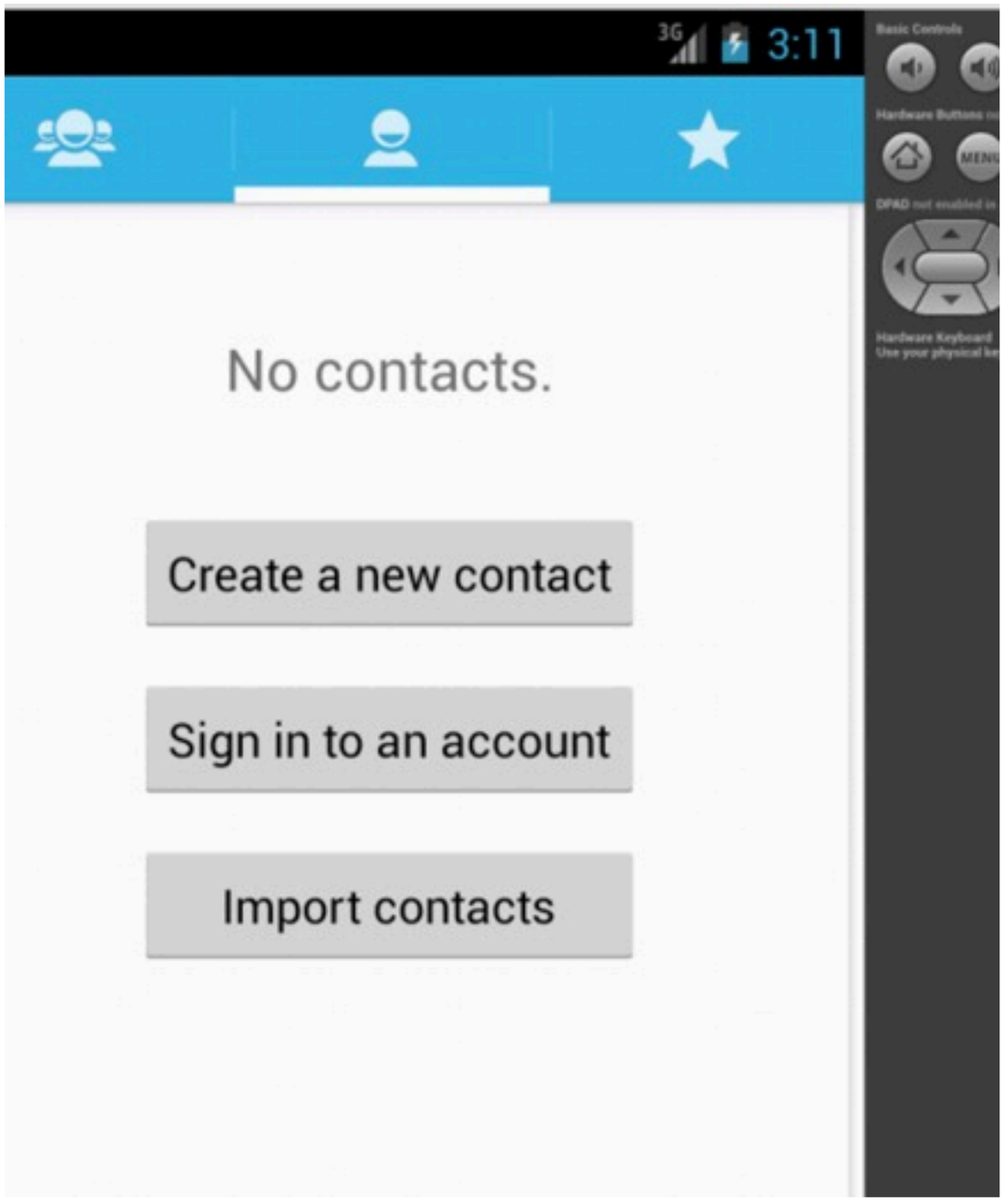
CHECK THE PLUMBING

- Drivers are pretty easily spotted as well
  - Just follow the goldfish

```
root@android:/proc/tty # ls -l
dr-x------ root     root              2013-07-18 03:06 driver
-r--r--r-- root     root            0 2013-07-18 03:06 drivers
dr-xr-xr-x root     root              2013-07-18 03:06 ldisc
-r--r--r-- root     root            0 2013-07-18 03:06 ldiscs
root@android:/proc/tty # cat drivers
/dev/tty             /dev/tty        5       0 system:/dev/tty
/dev/console         /dev/console    5       1 system:console
/dev/ptmx            /dev/ptmx       5       2 system
/dev/vc/0            /dev/vc/0       4       0 system:vtmaster
goldfish             /dev/ttyS     253 0-7 serial
pty_slave            /dev/pts      136 0-1048575 pty:slave
pty_master           /dev/ptm      128 0-1048575 pty:master
unknown              /dev/tty        4 1-63 console
```

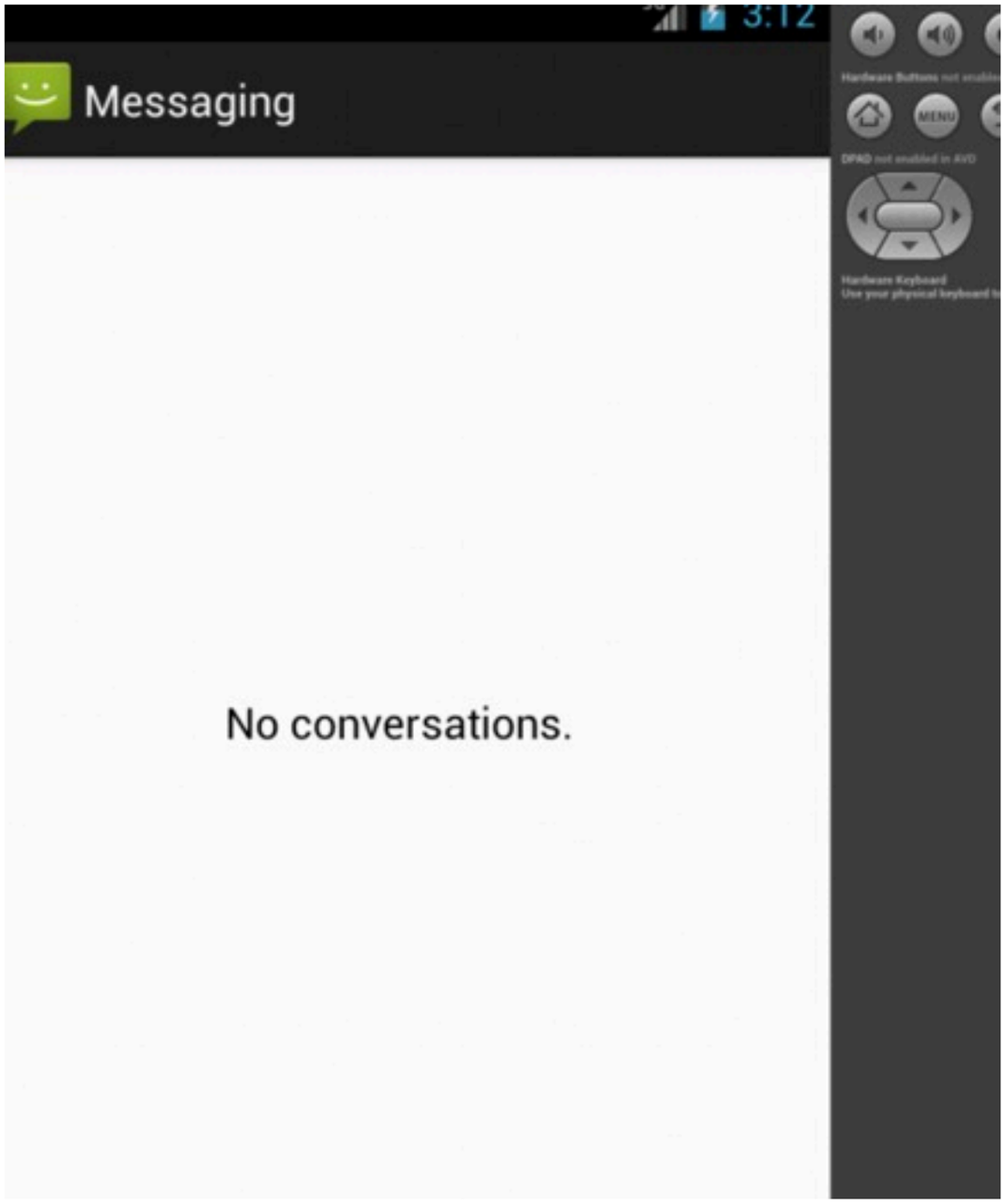
CHECKING THE CONTENT

- How many devices are being targeted?
- Know the targets
- Do people who download infected Angry Birds have NO data?



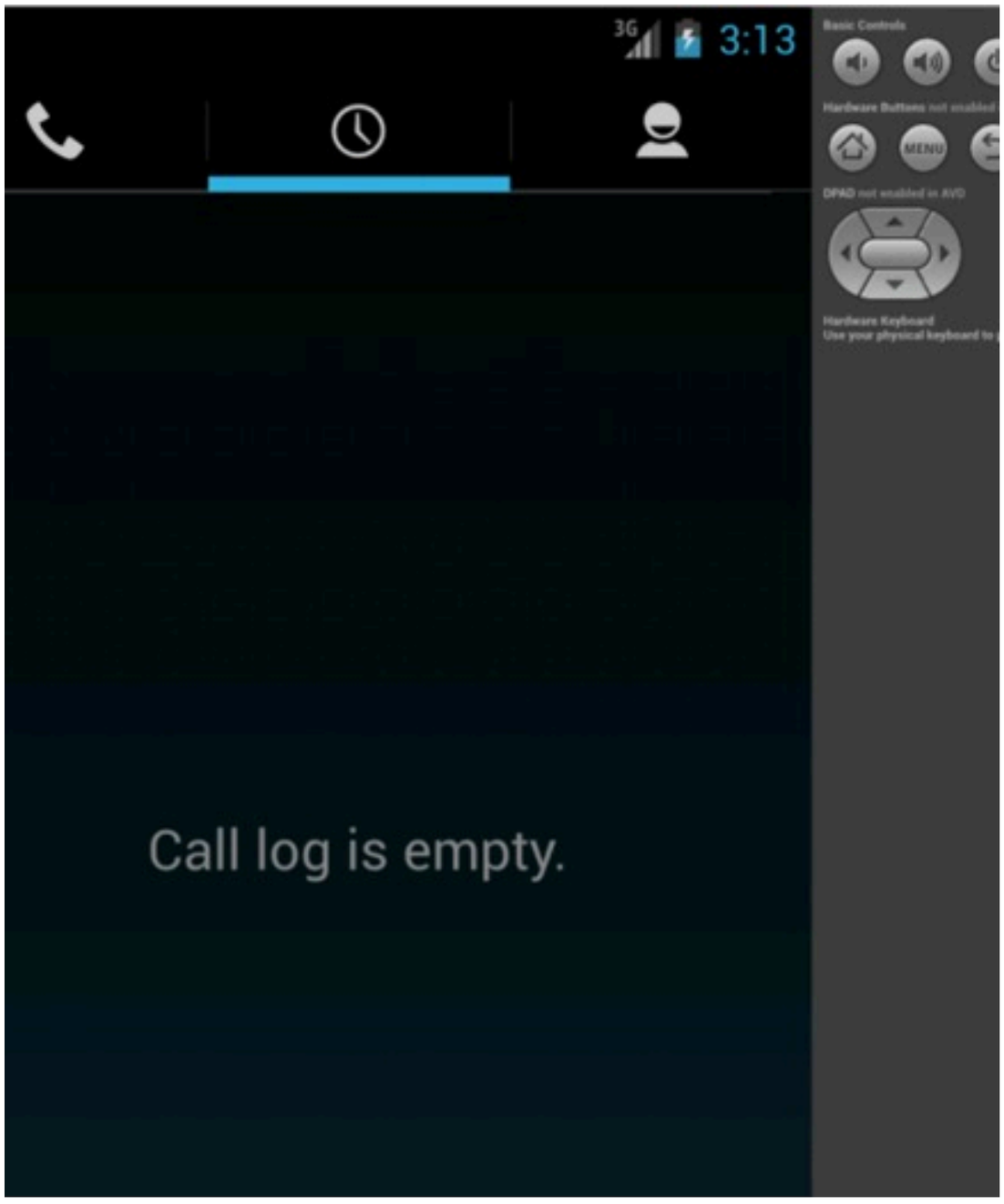
CHECKING THE CONTENT

- How many toll fraud targets are there?
- Do they normally send NO sms?
- Malware already has access to these...



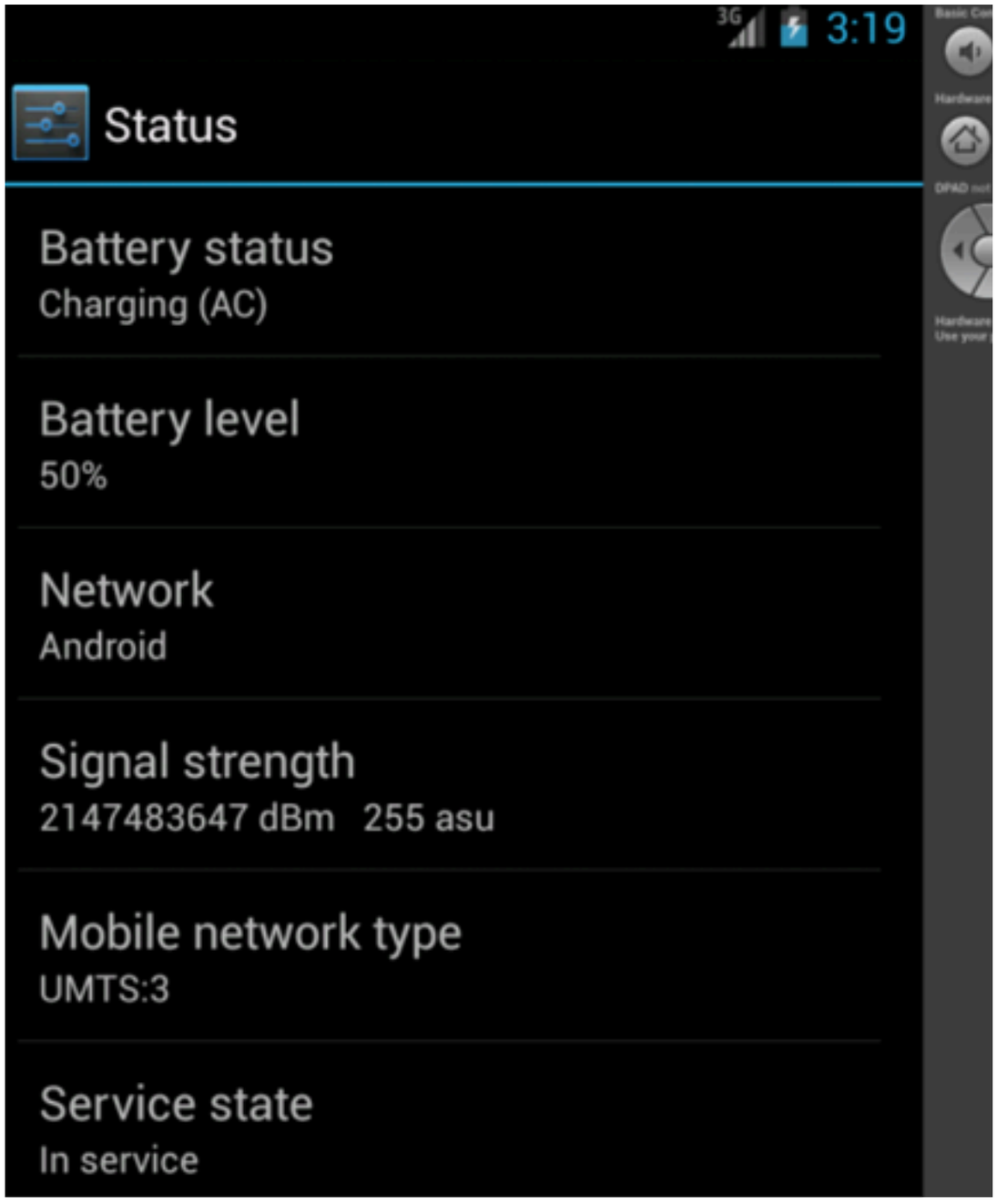
CHECKING THE CONTENT

- People may not use smartphones as phones much
- But is it expected that they've never made a call? Ever?



CHECKING THE CONTENT

- Is the device always “charging”?
- But is always at 50%?
- Never roaming?
- Always at emulator defaults?



CONTENT IS KEY

- Malware authors know who they target
    - They also know who they can affect
- Large dynamic emulator systems need context and content
- Emulation must be emulating the victim, not just the victims system

CONCEPTUALIZING DETECTION

- Talk with fG+; economics are importnat
- People (malware authors) want a ROI
- Low bar for detection means less work
- Less work on the code leads to more work on infections
- Talk at defcon about investigating Russian Toll Fraud
    - They essentially run agile shops!

THANKS!

@timstrazz
diff@xlookout.com
strazzere.com/blog
github.com/strazzere

Should follow for good info;
@osxreverser @snare @pof @jduck @thomas_cannon
@TeamAndIRC @Gunther_AR

Greets;
fG!, Lohan+, jcase, jon larimer, zuk, jduck, JF, pof, thomas cannon, snare, crypto girl, collinrm, gunther and others