Reducing the Window of Opportunity for Android Malware

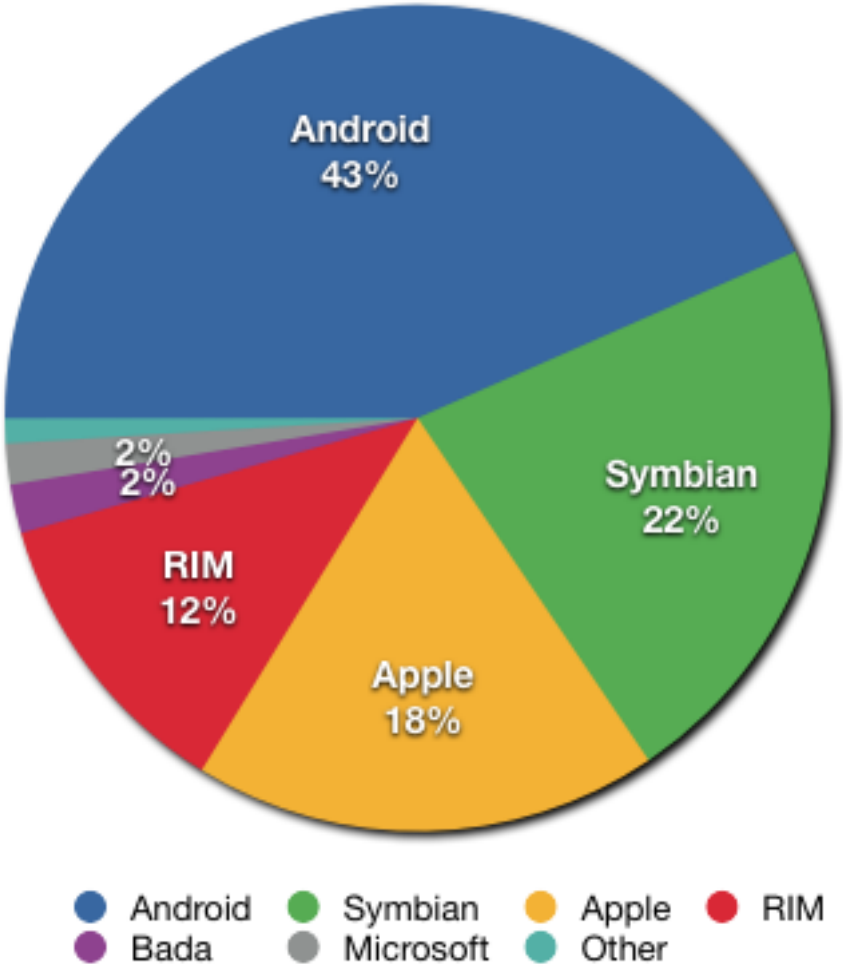
Gotta catch 'em all

Axelle Apvrille, Fortinet

Tim Strazzere, Lookout Mobile Security

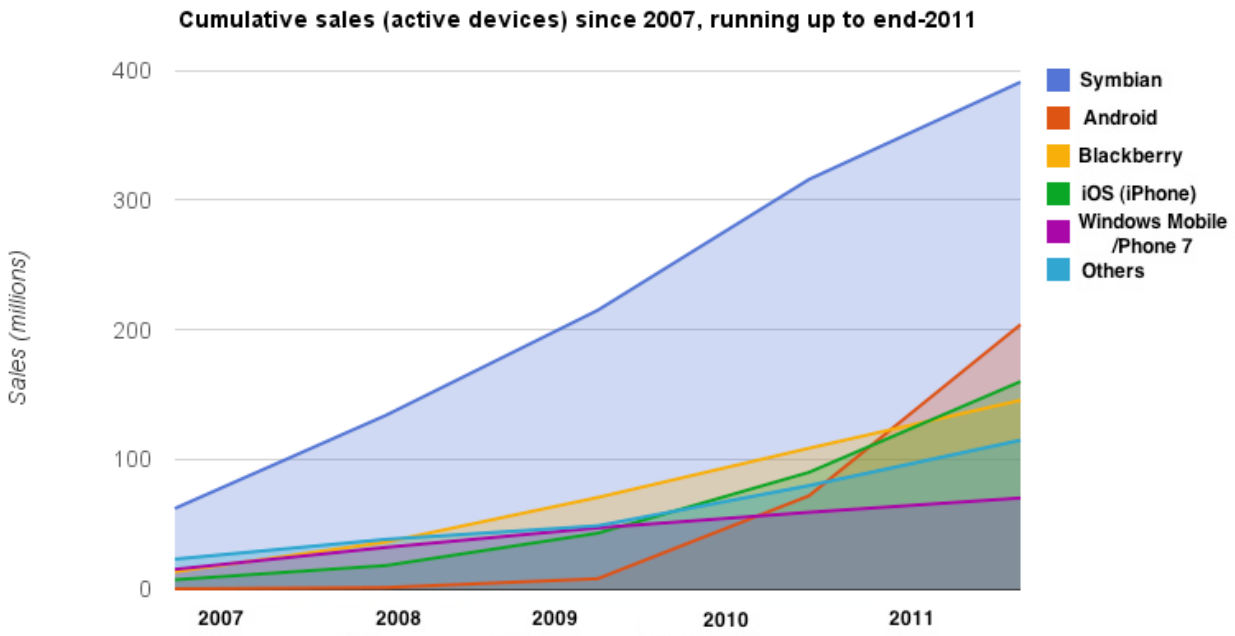
EICAR Conference, May 2012

Stats are difficult to compute



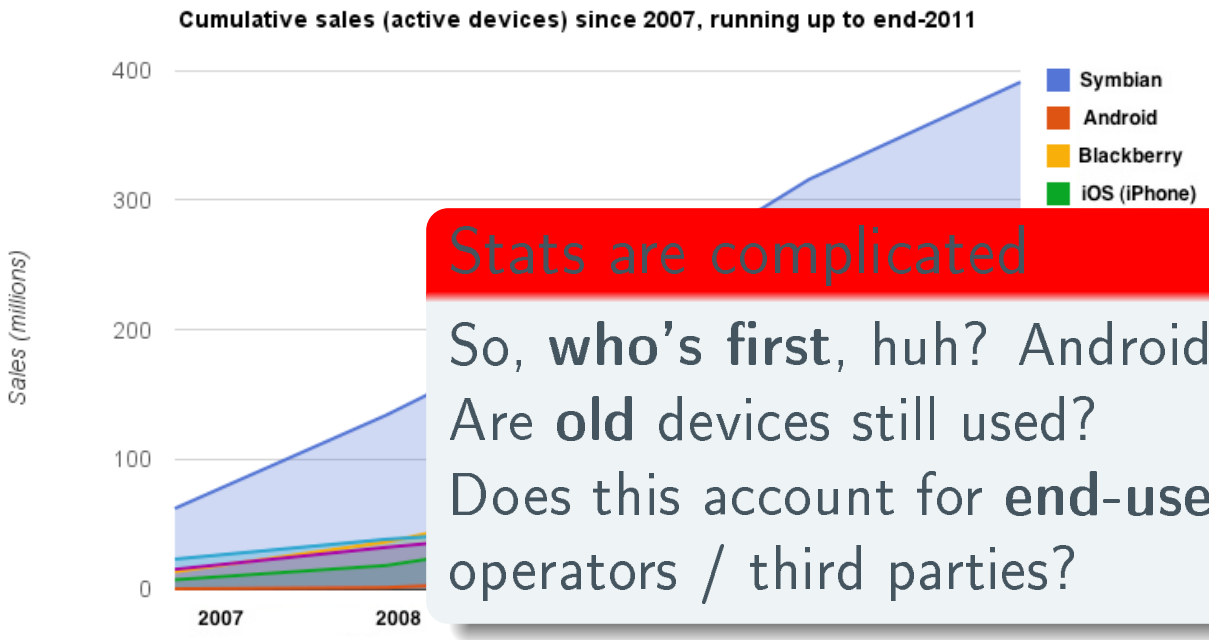
- Q2 2011 sales [Source: Gartner]

Stats are difficult to compute



- Cumulative sales [Source: David Litchfield]

Stats are difficult to compute

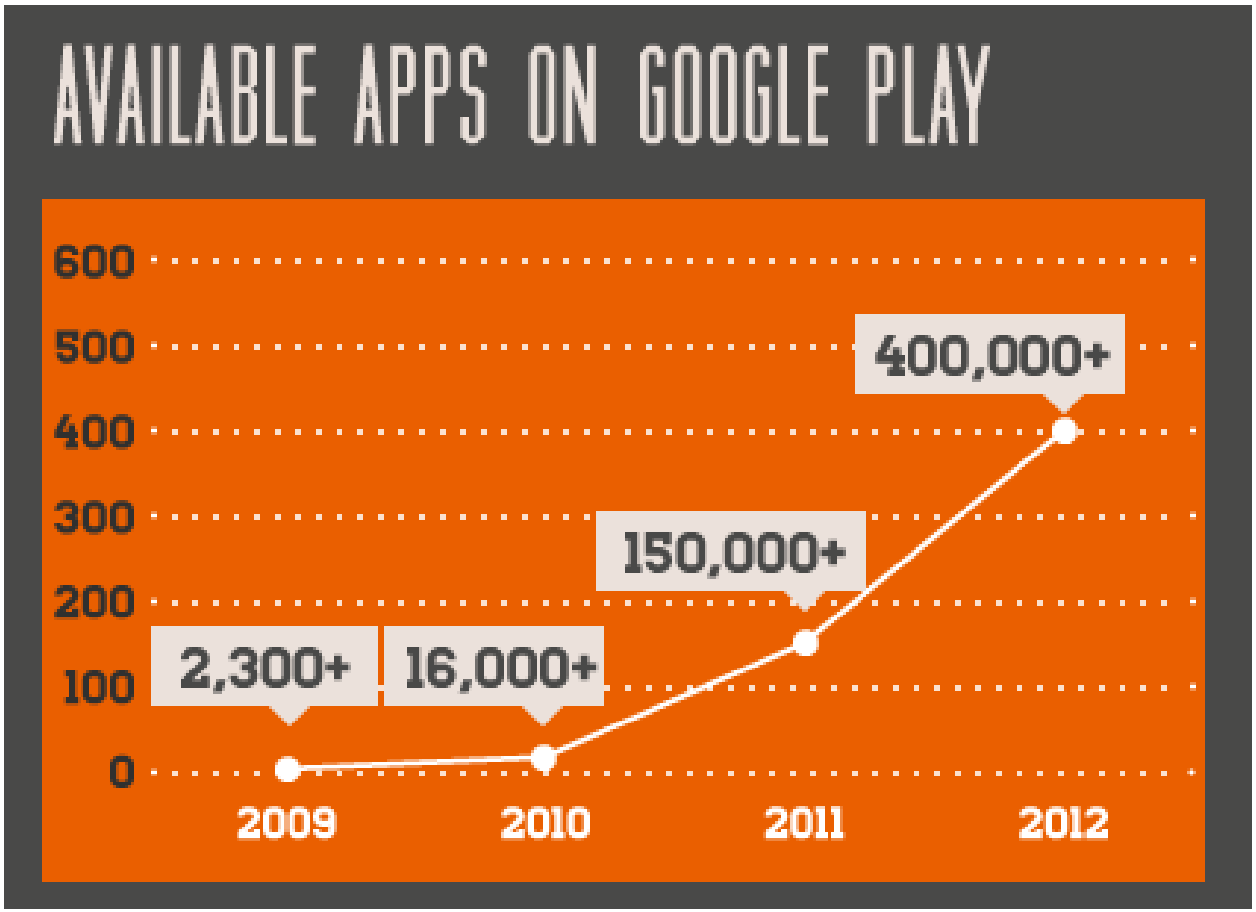


- Cumulative sales [Source: David Litchfield]

Stats are complicated

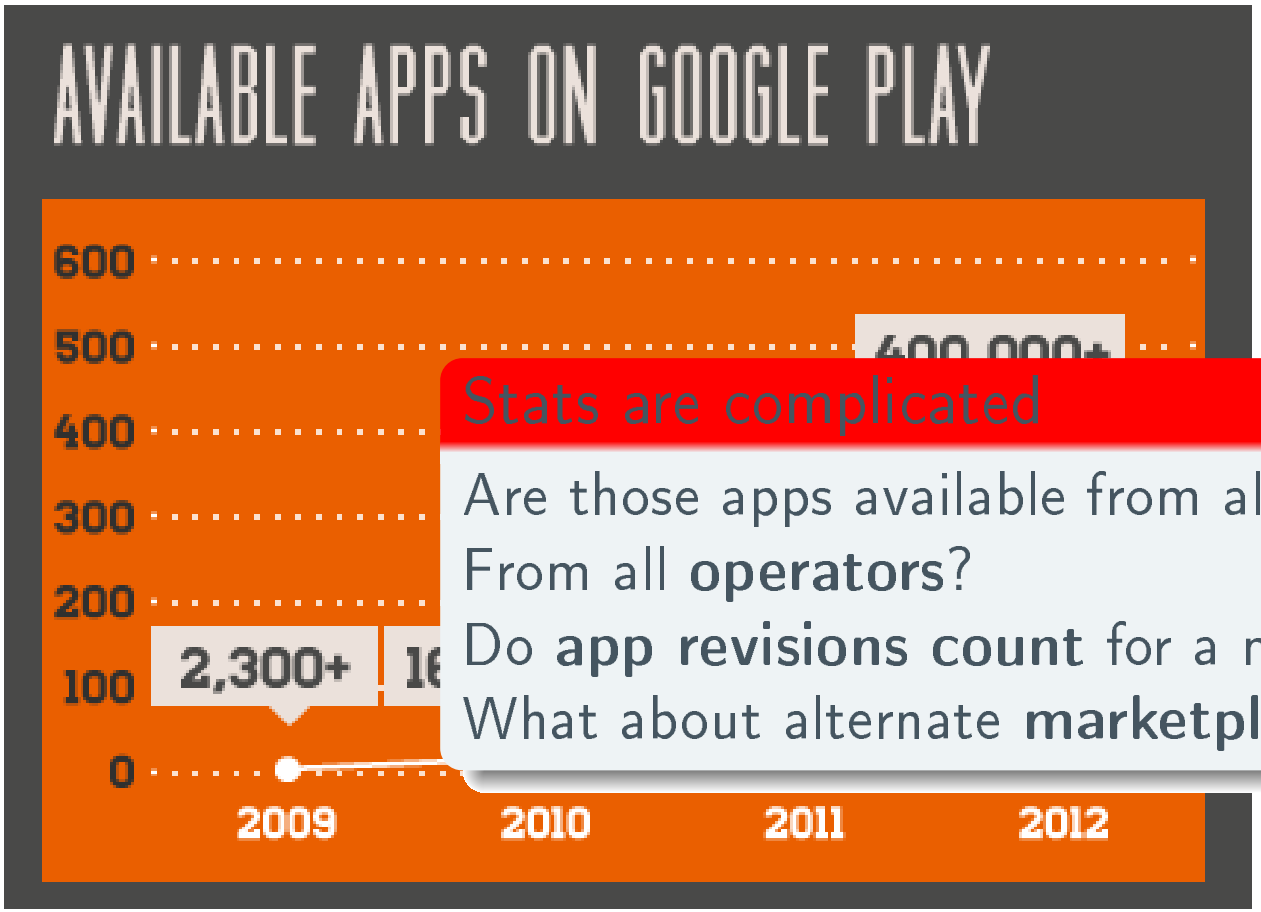
So, **who's first**, huh? Android? Symbian?
Are **old** devices still used?
Does this account for **end-user sales** or sales to operators / third parties?

Stats are difficult to compute



- Available apps [Source: Lookout]

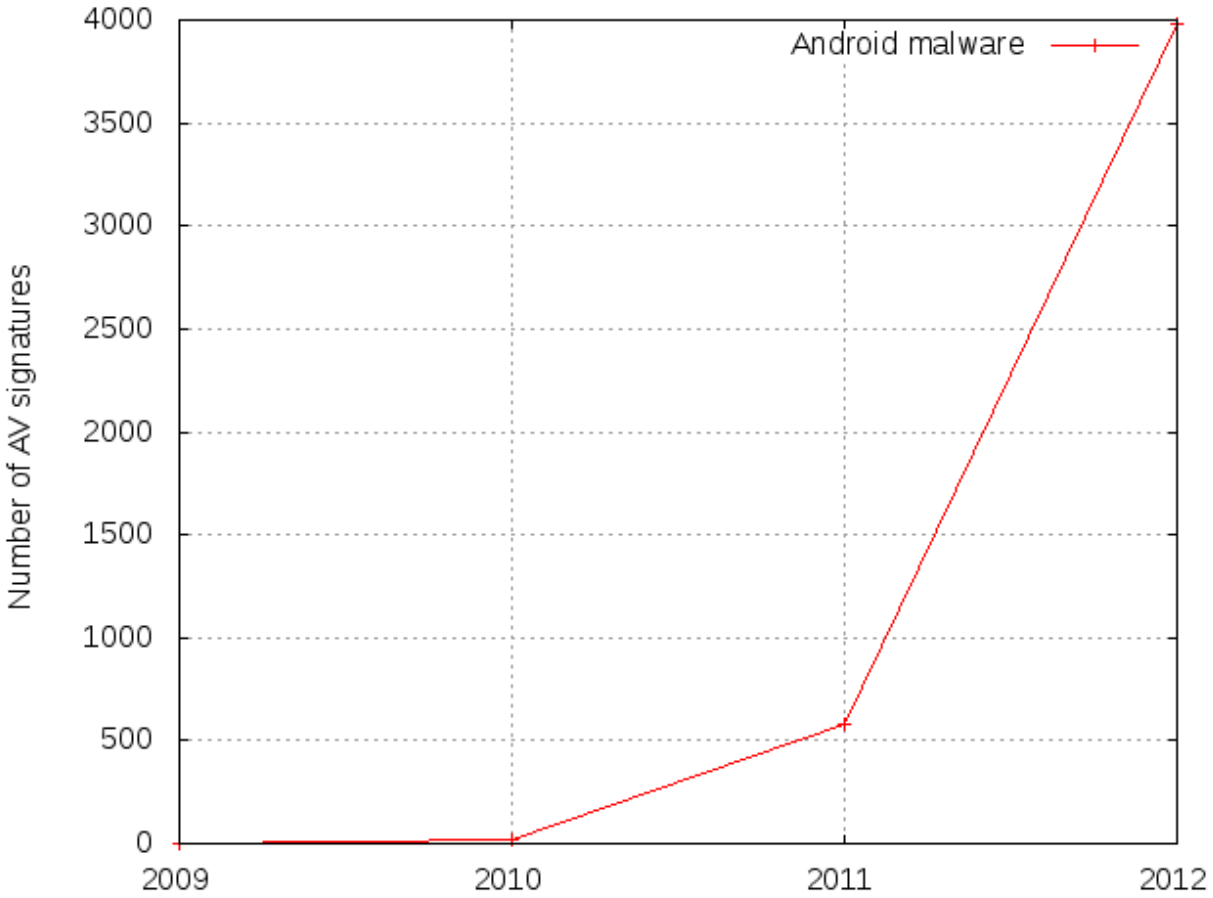
Stats are difficult to compute



Stats are complicated

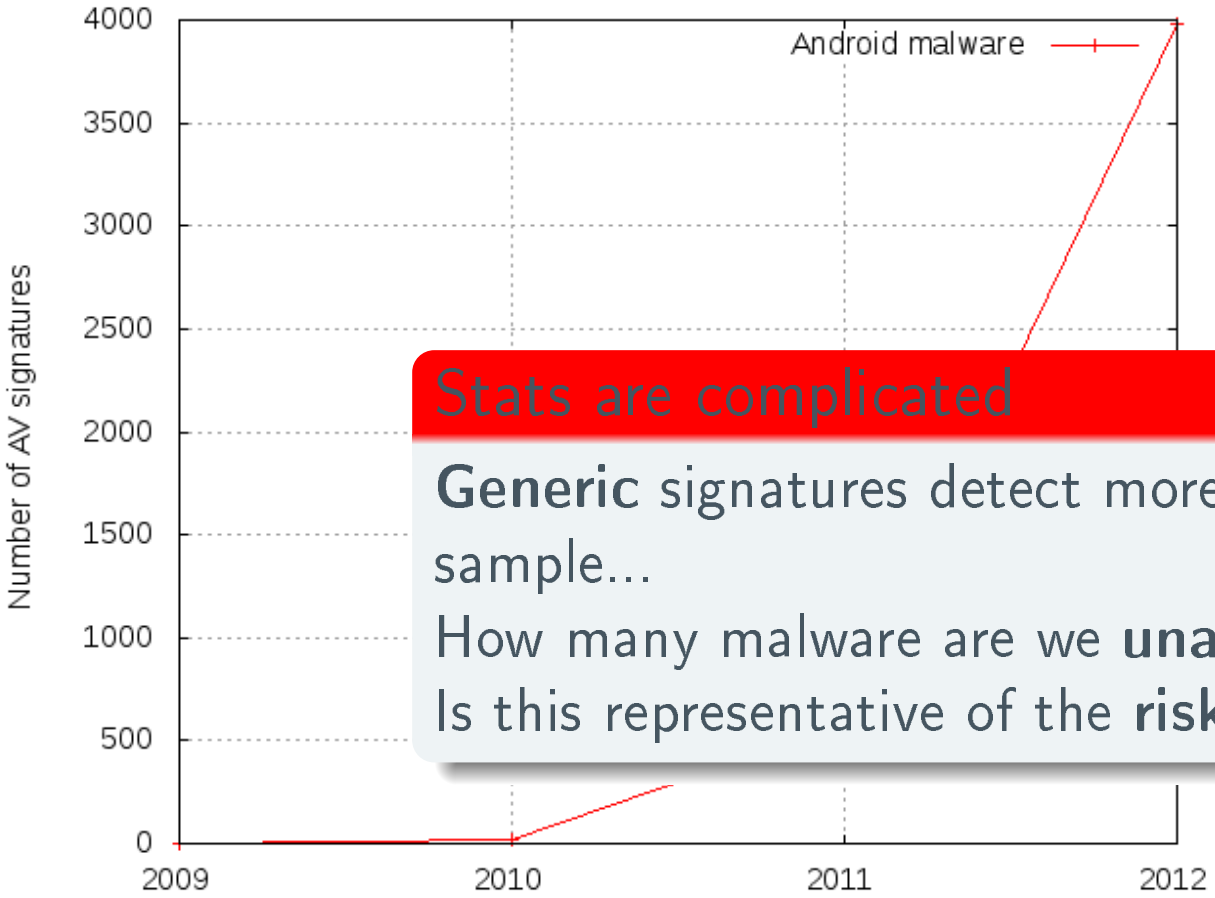
Are those apps available from all **countries**?
From all **operators**?
Do **app revisions count** for a new app?
What about alternate **marketplaces**?

Stats are difficult to compute



- Nb of signatures [Source: Fortinet]

Stats are difficult to compute



Stats are complicated

Generic signatures detect more than one sample...

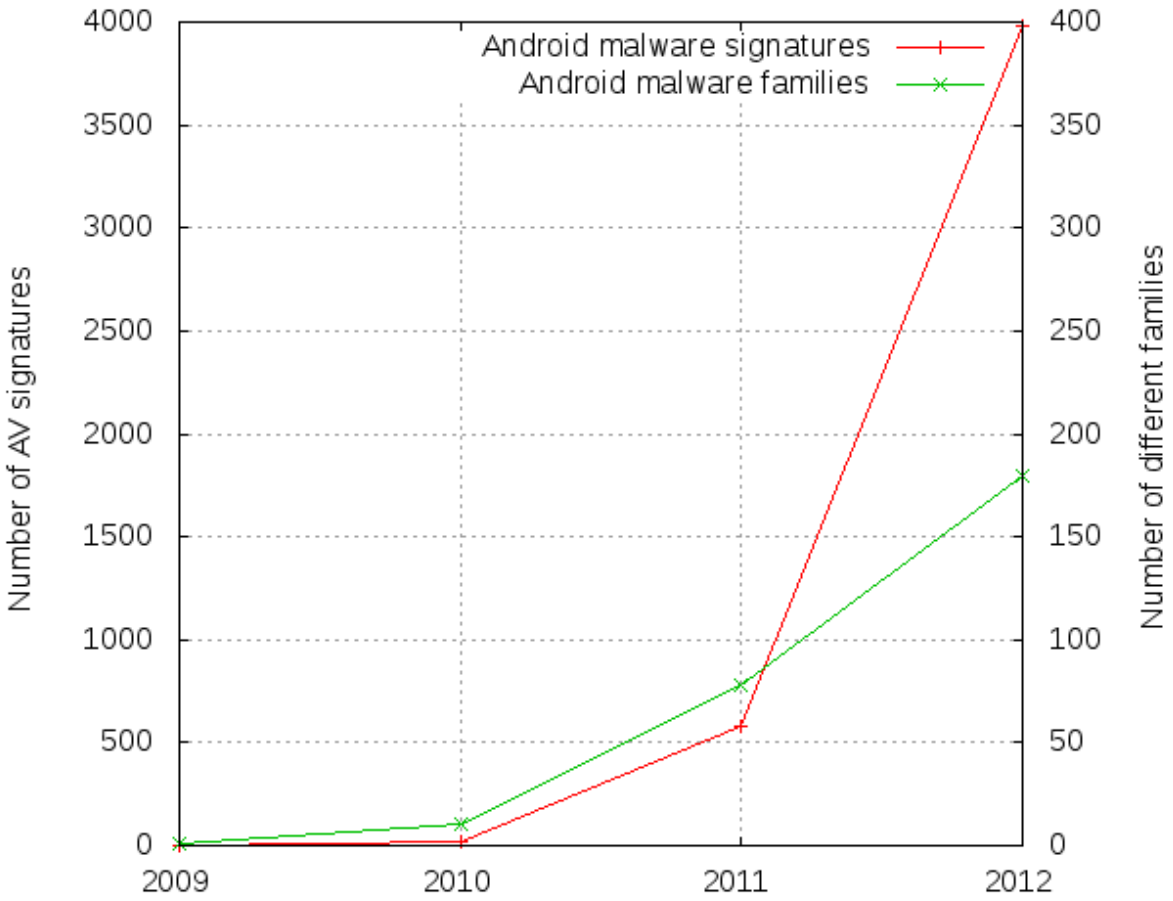
How many malware are we **unaware** of?

Is this representative of the **risk** for end-users?

res

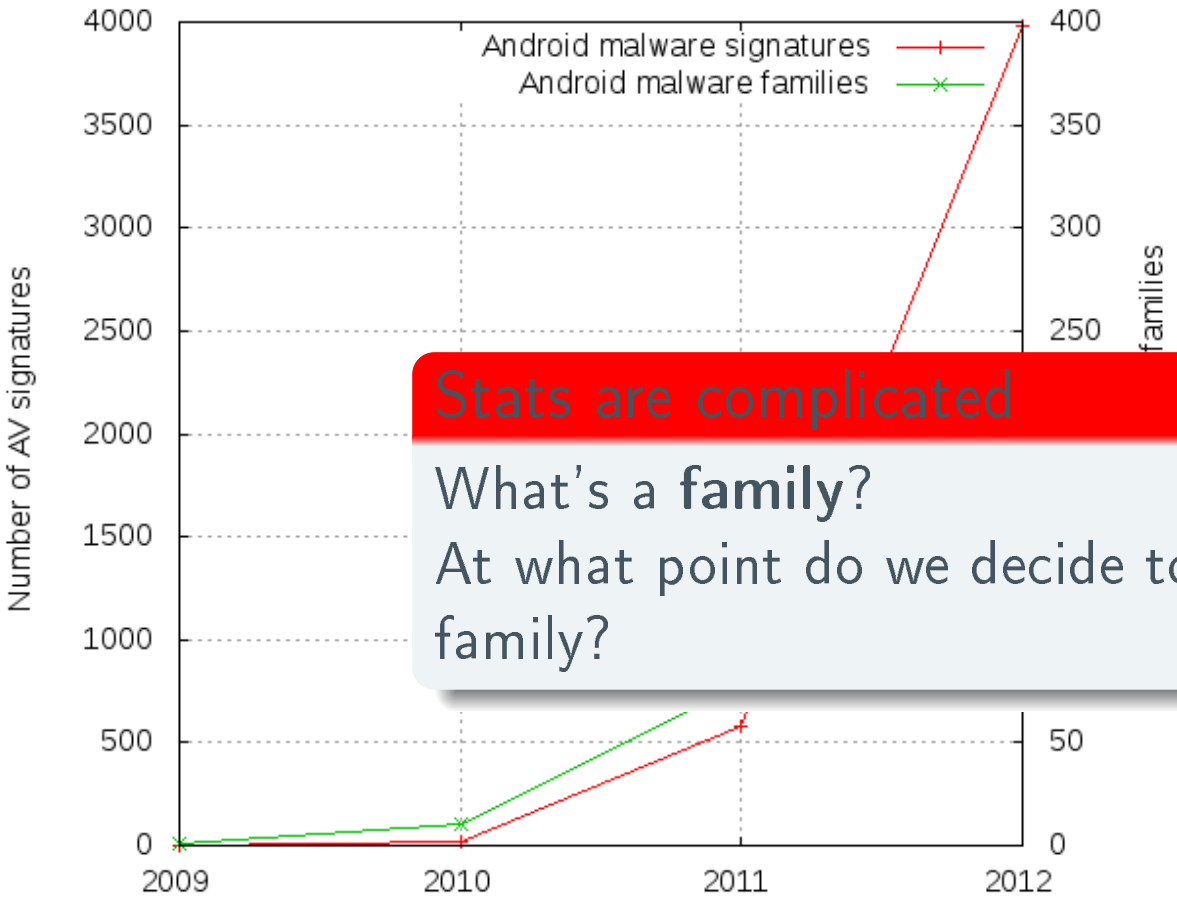
net]

Stats are difficult to compute



- Nb of different families [Source: Fortinet]

Stats are difficult to compute



Stats are complicated

What's a **family**?

At what point do we decide to create a new family?

- Nb of different families [Source: Fortinet]

Our perception of world depends on our knowledge

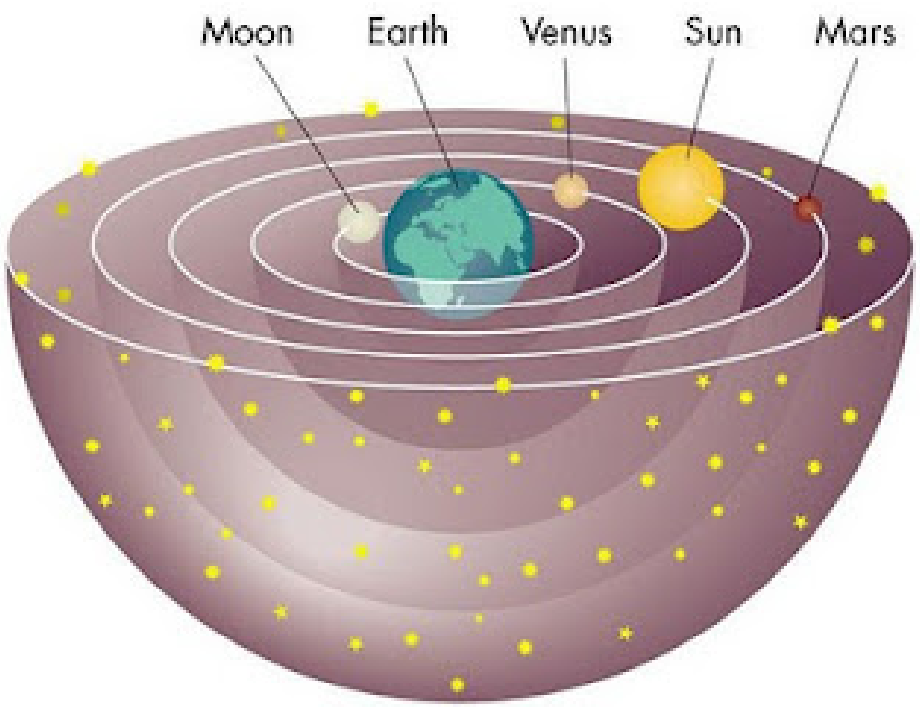


Figure: Aristotle's Universe (source: AlienCitadel)

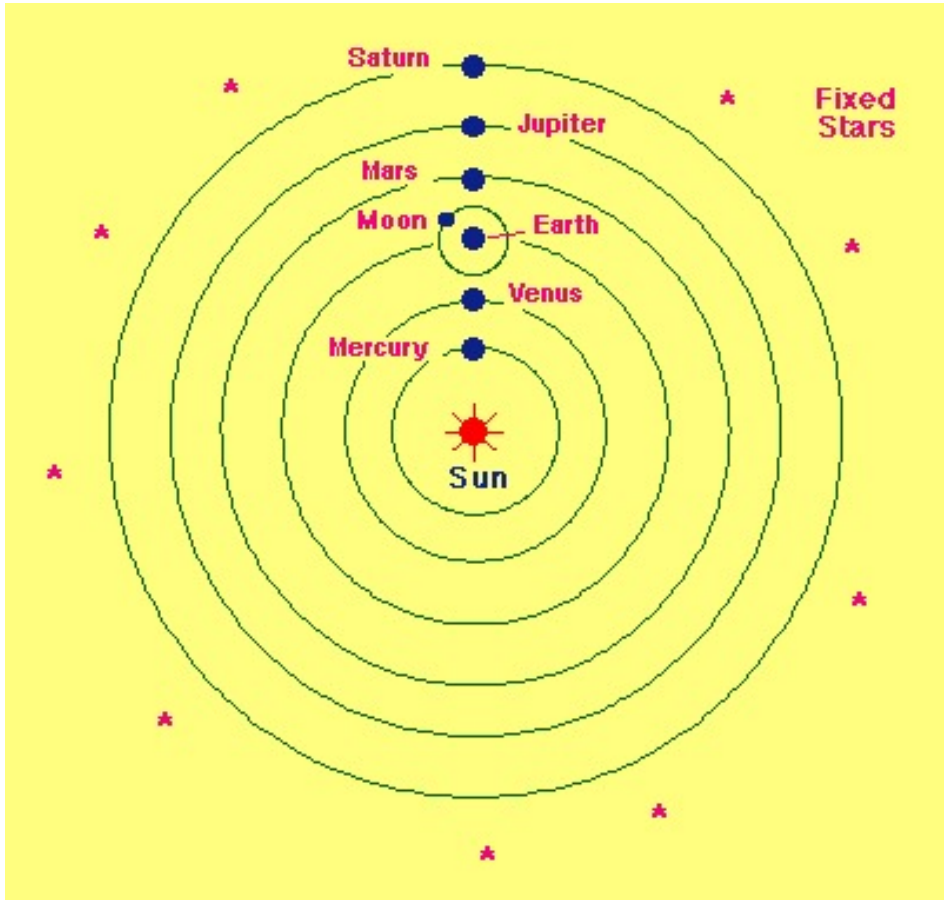
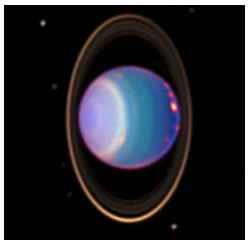
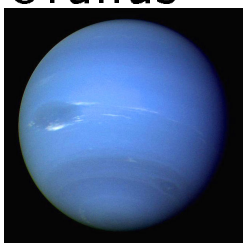
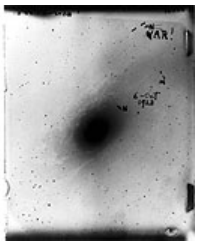


Figure: The Copernican Universe (source: Tpellk)

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
...
```

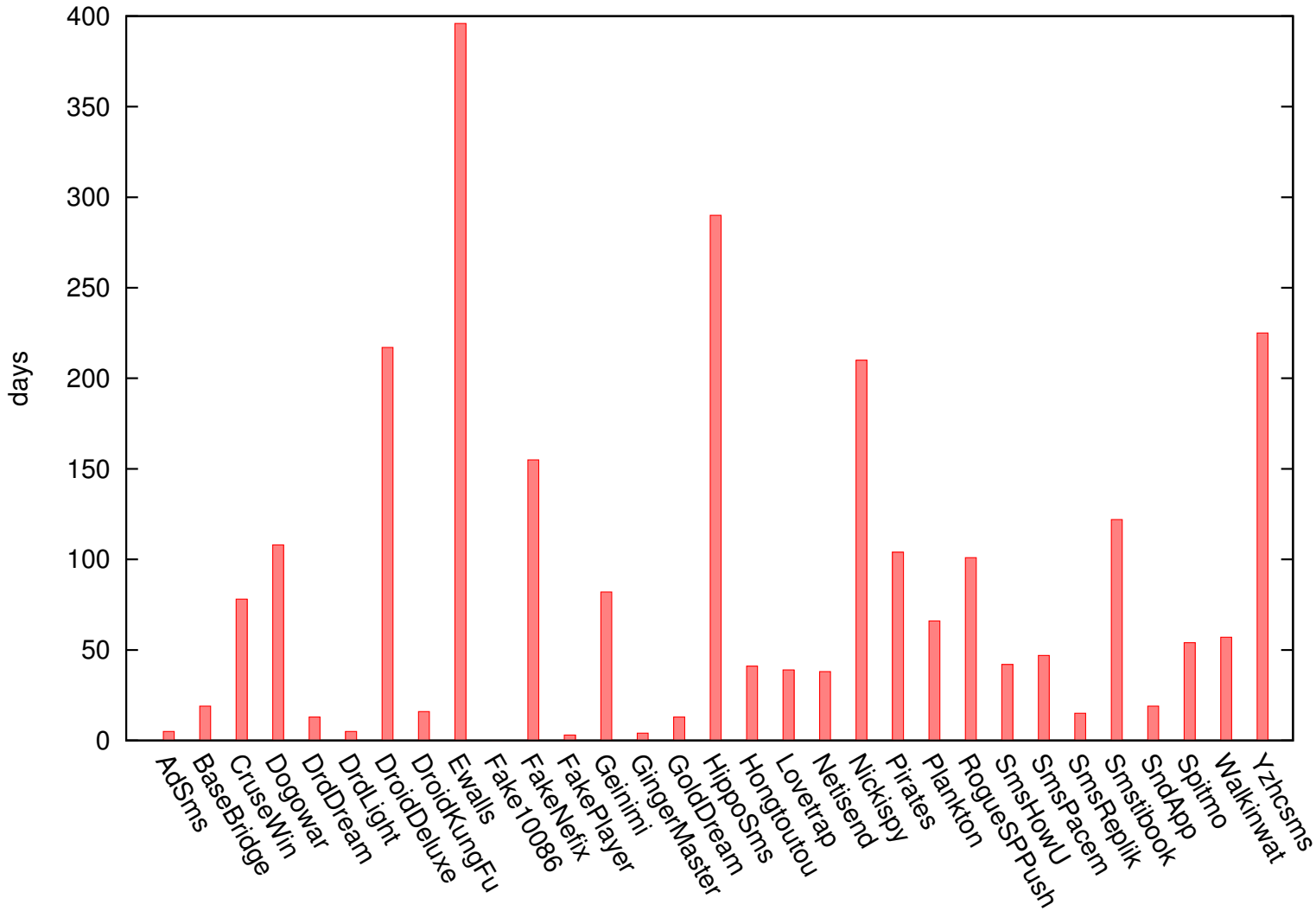
- **Approximation**. Day the certificate was created.
- Does not work for AOSP keys.

Package's zip date

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

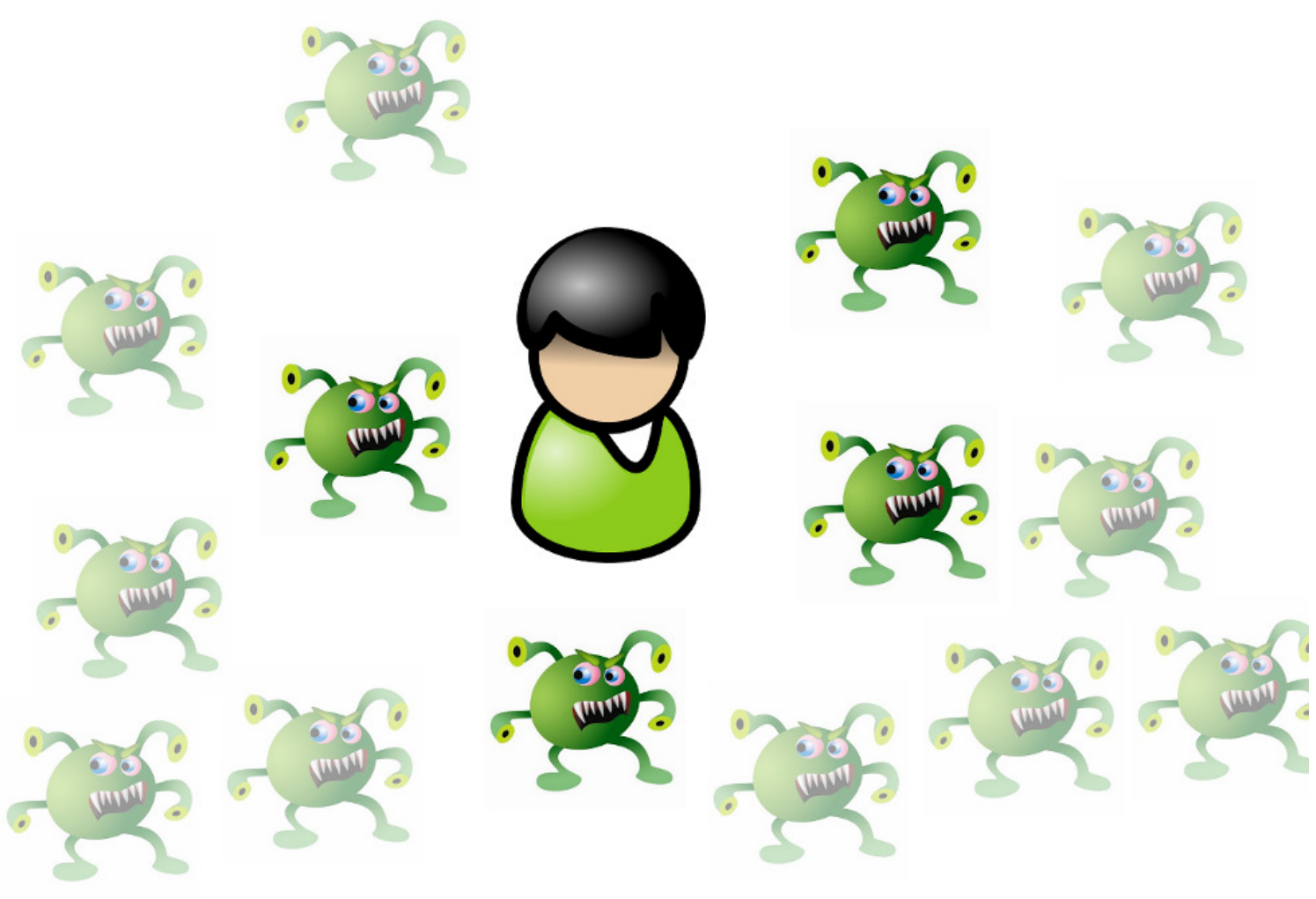
- **Dec 20** (2011): **Approximate and unsecure**.
- But... gives better results

Release date vs Detection date



Average: **80** days after release!

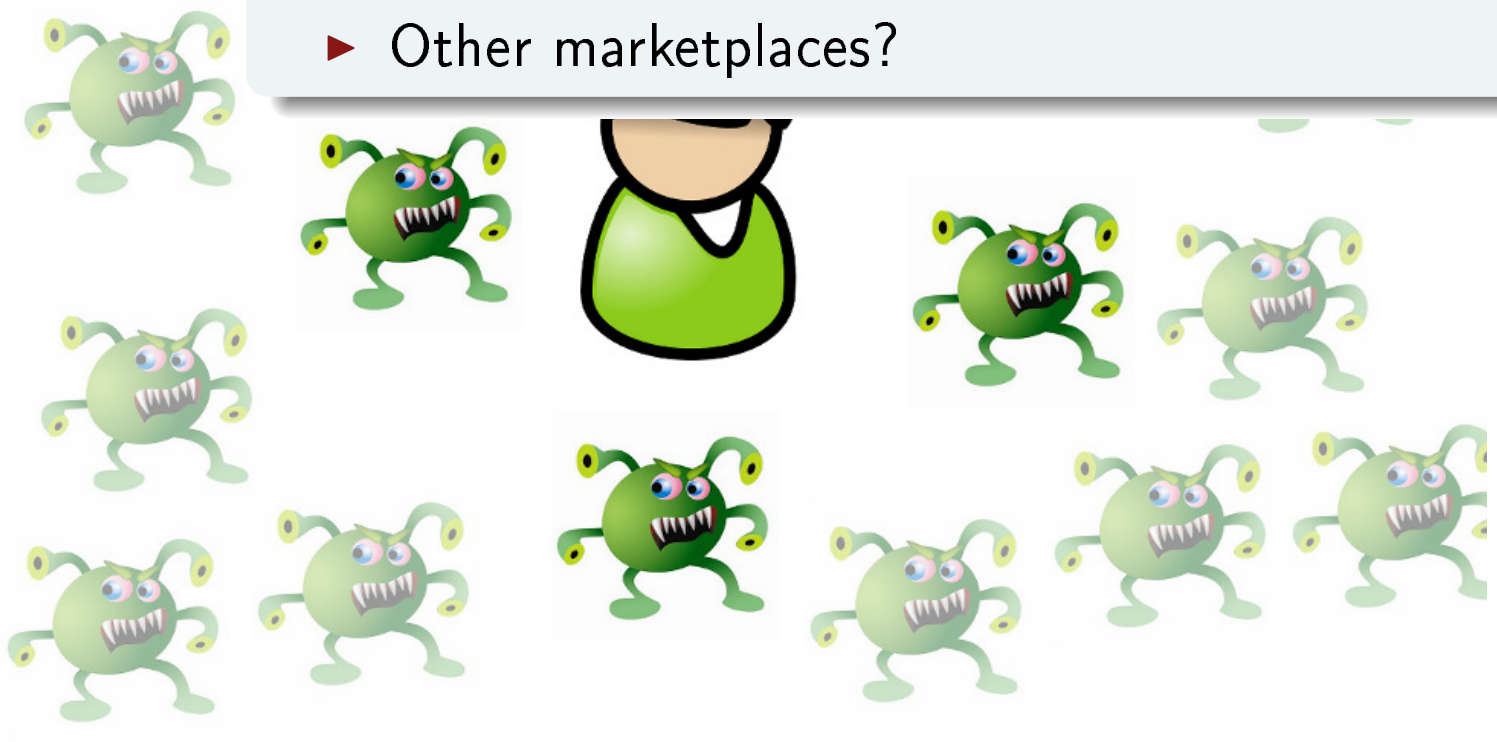
Why are we missing malware?



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?



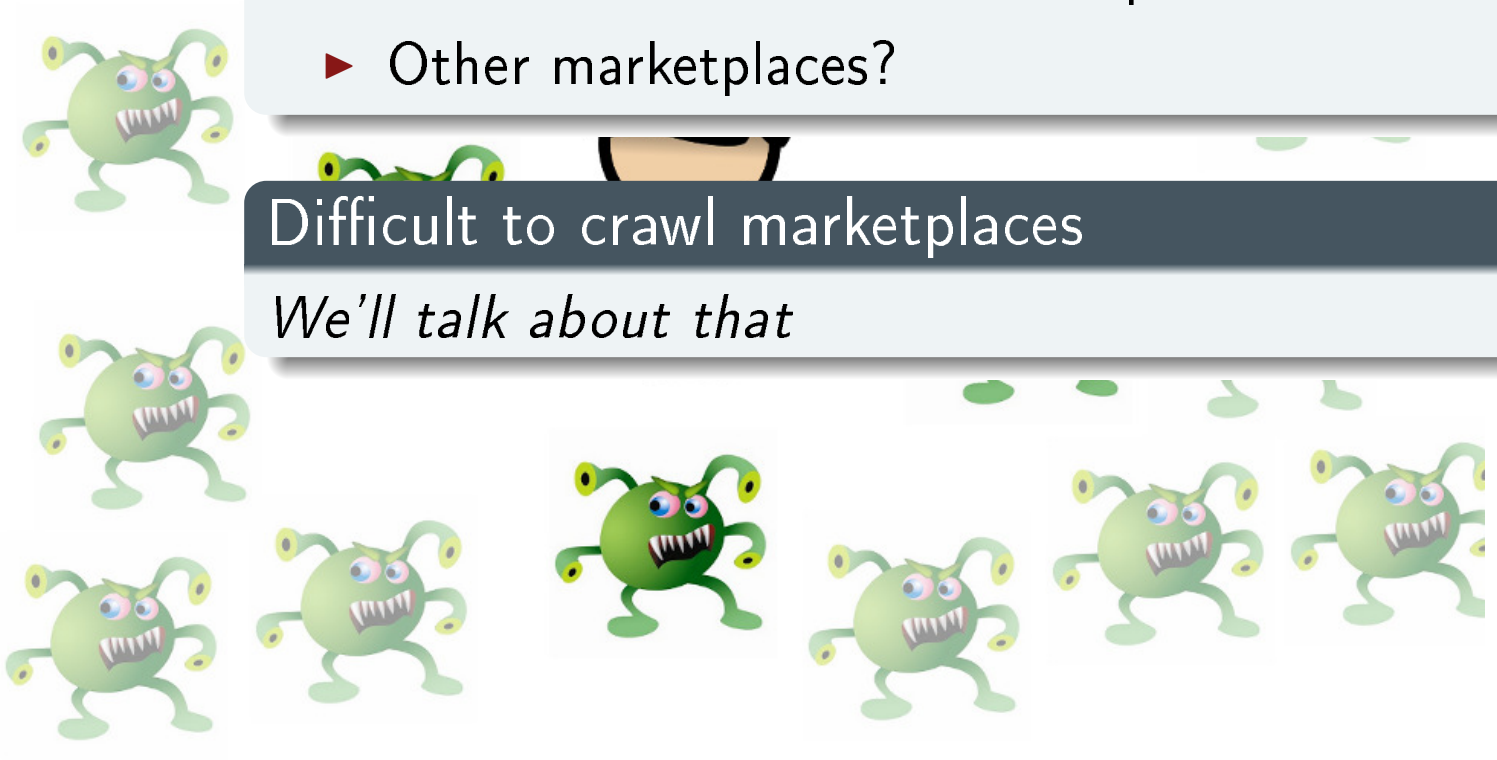
Why are we missing malware?

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

Google Play (Android Market) Crawling

Once a crawler - always a crawler, right?

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

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

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?

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?
- 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 $\rightarrow$ Receive device specifics
- Get an auth-token $\rightarrow$ refresh every two weeks
- Store accounts in DB for later use in metadata / download retrieval

Catching' em all

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

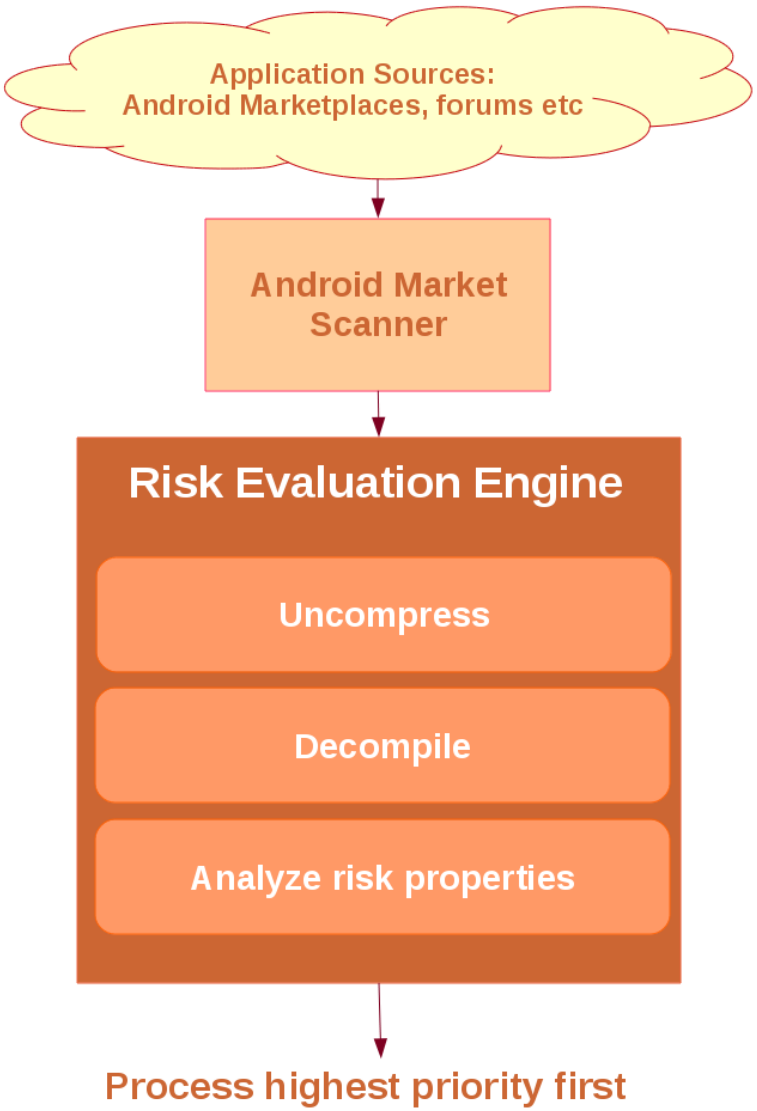
Downloading the APKs

- 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

Property detectors

What is a property detector?

- Detect risky situations
- Static check against the package
- (Relatively) simple test
- States a tendancy, never guarantees clean/malicious

Detector examples

- Use of AOSP signing certificate $\rightarrow$ Risk for users with custom ROM
- Call to `Runtime.exec()` $\rightarrow$ Run Unix commands, e.g `pm install`

40+ property detectors

Type	Example	Location
Permissions	SEND_SMS, RECEIVE_SMS ...	Manifest

40+ property detectors

Type	Example	Location
Permissions	SEND_SMS, RECEIVE_SMS ...	Manifest
API calls	`sendTextMessage()`, `getDeviceId()`, `DexClassLoader`, `KeySpec` ...	Code

40+ property detectors

Type	Example	Location
Permissions	SEND_SMS, RECEIVE_SMS ...	Manifest
API calls	`sendTextMessage()`, `getDeviceId()`, `DexClassLoader`, `KeySpec` ...	Code
Hidden executables	ARM, zip, jar	assets, res/raw, lib

40+ property detectors

Type	Example	Location
Permissions	SEND_SMS, RECEIVE_SMS ...	Manifest
API calls	`sendTextMessage()`, `getDeviceId()`, `DexClassLoader`, `KeySpec` ...	Code
Hidden executables	ARM, zip, jar	assets, res/raw, lib
Unix commands	`pm install` ...	Executables

40+ property detectors

Type	Example	Location
Permissions	SEND_SMS, RECEIVE_SMS ...	Manifest
API calls	`sendTextMessage()`, `getDeviceId()`, `DexClassLoader`, `KeySpec` ...	Code
Hidden executables	ARM, zip, jar	assets, res/raw, lib
Unix commands	`pm install` ...	Executables
Geography	+86 ...	classes.dex, certificate

40+ property detectors

Type	Example	Location
Permissions	SEND_SMS, RECEIVE_SMS ...	Manifest
API calls	`sendTextMessage()`, `getDeviceId()`, `DexClassLoader`, `KeySpec` ...	Code
Hidden executables	ARM, zip, jar	assets, res/raw, lib
Unix commands	`pm install` ...	Executables
Geography	+86 ...	classes.dex, certificate
URL detectors	C&Cs	Executabes

40+ property detectors

Type	Example	Location
Permissions	SEND_SMS, RECEIVE_SMS ...	Manifest
API calls	`sendTextMessage()`, `getDeviceId()`, `DexClassLoader`, `KeySpec` ...	Code
Hidden executables	ARM, zip, jar	assets, res/raw, lib
Unix commands	`pm install` ...	Executables
Geography	+86 ...	classes.dex, certificate
URL detectors	C&Cs	Executabes
Package properties	Size, AOSP platform cert	Package

40+ property detectors

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

Writing property detectors

Make a call

In AndroidManifest.xml:

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

```
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

```
$grep = `egrep -rl 'KeySpec|SecretKey|Cipher'
  "$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;
} ...
```

Weight

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

Filename	Risk	Internet	SMS	MMS	Install	Store	Enc	GPS	Version
./3Banana.notes.apk	11	**Yes**						**Yes**	1.5
./47_32590_11073013120108667jng9o1.apk	22	**Yes**	**Yes**					**Yes**	
./AdvancedTaskManager-3.7-.apk	4								1.5
./DroidBreakout_v1.-.4.apk	0								
./DungeonHunter.working.apk	3	**Yes**							2.0

Risk Evaluation Engine Output

Automatic Analysis Report

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

									ersion
./3Banana.notes.apk	11	**Yes**						**Yes**	1.5
./47_32590_11073013120108667jng9o1.apk	22	**Yes**	**Yes**					**Yes**	
./AdvancedTaskManager-3.7-.apk	4								1.5
./DroidBreakout_v1.-.4.apk	0								
./DungeonHunter.working.apk	3	**Yes**							2.0

Highest scores

Sample	Score	Name
7734626341799e6ec8c3db21722b	61	Android/DroidKung-Fu.B!tr
0f2375e7c3239b569a0b0322261b	58	Android/Pjapps.B!tr
com.swampy.sexpos.apk	58	Android/Geinimi.A!tr
Andr_PJApps-Gen_f051eeab57e42d5...apk	57	Android/Pjapps.A!tr
jeecalendar.apk	56	Android/CrazyVampire-.A!tr
0091556ed96b3b5aa0af62e70751	54	Android/DroidKungFu-.D!tr
BatterySaver.apk	52	Android/FakeDoc.A!tr
6_35228_1c0a6b1c5d24cbba9b1	51	Android/DroidCoupon-.A!tr
golddream_sample.apk	51	Android/GoldDream.A!tr

Detailed output for Android/PJapps

./steamy-PJAPPS-iNFECTED.apk	52	**Yes**	**Yes**		

```
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://xxxxxxxxx9:8618/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
```

Detailed output for Android/PJapps

./steamy-PJAPPS-iNFECTED.apk	52	**Yes**	**Yes**		

```
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
```

Android/PJapps: How accurate?

Raising the alarm	YES

Android/PJapps: How accurate?

Raising the alarm	YES
Sends SMS	YES

Android/PJapps: How accurate?

Raising the alarm	YES
Sends SMS	YES
Contacts a remote server	Yes, but did not spot the right URL (obfuscated)

Android/PJapps: How accurate?

Raising the alarm	YES
Sends SMS	YES
Contacts a remote server	Yes, but did not spot the right URL (obfuscated)
POSTs information	Yes, but not used in the malicious part

Android/PJapps: How accurate?

Raising the alarm	YES
Sends SMS	YES
Contacts a remote server	Yes, but did not spot the right URL (obfuscated)
POSTs information	Yes, but not used in the malicious part
Retrieves IMEI, IMSI, operator, phone number	YES

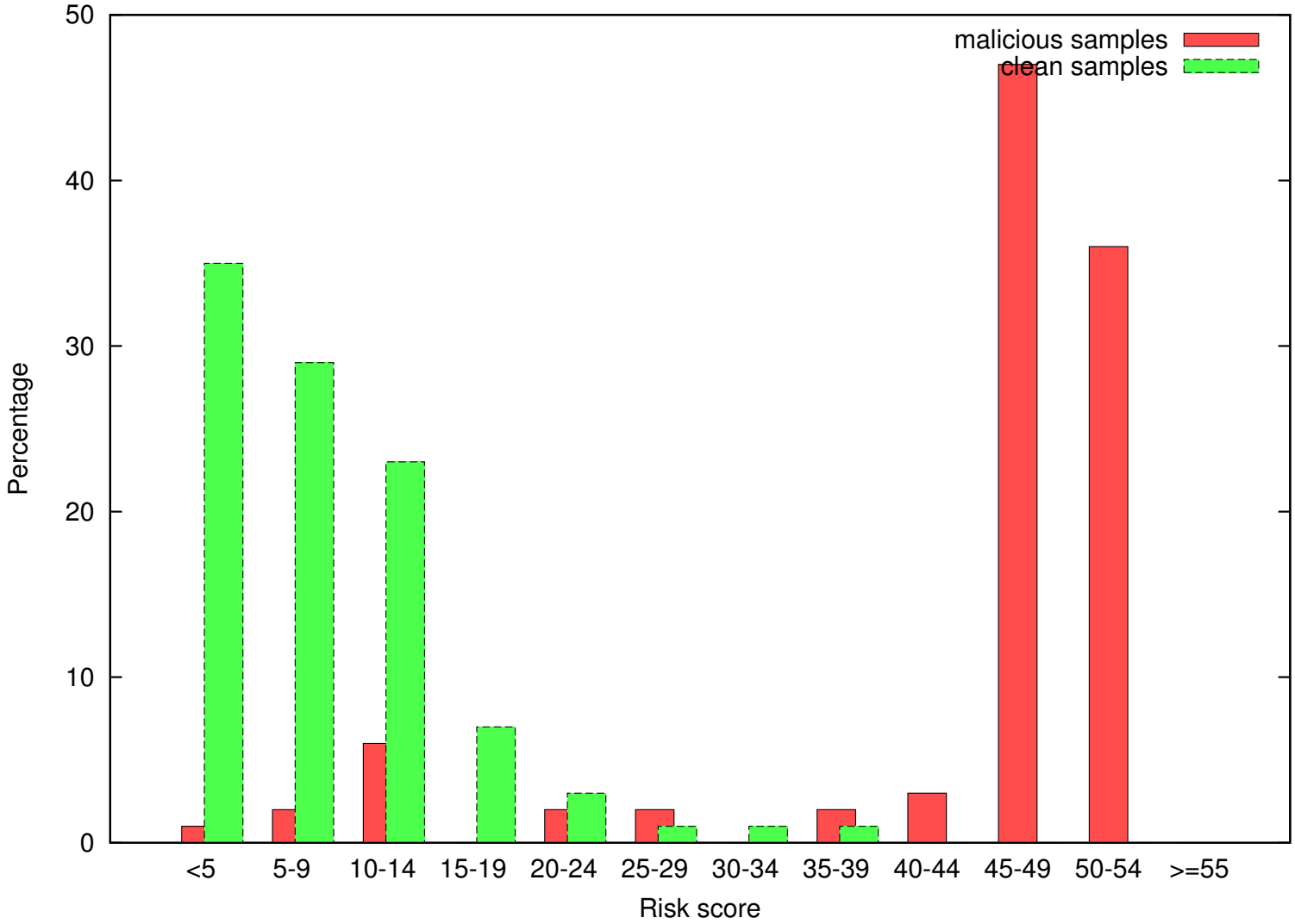
Android/PJapps: How accurate?

Raising the alarm	YES
Sends SMS	YES
Contacts a remote server	Yes, but did not spot the right URL (obfuscated)
POSTs information	Yes, but not used in the malicious part
Retrieves IMEI, IMSI, operator, phone number	YES
Lists installed packages	YES

Android/PJapps: How accurate?

Raising the alarm	YES
Sends SMS	YES
Contacts a remote server	Yes, but did not spot the right URL (obfuscated)
POSTs information	Yes, but not used in the malicious part
Retrieves IMEI, IMSI, operator, phone number	YES
Lists installed packages	YES
Sends emails	Yes, but not used in the malicious part

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

Limits 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 LOBSTER=LaTeX+*Beamer* + *Editor*