Reducing the Window of Opportunity for Android Malware
Gotta catch’em all

Axelle Apvrille, Fortinet
Tim Strazzere, Lookout Mobile Security

EICAR Conference, May 2012
Stats are difficult to compute

- Q2 2011 sales
  [Source: Gartner]
Stats are difficult to compute

Cumulative sales
[Source: David Litchfield]
Stats are difficult to compute

So, who’s first, huh? Android? Symbian? Are old devices still used? Does this account for end-user sales or sales to operators / third parties?
Stats are difficult to compute

Available apps
[Source: Lookout]
Stats are difficult to compute

AVAILABLE APPS ON GOOGLE PLAY

- Stats are complicated
- Are those apps available from all countries?
- From all operators?
- Do app revisions count for a new app?
- What about alternate marketplaces?
Stats are difficult to compute

- Q2 2011 sales [Source: Gartner]
- Cumulative sales [Source: David Litcheld]
- Available apps [Source: Lookout]
- Nb of signatures [Source: Fortinet]
- Nb of different families [Source: Fortinet]

A. Apvrille, T. Strazzere
Stats are difficult to compute

- Q2 2011 sales
  - Source: Gartner
- Cumulative sales
  - Source: David Litcheld
- Available apps
  - Source: Lookout
- Nb of signatures
  - Source: Fortinet
- Nb of different families
  - Source: Fortinet

Stats are complicated

Who's first, huh? Android? Symbian?

Are old devices still used?

Does this account for end-user sales or sales to operators/third parties?

Stats are complicated

Are those apps available from all countries?

From all operators?

Do app revisions count for a new app?

What about alternate marketplaces?

Stats are complicated

Generic signatures detect more than one sample...

How many malware are we unaware of?

Is this representative of the risk for end-users?

Stats are complicated

What's a family?

At what point do we decide to create a new family?
Stats are difficult to compute

- Q2 2011 sales
  - Source: Gartner
- Cumulative sales
  - Source: David Litcheld
- Available apps
  - Source: Lookout
- Nb of signatures
  - Source: Fortinet
- Nb of different families
  - Source: Fortinet

A. Apvrille, T. Strazzere
Stats are difficult to compute

- Q2 2011 sales [Source: Gartner]
- Cumulative sales [Source: David Litcheld]
- Available apps [Source: Look out]
- Nb of signatures [Source: Fortinet]
- Nb of different families [Source: Fortinet]

Stats are complicated
What’s a family?
At what point do we decide to create a new family?
How many Android malware? How much is it growing?

Our perception of world depends on our knowledge

**Figure:** Aristotle’s Universe (source: AlienCitadel)

**Figure:** The Copernican Universe (source: Tpellk)
What are we missing?

1781: William Herschel discovers Uranus

1846: Johan Galle discovers Neptune

1924: Edwin Hubble discovers new galaxies

Android malware

- How blind are we?
- Is there something to see and how much?
- How long have malware been in the wild?
What our paper is about

Figure: Galilei’s telescope

(Well, that wouldn’t be very modest, of course...)

Our goal - Android only

- Estimate age of malicious samples
- Preliminary tools and methods to unknown malware in the wild
- Reducing the window of opportunity of Android malware
Aging malicious samples

Certificate’s begin date

$ keytool -printcert -file ./META-INF/CERT.RSA
...
Valid from: Wed Mar 02 19:15:44 CET 2011
...

▶ Approximation. Day the certificate was created.
▶ Does not work for AOSP keys.

Package’s zip date

-rw-r--r-- 1 axelle axelle 664 Dec 20 03:36 CERT.RSA

▶ But... gives better results
Average: 80 days after release!
Why are we missing malware?

Difficult to make an inventory of Android apps
- 400,000 apps in Google Play
- 199,917 in 10 other marketplaces
- No count for 37 other marketplaces

Difficult to crawl marketplaces
We'll talk about that + classical failures
Did not spot the malicious parts etc
Why are we missing malware?

Difficult to make an inventory of Android apps

- 400,000 apps in Google Play
- 199,917 in 10 other marketplaces
- No count for 37 other marketplaces
- Other marketplaces?
Why are we missing malware?

**Difficult to make an inventory of Android apps**
- 400,000 apps in Google Play
- 199,917 in 10 other marketplaces
- No count for 37 other marketplaces
- Other marketplaces?

**Difficult to crawl marketplaces**
*We’ll talk about that*
Why are we missing malware?

Difficult to make an inventory of Android apps
- 400,000 apps in Google Play
- 199,917 in 10 other marketplaces
- No count for 37 other marketplaces
- Other marketplaces?

Difficult to crawl marketplaces
We’ll talk about that

+ classical failures
Did not spot the malicious parts etc
Once a crawler - always a crawler, right?

- Not as simple as a normal crawler
- Requires reversing of Vending.apk → No official public API
- $v1 = \text{Base64( Protobuf ( commands) )} \rightarrow \text{return Base64( Protobuf ( results ) )}$
- $v2 = \text{RESTFUL} \rightarrow \text{return of Base64( Protobuf ( results ) )}$
Different Contexts

### Normal Crawler Context
- Sign in (optional)
- Enumerate all apps, collecting meta data → Often new apps are highlighted/easy to find
- Download all new APKs
- Rate limit along the way to prevent bans

### Google Play Contexts
- Must mock an actual device → Only see applications viewable to the device
- Enumerate applications (limited to 500 per category/search
- No more "just-in" category anymore
- Emulate only a few contexts for each account to prevent bans
What makes a Google Play context?

So many different details!

- 1,312+ devices accessing the market
- 136+ countries officially accessible
- 109+ carriers officially supported
- 20+ languages supported
- 12+ device SDK levels
- Lucky we can get most of the apps by targeting the majority of devices
Why so many contexts?

- Most (malware) devs have been targeting the bulk, getting the largest ROI.
- This could easily change, devs can target their apps to an audience.

What happens if someone wants to target a specific vulnerability?

- They also want to target a specific country since they can only use premium SMS on a specific carrier.
- Devs can target the device specifically, the country, and even the carrier—generic crawlers could easily miss this.
Who cares?

Why so many contexts?

- Most (malware) devs have been targeting the bulk, getting the largest ROI
- This could easily change, devs can target their apps to an audience
- What happens if someone wants to target a specific devices vuln?
Who cares?

Why so many contexts?

▶ Most (malware) devs have been targeting the bulk, getting the largest ROI
▶ This could easily change, devs can target their apps to an audience
▶ What happens if someone wants to target a specific devices vuln?
▶ They also want to target a specific country since they can only use premium SMS on a specific carrier?
Why so many contexts?

- Most (malware) devs have been targeting the bulk, getting the largest ROI.
- This could easily change, devs can target their apps to an audience.
- What happens if someone wants to target a specific devices vuln?
- They also want to target a specific country since they can only use premium SMS on a specific carrier.
- Devs can target the device specifically, the country and even the carrier - generic crawlers could easily miss this.
Creating accounts

Building a crawling robot army

- Create a new account
- Allow the account to only access a few contexts
- Initial sync with Google Play → Receive device specifics
- Get an auth-token → refresh every two weeks
- Store accounts in DB for later use in metadata / download retrieval
Ensure rate-limiting (different limits for each part)

### Getting Metadata
- Select context to search
- Enumerate apps from all 24 app categories / 6 game categories
- Repeat the enumeration for free / paid and trending (500 max for each)
- Save metadata and context, if was new, to DB/storage
- Enqueue for download if binary appears new

### Downloading the APKs
- Retrieve new metadata results, load the context used
- Issue download request (follow redirect)
- Store binary
Maintaining this beast of a crawler

- Make sure rate limiting steady, otherwise bans occur to accounts or IP address
- Keep accounts "healthy", should attempt to look like real accounts
- Monitor ROI for contexts (enable more accounts if necessary)
- Monitor for protocol changes, backwards compat. seems good, but can always break
Risk Evaluation Engine - Heuristics

- Unpack APK, ZIP
- Disassemble using APKTool or Baksmali
- Test package properties
- Help analyst: dex2jar, unzip, unjar
- Manifest properties
- Signing certificate properties
- Search for embedded executables and inspect
- Code’s properties
- Search for given combinations
What is a property detector?
- Detect risky situations
- Static check against the package
- (Relatively) simple test
- States a tendency, never guarantees clean/malicious

Detector examples
- Use of AOSP signing certificate → Risk for users with custom ROM
- Call to Runtime.exec() → Run Unix commands, e.g. pm install
### 40+ property detectors

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissions</td>
<td>SEND_SMS, RECEIVE_SMS</td>
<td>RE-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Manifest</td>
</tr>
</tbody>
</table>

A. Apvrille, T. Strazzere
## 40+ property detectors

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissions</td>
<td>SEND_SMS, RECEIVE_SMS ...</td>
<td>Manifest</td>
</tr>
<tr>
<td>API calls</td>
<td>sendTextMessage(), getDeviceId(), DexClassLoader,KeySpec ...</td>
<td>Code</td>
</tr>
</tbody>
</table>
### 40+ property detectors

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissions</td>
<td>SEND_SMS, RECEIVE_SMS ...</td>
<td>Manifest</td>
</tr>
<tr>
<td>API calls</td>
<td>sendTextMessage(), getDeviceId(), DexClassLoader,KeySpec ...</td>
<td>Code</td>
</tr>
<tr>
<td>Hidden executables</td>
<td>ARM, zip, jar</td>
<td>assets, res/raw, lib</td>
</tr>
</tbody>
</table>
## 40+ Property Detectors

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissions</td>
<td>SEND_SMS, RECEIVE_SMS ...</td>
<td>Manifest</td>
</tr>
<tr>
<td>API calls</td>
<td>sendTextMessage(), getDeviceId(), DexClassLoader, KeySpec ...</td>
<td>Code</td>
</tr>
<tr>
<td>Hidden executables</td>
<td>ARM, zip, jar</td>
<td>assets, res/raw, lib</td>
</tr>
<tr>
<td>Unix commands</td>
<td>pm install ...</td>
<td>Executables</td>
</tr>
</tbody>
</table>
## 40+ property detectors

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissions</td>
<td>SEND_SMS, RECEIVE_SMS</td>
<td>Manifest</td>
</tr>
<tr>
<td>API calls</td>
<td>sendTextMessage(), getDeviceId(), DexClassLoader, KeySpec</td>
<td>Code</td>
</tr>
<tr>
<td>Hidden executables</td>
<td>ARM, zip, jar</td>
<td>assets, res/raw, lib</td>
</tr>
<tr>
<td>Unix commands</td>
<td>pm install ...</td>
<td>Executables</td>
</tr>
<tr>
<td>Geography</td>
<td>+86 ...</td>
<td>classes.dex, certificate</td>
</tr>
</tbody>
</table>
## 40+ Property Detectors

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissions</td>
<td>SEND_SMS, RECEIVE_SMS ...</td>
<td>Manifest</td>
</tr>
<tr>
<td>API calls</td>
<td>sendTextMessage(), getDeviceId(), DexClassLoader, KeySpec ...</td>
<td>Code</td>
</tr>
<tr>
<td>Hidden executables</td>
<td>ARM, zip, jar</td>
<td>assets, res/raw, lib</td>
</tr>
<tr>
<td>Unix commands</td>
<td>pm install ...</td>
<td>Executables</td>
</tr>
<tr>
<td>Geography</td>
<td>+86 ...</td>
<td>classes.dex, certificate</td>
</tr>
<tr>
<td>URL detectors</td>
<td>C&amp;Cs</td>
<td>Executables</td>
</tr>
</tbody>
</table>
## 40+ property detectors

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissions</td>
<td>SEND_SMS, RECEIVE_SMS ...</td>
<td>Manifest</td>
</tr>
<tr>
<td>API calls</td>
<td>sendTextMessage(), getDeviceId(), DexClassLoader,KeySpec ...</td>
<td>Code</td>
</tr>
<tr>
<td>Hidden executables</td>
<td>ARM, zip, jar</td>
<td>assets, res/raw, lib</td>
</tr>
<tr>
<td>Unix commands</td>
<td>pm install ...</td>
<td>Executables</td>
</tr>
<tr>
<td>Geography</td>
<td>+86 ...</td>
<td>classes.dex, certificate</td>
</tr>
<tr>
<td>URL detectors</td>
<td>C&amp;Cs</td>
<td>Executables</td>
</tr>
<tr>
<td>Package properties</td>
<td>Size, AOSP platform cert</td>
<td>Package</td>
</tr>
</tbody>
</table>
## 40+ property detectors

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissions</td>
<td>SEND_SMS, RECEIVE_SMS ...</td>
<td>Manifest</td>
</tr>
<tr>
<td>API calls</td>
<td>sendTextMessage(), getDeviceId(),</td>
<td>Code</td>
</tr>
<tr>
<td></td>
<td>DexClassLoader, KeySpec ...</td>
<td></td>
</tr>
<tr>
<td>Hidden executables</td>
<td>ARM, zip, jar</td>
<td>assets, res/raw, lib</td>
</tr>
<tr>
<td>Unix commands</td>
<td>pm install ...</td>
<td>Executables</td>
</tr>
<tr>
<td>Geography</td>
<td>+86 ...</td>
<td>classes.dex, certificate</td>
</tr>
<tr>
<td>URL detectors</td>
<td>C&amp;Cs</td>
<td>Executabes</td>
</tr>
<tr>
<td>Package properties</td>
<td>Size, AOSP platform cert</td>
<td>Package</td>
</tr>
<tr>
<td>Combinations</td>
<td>LOCATION + INTERNET</td>
<td>Manifest, code</td>
</tr>
</tbody>
</table>
Writing property detectors

Make a call

In AndroidManifest.xml:

```xml
<uses-permission android:name="android.permission.CALL_PHONE"/>
```

```java
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:1234"));
```

To consider...

- `CALL_PRIVILEGED` permission
- `ACTION_DIAL`: does not call, but enters the phonenumber
- `Uri.parse("content://contacts/people/1")`
- Beware `PROCESS_OUTGOING_CALLS`
- Advertisement libs use it (e.g Admob)
Properties are never trivial

```perl
$grep = 'egrep -rl 'KeySpec|SecretKey|Cipher'
   "$location/smali"';
@grep_list = split( '\n/', $grep );
if (@grep_list) {
    foreach my $grep (@grep_list) {
        if ($grep !~ /com\/google\/ads/ &&
            $grep !~ /mobileads\/google\/com/ &&
            $grep !~ /com\/android\/vending\/licensing/ &&
            $grep !~ /openfeint/ &&
            $grep !~ /gameloft/ &&
            $grep !~ /javax\/microedition\/io\/SecurityInfo/ &&
            $grep !~ /oauth\/signpost\/signature/ &&
            $grep !~ /org\/apache\/james\/mime4j\// &&
            $grep !~ /com\/google\/android\/youtube\/core/ ) {
            $self->{sample}->report2file("Use of encryption: ");
            $self->{sample}->{encryption} = true;
        }
    }
} ...
```
### Context
- A subset of 97 malware + 217 clean files
- Assign weights: difference of percentages

### Statistics (see paper)
Malware send or receive SMS more than clean files
- 59% of malware send SMS against 6% of clean files

Other things malware like:
- Use HTTP POSTs (68% - 25%)
- Request both SMS and INTERNET permission (46% - 6%)
- Retrieve phone’s IMEI (63% - 20%)
- Use encryption (34% - 10%)
- List installed packages (33% - 5%)
Automatic Analysis Report

Thu Apr 19 14:32:34 2012

light grayed italic lines indicate samples this script was unable to analyze successfully

Internet = does sample connect to Internet?
SMS = does sample send/receive SMS?
MMS = does sample send/receive MMS?
Install = does sample install other applications?
Store = can the sample be downloaded from an AppStore/Android Market?
Enc = does sample use encryption
GPS = does sample use phone GPS
Version = which OS version does the sample require

<table>
<thead>
<tr>
<th>Filename</th>
<th>Risk</th>
<th>Internet</th>
<th>SMS</th>
<th>MMS</th>
<th>Install</th>
<th>Store</th>
<th>Enc</th>
<th>GPS</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>./3Banana.notes.apk</td>
<td>11</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.5</td>
</tr>
<tr>
<td>./47_32590_11073013120108667jng9o1.apk</td>
<td>22</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.5</td>
</tr>
<tr>
<td>./AdvancedTaskManager-3.7-.apk</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.5</td>
</tr>
<tr>
<td>./DroidBreakout_v1-.4.apk</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./DungeonHunter.working.apk</td>
<td>3</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.0</td>
</tr>
</tbody>
</table>
A fair dataset

- **947** samples, checked to be *clean*
- **107** *malicious* samples, taken from Contagio’s dump and exchange with NetQin
- Do not re-use samples used for weight
- Do not use our own malicious samples
# Highest scores

<table>
<thead>
<tr>
<th>Sample</th>
<th>Score</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>7734626341799e6ec8c3db21722b</td>
<td>61</td>
<td>Android/DroidKung-Fu.B!tr</td>
</tr>
<tr>
<td>0f2375e7c3239b569a0b0322261bcom.swampy.sexpos.apk</td>
<td>58</td>
<td>Android/Pjapps.B!tr</td>
</tr>
<tr>
<td>Andr_PJApps-Gen_f051eeab57e42d5...apk</td>
<td>57</td>
<td>Android/Pjapps.A!tr</td>
</tr>
<tr>
<td>jeeccalendar.apk</td>
<td>56</td>
<td>Android/CrazyVampire-A!tr</td>
</tr>
<tr>
<td>0091556ed96b3b5aa0af62e70751</td>
<td>54</td>
<td>Android/DroidKungFu-D!tr</td>
</tr>
<tr>
<td>BatterySaver.apk</td>
<td>52</td>
<td>Android/FakeDoc.A!tr</td>
</tr>
<tr>
<td>6_35228_1c0a6b1c5d24cbba9b</td>
<td>51</td>
<td>Android/DroidCoupon-A!tr</td>
</tr>
<tr>
<td>golddream_sample.apk</td>
<td>51</td>
<td>Android/GoldDream.A!tr</td>
</tr>
</tbody>
</table>
Detailed output for Android/PJapps

<table>
<thead>
<tr>
<th>APK Name</th>
<th>Score</th>
<th>Requires</th>
<th>Uses HTTP</th>
<th>HTTP POSTs</th>
<th>Connects to Internet</th>
<th>SMS/HTTP POSTs</th>
<th>Install Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>/steamypjapps-infected.apk</td>
<td>52</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

URL: http://ads.dt.mydas.mobi/getAd.php5?asid=
URL: http://www.latest.androidpickup.appspot.com/request
URL: http://androidpickup.appspot.com/signup?
URL: http://xxxxxxxxxx9:8618/client/android/a.apk

Trying to download an APK (1)

Uses HTTP (3)
Probably does HTTP POSTs (7)
Probably connects to Internet (10)
Permission to write/send SMS (15)
Permission/Action filter to receive SMS/WAP Push (19)
Requesting permission to install packages (20)
Package signed on Feb 29 2008
Certificate info:
Owner: EMAILADDRESS=android@android.com ..
Serial no: 936eacbe07f201df
Uses Android Dev Certificate (21)
.. 
Code sends SMS: sendTextMessage|
   sendMultipartTextMessage spotted (30)
Code probably reads SMS: SMS stuff spotted (35)
Reads phone IMEI: getDeviceId spotted (39)
Reads phone IMSI: getSubscriberId spotted (42)
Gets carrier: getNetworkOperator spotted (43)
Gets phone number: getLine1Number spotted (45)
getSimSerialNumber spotted (47)
.. 
Possibly sending email. (50)
Listing installed packages spotted ...
RISK SCORE: 52
<table>
<thead>
<tr>
<th>Activity</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raising the alarm</td>
<td>YES</td>
</tr>
<tr>
<td>Sends SMS</td>
<td>YES</td>
</tr>
<tr>
<td>Contacts a remote server</td>
<td>YES, but did not spot the right URL (obfuscated)</td>
</tr>
<tr>
<td>Posts information</td>
<td>YES, but not used in the malicious part</td>
</tr>
<tr>
<td>Retrieves IMEI, IMSI, operator, phone number</td>
<td>YES</td>
</tr>
<tr>
<td>Lists installed packages</td>
<td>YES</td>
</tr>
<tr>
<td>Sends emails</td>
<td>YES, but not used in the malicious part</td>
</tr>
</tbody>
</table>
### Android/PJapps: How accurate?

<table>
<thead>
<tr>
<th>Function</th>
<th>YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raising the alarm</td>
<td>YES</td>
</tr>
<tr>
<td>Sends SMS</td>
<td>YES</td>
</tr>
</tbody>
</table>
### Android/PJapps: How accurate?

<table>
<thead>
<tr>
<th>Action</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raising the alarm</td>
<td>YES</td>
</tr>
<tr>
<td>Sends SMS</td>
<td>YES</td>
</tr>
<tr>
<td>Contacts a remote server</td>
<td>Yes, but did not spot the right URL (obfuscated)</td>
</tr>
</tbody>
</table>
## Android/PJapps: How accurate?

<table>
<thead>
<tr>
<th>Activity</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raising the alarm</td>
<td>YES</td>
</tr>
<tr>
<td>Sends SMS</td>
<td>YES</td>
</tr>
<tr>
<td>Contacts a remote server</td>
<td>Yes, but did not spot the right URL (obfuscated)</td>
</tr>
<tr>
<td>POSTs information</td>
<td>Yes, but not used in the malicious part</td>
</tr>
<tr>
<td>Activity</td>
<td>Status</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Raising the alarm</td>
<td>YES</td>
</tr>
<tr>
<td>Sends SMS</td>
<td>YES</td>
</tr>
<tr>
<td>Contacts a remote server</td>
<td>Yes, but did not spot the right URL (obfuscated)</td>
</tr>
<tr>
<td>POSTs information</td>
<td>Yes, but not used in the malicious part</td>
</tr>
<tr>
<td>Retrieves IMEI, IMSI, operator, phone number</td>
<td>YES</td>
</tr>
</tbody>
</table>
### Android/PJapps: How accurate?

<table>
<thead>
<tr>
<th>Feature</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raising the alarm</td>
<td>YES</td>
</tr>
<tr>
<td>Sends SMS</td>
<td>YES</td>
</tr>
<tr>
<td>Contacts a remote server</td>
<td>Yes, but did not spot the right URL (obfuscated)</td>
</tr>
<tr>
<td>POSTs information</td>
<td>Yes, but not used in the malicious part</td>
</tr>
<tr>
<td>Retrieves IMEI, IMSI, operator, phone number</td>
<td>YES</td>
</tr>
<tr>
<td>Lists installed packages</td>
<td>YES</td>
</tr>
</tbody>
</table>
## Android/PJapps: How accurate?

<table>
<thead>
<tr>
<th>Activity</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raising the alarm</td>
<td>YES</td>
</tr>
<tr>
<td>Sends SMS</td>
<td>YES</td>
</tr>
<tr>
<td>Contacts a remote server</td>
<td>Yes, but did not spot the right URL (obfuscated)</td>
</tr>
<tr>
<td>POSTs information</td>
<td>Yes, but not used in the malicious part</td>
</tr>
<tr>
<td>Retrieves IMEI, IMSI, operator, phone number</td>
<td>YES</td>
</tr>
<tr>
<td>Lists installed packages</td>
<td>YES</td>
</tr>
<tr>
<td>Sends emails</td>
<td>Yes, but not used in the malicious part</td>
</tr>
</tbody>
</table>
Why does it work?
Limitations... by design?

False positives / negatives depend on threshold.

Score too high (false positive)

Prepay Widget - display plan’s balance - risk score: 36

- sends USSD commands: call property detector
- read incoming SMS for operator’s reply to USSD commands: SMS receiver detector
- Russian certificate: geographical detector
- Test if rooted (dialer in background): `Runtime.exec()` detector

Typically also for hacking, rooting, system tools.

Score too low (false negative)

- Fail to disassemble: code property detectors not run. Solution: use another tool.
- Very simple malware: triggers only few detectors
Limit Example: Android/SndApp

Android/SndApp
Collects IMEI, phone number, network country, operator’s name, email address of the victim.
Sends this to a remote web site.

▶ Retrieve IMEI, operator and phone number: DETECTED

Reads phone IMEI: getDeviceId spotted
Gets carrier: getNetworkOperator spotted
Gets phone number: getLine1Number spotted

▶ Retrieve network country: Not detected, but not sensible?
▶ Retrieve email addresses: TO DO
▶ URL information is sent to: DETECTED

Risk score: 12
Not enough detectors are raised.
Raise weight of these detectors?
Create combination detector for sending private data?
Future Work

- Performance: search in parallel or apply pre-filtering etc
- Adding / improving new detectors (e.g. use of AccountManager)
  - Searching for commands in executables (chmod, execve, mounting system partition) - NEW
  - Detect executables in ./lib - NEW
  - Detecting AOSP certificate - NEW
  - Combinations: concealing SMS with abortBroadcast, AOSP & INSTALL_PACKAGES NEW
  - Improve malicious URL detection: use prior work and apply to mobile world?
- Data mining to compute weights
- Test against larger sets
Thank You!

Axelle Apvrille
aapvrille@fortinet.com
http://blog.fortiguard.com
twitter: @cryptax

Tim Strazzere
strazz@gmail.com
http://www.strazzere.com/blog/
twitter: @timstrazz

Slides edited with \texttt{LOBSTER=\LaTeX+Beamer+Editor}