BUSTICATI PRODUCTIONS PRESENTS

DISSECTING THE ANDROID BOUNCER

STARRING

DR. OBERHEIDE and DR. MILLER
THE FOLLOWING PREVIEW HAS BEEN APPROVED FOR ALL AUDIENCES BY THE MOTION PICTURE ASSOCIATION OF AMERICA

THE FILM ADVERTISED HAS BEEN RATED

RESTRICTED
UNDER 17 REQUIRES ACCOMPANYING PARENT OR ADULT GUARDIAN

For information on film ratings, go to www.filmratings.com
PINATA TIME!

- CANDY!
- SNACKS!
- BEER!
The Android Market app used to primarily use HTTP as a transport!
BACK IN THE GOOD OL' DAYS

So you could MITM the protobuf, inject your app into search results, trick people into installing malicious apps, etc
A NEW APP STORE?

Can't do as much nowadays, but still can play some tricks...

Fire up your Google Play app if you're on the wifi!
Welcome to the Jono and Charlie App Store
Welcome to the Jono and Charlie App Store

SECURE TIPS
CERTIFIED!

NOOK for Android by B&N
Barnes and Noble
Free

Formulae Pro
Espian(dev)
Installed
$1.59

The Soup
COMCAST ENTERTA...
Download

4.5 ★ ★ ★ ★
391
100,000+ downloads
December 30, 2011
7.42MB

TOP DEVELOPER
DESCRIPTION

Every day Joel McHale and The Soup crew overload their TiVos with the strange, obscure, and totally unbelievable moments in pop culture, celebrity news, and reality TV. Then they take a nap. But right after that they add a heaping helping of humor and serve up everything you need to know about the

Dissecting the Android Bouncer – Jon Oberheide / Charlie Miller – SummerCon 2012
Coming soon to a GitHub repository near you?
FEATURE PRESENTATION

A News Corporation Company

Dissecting the Android Bouncer – Jon Oberheide / Charlie Miller – SummerCon 2012
STARRING...

Google Android's Bouncer
OPERATING PLAN

• Diagnosis
  • Intro to Bouncer and Google Play

• Exploratory surgery
  • Fingerprinting Bouncer and its environment

• Open surgery
  • Abusing Bouncer in all sorts of fun ways

• Suture and close
  • How Google can fix up Bouncer
Android and Security, Feb 2, 2012

Today we’re revealing a service we’ve developed, codenamed Bouncer, which provides **automated scanning of Android Market** for potentially malicious software without disrupting the user experience of Android Market or requiring developers to go through an application approval process.

The service **performs a set of analyses** on new applications, applications already in Android Market, and developer accounts. Here’s how it works: once an application is uploaded, the service immediately starts **analyzing it for known malware, spyware and trojans**. It also looks for **behaviors** that indicate an application might be misbehaving, and compares it against previously analyzed apps to detect possible red flags. We actually **run every application** on Google’s cloud infrastructure and **simulate how it will run on an Android device** to look for hidden, malicious behavior.
• Bouncer is easily bypassed
  • No surprise there
  • Google is trying to solve a very difficult problem

• We'll show a bunch of ways
  • System, network, framework, timing, etc

• Story of how we analyzed Bouncer
  • Full of mystery and intrigue!
  • Also, pinatas and beer!
GETTING STARTED

• How do we go about dissecting Bouncer?
• How would we create such a system?
• We had lots of unanswered questions:
  • Does Bouncer use static/dynamic analysis?
  • When does Bouncer analyze the app? Are all apps analyzed?
  • How do we get Market accounts to start figuring this out?
  • Network access: open, filtered, emulated, unrestricted?
  • Environment: what's the system execution environment look like?
  • Timing: how long does our app run? Accelerated clock?
  • Input: Artificial input to the app? Program state exploration?
  • Any triggers, vulnerable services, etc?
FIRST THINGS FIRST

We need some Play accounts...
WHAT YOU NEED

• Money
• Prepaid phones
• Prepaid CCs
• EC2 micros
BURNERS FOR GMAIL
PAYMENT LOOPHOLE

We can submit apps without paying!
HOW DO WE START?

- How do we start?
- Submit a simple app that phones home to our C&C server
- See what happens?
FIRST DO NO HARM

• Hippocratic Oath forbids us from pushing malware onto innocent bystanders
  • Put warnings in the description
  • Only make available to impossible hardware
  • Make the app not interesting
  • ...ugh...

• Any other way???
SUBMISSION STEP 1

Upload your APK...

Required: Select your application's APK

Optional: Add an expansion file
If your app exceeds the 50MB APK limit, you can add expansion files. Learn more
# SUBMISSION STEP 2

## Edit Application

<table>
<thead>
<tr>
<th>Product details</th>
<th>APK files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Upload assets</strong></td>
<td></td>
</tr>
<tr>
<td>Screenshots at least 2</td>
<td>Add a screenshot: Browse... Upload</td>
</tr>
<tr>
<td>High Resolution Application Icon</td>
<td>Add a hi-res application icon: Browse... Upload</td>
</tr>
<tr>
<td>Promotional Graphic optional</td>
<td>Add a promotional graphic: Browse... Upload</td>
</tr>
<tr>
<td>Feature Graphic optional</td>
<td>Add a feature graphic: Browse... Upload</td>
</tr>
<tr>
<td>Promotional Video optional</td>
<td>Add a promotional video link: http://</td>
</tr>
<tr>
<td>Privacy Policy</td>
<td>Add a privacy policy link: http://</td>
</tr>
<tr>
<td>Marketing Opt-Out</td>
<td>Do not promote my application except in Google Play and in any Google-owned online or mobile properties. I understand that any changes to this preference may take sixty days to take effect.</td>
</tr>
</tbody>
</table>

## Fill in application metadata...
SUBMISSION STEP 3

- Press “Save” button...

```
74.125.19.84 -- [08/Apr/2012:23:33:05 -0400]
"GET /?id=9774d56d682e549c HTTP/1.1" 200 5 "-
"Apache-HttpClient/UNAVAILABLE (java 1.4)" "-
```

- Wait, what was that?!?

- Looks like Bouncer ran our app!
  - Before it was actually published to the market!
OPERATING PLAN

• Diagnosis
  • Intro to Bouncer and Google Play

• Exploratory surgery
  • Fingerprinting Bouncer and its environment

• Open surgery
  • Abusing Bouncer in all sorts of fun ways

• Suture and close
  • How Google can fix up Bouncer
• Bouncer in a nutshell
  • Dynamic runtime analysis of app
  • Emulated Android environment
  • Runs for 5 minutes
  • On Google's infrastructure
  • Allows external network access

• If we can fingerprint the environment
  • Pretend to be benign when run on Bouncer
  • Execute malicious activity when run on real devices
FINGERPRINT CLASSIFICATION

- Underlying system
  - Linux, QEMU emulator, system properties, etc
- Android Framework
  - Sensors: camera, accelerometer, gps, etc
  - Data sources: address book, sms, photos, files, etc
- Environment and behaviors
  - IP address, timing attacks, input automation, etc
SYSTEM/QEMU IDENTIFIERS

• Lots of low-hanging fruit
  • /proc/cpuinfo: goldfish
  • getprop attributes: ro.kernel.qemu
  • Obvious QEMU stuff: /sys/qemu_trace, etc
  • Many many more...

• Once the easy stuff is fixed
  • Fingerprinting QEMU based on emulation discrepancies
  • http://static.usenix.org/event/woot09.tech/full_papers/paleari.pdf
  • Could fingerprint the exact QEMU version (and exploit ;-)

Dissecting the Android Bouncer – Jon Oberheide / Charlie Miller – SummerCon 2012
# SYSTEM VITAL SIGNS

<table>
<thead>
<tr>
<th></th>
<th>Galaxy Nexus</th>
<th>Bouncer*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Brand</strong></td>
<td>Google</td>
<td>Tmobile</td>
</tr>
<tr>
<td><strong>CPUABI</strong></td>
<td>arbeabi-v7a</td>
<td>armeabi</td>
</tr>
<tr>
<td><strong>CPUABI2</strong></td>
<td>armeabi</td>
<td>unknown</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>vpbs3.mtv.corp.google.com</td>
<td>android-test-2.mtv.corp.google.com</td>
</tr>
<tr>
<td><strong>Manufacturer</strong></td>
<td>samsung</td>
<td>HTC</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>Galaxy Nexus</td>
<td>T-Mobile myTouch 3G</td>
</tr>
<tr>
<td><strong>Product</strong></td>
<td>yakju</td>
<td>opal</td>
</tr>
<tr>
<td><strong>Serial</strong></td>
<td>01469107030XXXXX</td>
<td>unknown</td>
</tr>
</tbody>
</table>

*May be version dependent on requested SDK version of submitted application*
**INVASIVE VITAL SIGNS**

<table>
<thead>
<tr>
<th></th>
<th>Galaxy Nexus</th>
<th>Emulator</th>
<th>Bouncer*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phone number</strong></td>
<td>1248760XXXX</td>
<td>15555215554</td>
<td>155552155504</td>
</tr>
<tr>
<td><strong>Phone device</strong></td>
<td>358350040XXX</td>
<td>0000000000000000</td>
<td>112358132134559</td>
</tr>
<tr>
<td></td>
<td>XXX</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Phone serial</strong></td>
<td>8901260362485</td>
<td>890141032111118510720</td>
<td>89014103211118510720</td>
</tr>
<tr>
<td><strong>Sim name</strong></td>
<td>T-Mobile</td>
<td>Android</td>
<td>T-Mobile</td>
</tr>
<tr>
<td><strong>Network name</strong></td>
<td>T-Mobile</td>
<td>Android</td>
<td>T-Mobile</td>
</tr>
</tbody>
</table>

*May be version dependent on requested SDK version of submitted application*
MORE VITALS

- Android ID: 9774d56d682e549c
  - All emulators return this ID
  - Some older phones return this as well
  - Flashed OS mods tend to return this too

- More recent tests indicate this ID may be changing and/or dynamic
BOUNCER'S OWNER

- Google account associated with the Bouncer device:

  base64.b64decode('OyBtaWxlcy5rYXJsc29uQGdtYWlsLmNvbSwgY29tLmdvb2dsZQ=='; miles.karlson@gmail.com, com.google'

- miles.karlson@gmail.com
CONTACT DATABASE

• Who does Miles hang out with?
  • Check the Android contact lists

74.125.184.94 - - [10/May/2012:09:34:19 -0500] "GET /index.html?
q=TWljaGVsbGUgTG52aW4gbWljaGVsbGUuay5sZXZp
bkBnbWFpbC5jb20= HTTP/1.1" 200 44

• michelle.k.levin@gmail.com
WHO IS MICHELLE?
LET'S GET IN TOUCH!

Michelle,

It's been so long since we've talked! How are things?

Charlie and I are doing well. We're very happy together.

I hope to see you soon in New York!

XOXO,

Jon Oberheide
SDCARD CONTENTS

- download/cat.jpg
- download/lady-gaga-300.jpg
- DCIM/Camera/IMG_20120302_142816.jpg
- android/data/passwords.txt
BOUNCER IP RANGES

- Bouncer allows Internet access
- So what IPs does it come from?
  - 74.125.0.0/16
  - Also in recent tests: 209.85.128.0/17
  - Manual review: 173.194.99.0/16

```bash
$ whois 74.125.19.84 | grep OrgName
OrgName: Google Inc.

$ whois 173.194.99.18 | grep OrgName
OrgName: Google Inc.
```
TIMING CONSIDERATIONS

• Bouncer runs your app for 5 minutes
  • Don't do anything bad for 5 minutes! Duh.
  • Not long term. Could be run later, longer...

• Timing attacks
  • Bouncer is not a physical device, QEMU is SLOW!
  • Performance/benchmark fingerprinting
  • NEON, Thumb, etc make it even more obvious
INPUT EMULATION

• Bouncer explores the app by emulating UI input, clicking, etc:

74.125.184.81 – – [10/May/2012:10:41:10 –0500] "GET /foo?q=opened HTTP/1.1" 200 413
74.125.184.89 – – [10/May/2012:10:41:11 –0500] "GET /foo?q=after_alert HTTP/1.1" 200 413
74.125.184.32 – – [10/May/2012:10:41:41 –0500] "GET /foo?q=clicked_ok HTTP/1.1" 200 413

• Predictable input actions can be used to fingerprint vs real user
OPERATING PLAN

- Diagnosis
  - Intro to Bouncer and Google Play

- Exploratory surgery
  - Fingerprinting Bouncer and its environment

- Open surgery
  - Abusing Bouncer in all sorts of fun ways

- Suture and close
  - How Google can fix up Bouncer
Remote connect-back shell demo!

http://www.youtube.com/watch?v=ZEIED2ZLEbQ
MEDICAL LICENSE ISSUES

We got caught a couple times in our early experiments doing blatantly stupid stuff
GETTING CAUGHT

- What happens when you get flagged?
- Inferred Bouncer process
  - Dynamic analysis of submitted app
  - If flagged, manual analysis by human operator
  - If deemed malicious, goodbye account!
- Manual analysis comes from different IP range (173.194.99.0/16)
  - Accidentally sent commands to the human operator once thinking it was my connect-back shell :-P
SUSPENDED

Charlie couldn't appeal :-(

Now banned from iOS AND Android!
PARASITIC COMPUTING?

• Hmm, Bouncer runs app for 5 minutes
  • 5 free minutes of Google's computation resources!

• What to do with this “free” compute power provided by Google?
  • Find aliens? Cure cancer? Nah...
  • Let's fuzz Android on Android using Android!
FUZZ ANDROID ON ANDROID

- Android self-fizzer
- Queries server for which file to test
- Grabs the file with the browser
- Checks logs for crashes
- Reports crashlog to server if crash
FUZZING LOGS

74.125.184.23 - - [11/May/2012:09:47:35 -0500] "GET /cgi-bin/getfile.pl HTTP/1.1" 200 78
74.125.184.42 - - [11/May/2012:09:47:43 -0500] "GET /cgi-bin/getfile.pl HTTP/1.1" 200 78
74.125.184.46 - - [11/May/2012:09:47:46 -0500] "GET /cgi-bin/getfile.pl HTTP/1.1" 200 78
74.125.184.89 - - [11/May/2012:09:47:48 -0500] "GET /cgi-bin/getfile.pl HTTP/1.1" 200 78
74.125.184.41 - - [11/May/2012:09:47:51 -0500] "GET /cgi-bin/getfile.pl HTTP/1.1" 200 78
74.125.184.31 - - [11/May/2012:09:47:52 -0500] "GET /pngs/183.png HTTP/1.1" 200 371
74.125.184.82 - - [11/May/2012:09:47:55 -0500] "GET /cgi-bin/getfile.pl HTTP/1.1" 200 78
74.125.184.94 - - [11/May/2012:09:51:24 -0500] "GET /cgi-bin/getfile.pl HTTP/1.1" 200 78
EULA LEGALESE

- EULA FUN
  - Bouncer clicks dialogs
  - Our submitted app pops up a EULA dialog
  - Bouncer agrees to our EULA?!?!
- “You agree you are not Bouncer”, Bouncer will click yes! Liar!
AREAS WE SKIMPED ON

• Areas to explore further
  • Static analysis by Bouncer
  • Taint propagation disruption

• Challenges
  • Time, effort
  • Clean feedback loop
• Did submit rageinthecage once
  • Still ran in Bouncer?!?
  • But probably flagged.

• One would expect a static analysis stage to short-circuit dynamic run
  • But dynamic info may still be useful to Google
STATIC ANALYSIS

- Sometimes the APK never calls back
  - Presumably this means it wasn't dynamically tested
  - The guess is it fails some static detection
- One inferred signature: "/system/bin"
  - App with "/system/bin/ls" in it never called back
  - But did call back when string was constructed dynamically!
TAINT TRACKING

• Taint tracking!

• Example use case:
  • Snarf contact data and send over the network
  • Write “signature” to flag such suspicious

• Depends on propagating taint

• How to disrupt taint propagation?
  • Reflect/filter data off/through interfaces that do not track taint metadata
TAINT DISRUPTION

- Tricky interfaces to propagate through
  - Android's SharedPrefs
  - Android's Binder IPC
  - Android's LogCat interface
  - Java's DirectBuffer interface
- Implemented these “taint breakers”
- Not enough testing to conclude which were effective though
OPERATING PLAN

- Diagnosis
  - Intro to Bouncer and Google Play

- Exploratory surgery
  - Fingerprinting Bouncer and its environment

- Open surgery
  - Abusing Bouncer in all sorts of fun ways

- Suture and close
  - How Google can fix up Bouncer
WHAT CAN GOOGLE DO

• Some easy stuff
  • eg. hide strings, emulator identifiers, etc

• Some medium stuff
  • eg. diversify IP ranges (re-use Safe Browsing crawling infrastructure)

• Some hard stuff
  • eg. prevent a sufficiently convincing model of a real user's Android device

• Generally, avoid being an oracle
WHAT CAN GOOGLE DO

• Dynamic analysis is HARD.
  • That part of Bouncer will never be perfect
  • So, attack the problem from a different angle

• Dynamic analysis portion of Bouncer only looks at the submitted app

• There's a lot of metadata related the app submission that Google judges
  • eg. Charlie got his wife's CC rejected since he used the same IP to sign up for a subsequent account
WHAT CAN GOOGLE DO

CODE SIGNING!!!

- Over two years later, still no code signing
- Static and dynamic analysis suddenly becomes less horrible
- Good for exploit mitigation too
FINAL THOUGHTS

• Bouncer doesn't have to be perfect to be useful
  • It will catch crappy malware
  • It won't catch sophisticated malware
  • Same as AV, IDS, <your favorite security tech>

• How much does Bouncer raise the bar?
  • Currently: not much
  • Future: hopefully more?
GREETZ

• Special thanks
  • Dr. Valasek, Dr. Trumpbour, and Dr. Jimbo

• Greetz
  • #busticati
  • redpantz, jlamer, deft, redpig, krnlpool, bliss, nelhage, taviso, twiz, rocky, larry, deft, thing2, drb
  • Space Pope
QUESTIONS?

Jon Oberheide  
@jonoberheide  
jon@oberheide.org

Charlie Miller  
@0xcharlie  
cmiller@openrce.org