Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets by Yajin Zhou, et al. (*NDSS*, 2012)

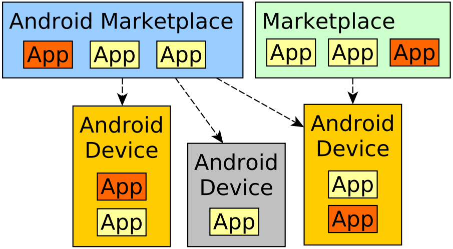
presented by
Jedidiah R. McClurg

Northwestern University

May 7, 2012

Background

- Smartphones are becoming increasingly common
  - Over 100 million sold in early 2011
  - Over 200K apps in Android (Google) Marketplace
- A recent survey of mobile malware [2] shows that malware is becoming more common in Android marketplaces



- The survey finds that Android is a natural target for malware, due to its openness/customizability, and lack of app regulation
- It is important to respond by assessing the *overall health* of the marketplaces in terms of the malware present

Motivation

- How can we obtain this global health measurement?
- One approach is to automatically *crawl* the marketplaces, download free apps, and perform malware detection
- This has been done for limited subsets of marketplace apps, but large-scale analysis is needed to obtain a better understanding of the global Android malware status
- There are some considerations needed in this approach:
  - Accuracy – we need low false negatives/positives
  - Efficiency and Scalability – at 6 seconds per sample, a collection of 200K apps would take over two weeks to analyze, so speed is very important
- A key idea is to rapidly *filter* apps which are unlikely to be malware, leaving only a small core to analyze
- The listed survey paper shows that *permissions* form a good indicator of maliciousness – this can be leveraged in a filter

Motivation (Cont.)

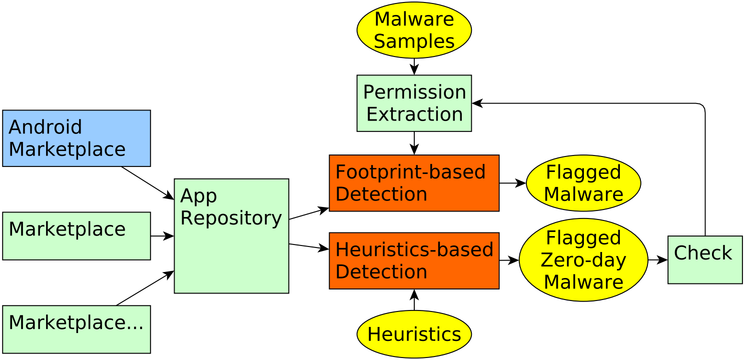
- **DroidRanger** [4] is a malware detection system based on this approach
- It has been used to crawl over 200K apps in several marketplaces (approx. 150K from the official Google marketplace)
- DroidRanger has two main functions
  - Detecting *known* malware via **permission-based behavioral footprinting**
  - Detecting *unknown* malware via **heuristics-based filtering**
- Found 171 infected apps (21 of them from the Google marketplace) and 2 unknown (zero-day) malware
- With "feedback" of unknown malware, found 211 infected apps total

Related Work

- Smartphone security
  - Systems that reveal *privacy* leaks (e.g. TaintDroid)
  - Systems that block certain *unsafe actions*, essentially providing finer-grained permissions (e.g. AppFence [3])
  - Study of 1,100 top free apps [1] which compiles information about android security issues but does not offer a malware detection system
- Mobile malware detection
  - Several systems run on the phone and detect certain behaviors
  - The DroidRanger approach operates offline (faster, more scalable)

DroidRanger System Architecture

- Five app marketplaces are crawled: Android Market (Google), eoeMarket, alcatelclub, gfan, mmoovv



- Over 200K Apps are loaded into a database and sent to the two DroidRanger modules (higlighted)

DroidRanger System Architecture (Cont.)

DroidRanger performs the following tasks:

1. Detecting known malware via *permission-based behavioral footprinting*
   - Filters based on permissions, then analyzes based on behavior
   - Uses a set of 10 known malware families as footprints
2. Detecting unknown malware via *heuristics-based filtering*
   - Filtering based on dynamic code loading/execution and native code use
   - Analysis based on dynamic monitoring of the execution
   - Confirmed malware are fed back to step 1

Detecting Known Malware

First we discuss **detecting known malware** in detail

Step I. Permission-based filtering

- This is intended to quickly exclude unrelated applications
- It works by matching each app's manifest permissions against permissions requested by known malware
- Only applications which need these "malware-friendly" permissions are included in the malware analysis
- For example, Zsone malware asks for RECEIVE_SMS and SEND_SMS, and DroidRanger focuses in on apps which request these two permissions...

Detecting Known Malware (Cont.)

- This "filtering" reduces the analysis work significantly:

Permission	RECEIVE_SMS	SEND_SMS	(both permissions)
Apps	5,214	8,235	3,204
Percentage	2.85%	4.50%	1.75%

- Note: it's important to select the *distinguishing* permissions, otherwise we can get many false negatives/positives
- For example, all variants of the Pjapps malware request INTERNET and RECEIVE_SMS, but only some variants request WRITE_HISTORY_BOOKMARKS (thus, we would use the former two, and not the latter)

Detecting Known Malware (Cont.)

Step II. Behavioral analysis

- After the filtering, there are potentially still thousands of apps left to analyze
- An attempt to run off-the-shelf mobile antivirus at this point missed 23.52% of malware, probably due to signature polymorphism
- Instead, DroidRanger analyzes app *behavior*
  - App Manifest contains useful info (e.g. receivers)
  - App bytecode contains info (e.g. calls to send SMS)
  - Hierarchical structure of decompiled code contains useful info

Detecting Known Malware (Cont.)

- For example, the Zsone trojan exhibits the following behaviors:
    - The app contains a receiver that listens to android.provider.Telephony.SMS_RECEIVED and calls abortBroadcast
    - The app sends SMS messages to certain "premium" numbers, such as "10621900" and "106691819"
    - The app receives and intercepts SMS messages from certain numbers, such as "10086" and "10000"
- DroidRanger is able to find 9 instances of this malware, based on this behavior

Detecting Unknown Malware

Now we discuss **detecting unknown malware** in detail

Step I. Heuristic-based filtering

- DroidRanger takes a heuristic-based approach to detecting *unknown* malware
- The first heuristic involves looking for dynamic loading of untrusted code (for example, use of DexClassLoader)
- This type of dynamic loading is present in 1,055 apps (0.58%), mostly for ads
- Discovered Plankton spyware this way

Detecting Unknown Malware (Cont.)

- The second heuristic involves looking for suspicious native code
- Out of all the apps studied, 8,272 (4.52%) use native code
- The app-specific directory lib/armeabi is the default place for native code
- Some apps try to hide native code in other places

Apps w/ native code	Code in "assets"	Code in "res"
8,272 (4.52%)	313 (0.17%)	195 (0.11%)

- Discovered DroidKungFu malware this way

Detecting Unknown Malware (Cont.)

Step II. Dynamic execution monitoring

- Dynamically execute the apps uncovered by step I
- For example, during a call to SmsManager.sendTextMessage, the analysis can get the destination phone number and content
- Log questionable system calls, e.g. sys_mount, a command which can be used to remount the sys partition as writeable if executed in root mode
- Flagged apps are manually inspected and included in the known malware detection engine if they are genuinely malicious

Evaluation of Known Malware Detection

The following steps were taken to set up the evaluation of DroidRanger:

- Crawled Android market and collected 200K free apps:

	Official Market	Alternative Android Markets			
		M1	M2	M3	M4
Num. of Apps	153,002 (74.98%)	17,229 (8.44%)	14,943 (7.33%)	10,385 (5.09%)	8,481 (4.16%)
Total Apps		51,038 (25.02%)			
	204,040				

- Used 10 known malware families for behavioral footprints:

Malware	Reported time	Comments
Gemini	12/2010	Trojan with bot-like capabilities
ADRD	02/2011	Trojan with bot-like capabilities
Pjapps	02/2011	Trojan with bot-like capabilities
Bgserv	03/2011	Trojan with bot-like capabilites
DroidDream	03/2011	Root exploits with Exploid, Rageagainstthecage
zHash	03/2011	Root exploit with Exploid
BaseBridge	05/2011	Root exploit with Rageagainstthecage
DroidDreamLight	05/2011	Trojan that sends premium-rate SMS messages
jSMSHider	06/2011	Trojan that targets custom firmware devices

Evaluation of Known Malware Detection (Cont.)

I. Permission-based filtering

- Extracted permissions from each of the test apps
- Pruned apps successfully by comparing with malware permissions (one exception was DroidDreamLight which required use of an additional piece of meta info)

Malware	Permissions	Pruned App #
Gemini	INTERNET, SEND_SMS	7,620 (4.17%)
ADRD	INTERNET, ACC_NET_STATE, ...	10,379 (5.68%)
Pjapps	INTERNET, RECEIVE_SMS	4,637 (2.54%)
Bgserv	INTERNET, RECV_SMS, SND_SMS	2,880 (1.58%)
DroidDream	CHANGE_WIFI_STATE	4,096 (2.24%)
zHash	CHANGE_WIFI_STATE	4,096 (2.24%)
BaseBridge	NATIVE_CODE	8,272 (4.52%)
DroidDreamLight	INTERNET, RD_PHONE_STATE	71,095 (38.89%)
jSMSHider	INSTALL_PACKAGES	1,210 (0.66%)

- The signing key of some third party firmware is available, so jSHSHider can request INSTALL_PACKAGES permission since it is signed with the same key as the firmware (normally apps can't have this permission)

Evaluation of Known Malware Detection (Cont.)

II. Behavioral footprint analysis

- Total scan time 4.5 hours
- Malware detection results

Malware	Official Market	Alternative Android Markets				Total	Distinct
		M1	M2	M3	M4		
Gemini	0	26	26	2	10	64	37
ADRD	0	1	1	4	3	9	8
Pjapps	0	12	9	14	8	43	31
Bgserv	0	0	0	0	1	1	1
DroidDream	0	6	6	0	0	12	6
zHash	0	1	1	0	1	3	2
BaseBridge	0	2	2	0	2	6	4
DroidDreamLight	12	0	0	0	0	12	12
jSMSHider	0	3	3	0	6	12	9
Total	21	51	48	20	31	171	119

- More infections in non-Google markets
- Verified manually that all of the above instances are malicious

Evaluation of Known Malware Detection (Cont.)

Testing for false negatives:

- Got contagio dump files, i.e. 27 malware samples in our 10 families
- Deleted ones that were already in our training set, leaving 24 samples
- Accuracy rate of DroidRanger was $23/24 = 4.2\%$ false negative rate
- Compared with Lookout Security and Antivirus
  - v. 6.3 has 23.52% false negatives
  - v. 6.11 has 5.04% false negatives

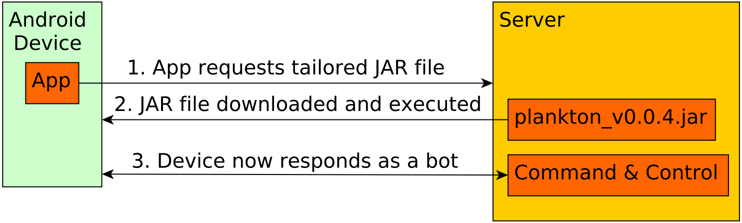
Evaluation of Unknown Malware Detection

Heuristic I. Dynamic code loading/execution

- Uncovered the Plankton malware
- Found in “Angry Birds Cheater” app
- Used the behavioral footprint to uncover another 10 similar instances in Google marketplace, and reported them to Google
- Google removed these 11 malicious apps on the same day they were reported

Evaluation of Unknown Malware Detection (Cont.)

The Plankton malware:


Android Device
App
1. App requests tailored JAR file
2. JAR file downloaded and executed
3. Device now responds as a bot
Server
plankton_v0.0.4.jar
Command & Control

- Attempts to load code plankton_v0.0.4.jar from remote site
- JAR contains bot-related functions that can be remotely invoked (e.g. get browser history, bookmarks, app log)

Evaluation of Unknown Malware Detection (Cont.)

Heuristic II. Non-standard placement of native code

- Found apps that try to remount system partition (this usually means app has gained root permissions)
- Found DroidKungFoo
- Contains encrypted Rageagainstthecage and Exploid
- App decrypts the root exploits and uses them to escalate privileges
- Installs apps, e.g. an indentical-looking Google Search app that acts as a bot client

Evaluation of Unknown Malware Detection (Cont.)

- Summary of results:

Malware	Official Market	Alternative Android Markets				Total	Distinct
		M1	M2	M3	M4		
Total (Known)	21	51	48	20	31	171	119
Total (Zero-day)	11	9	10	1	9	40	29
Total	32 (0.02%)	60 (0.35%)	58 (0.39%)	21 (0.20%)	40 (0.47%)	211	148

- Some observations:
  - Malware (e.g. DroidDream) can persist longer on non-Google markets
  - It has been shown that 4 out of the 10 examined malware families have a root exploit
  - Mobile malware software doesn't always detect malware

Future Work

- This study only looks at free apps, while 36.2% of all apps are paid – it would be useful to have information about paid apps
- The study only looks at five android markets – these techniques could be extended to other app markets, and even other platforms, such as iPhone and the Apple store
- DroidRanger uses only two heuristics for detecting zero-day malware – there are many other options which could be investigated

Conclusion

- Analysis is done using 200K+ free apps from marketplaces
- DroidRanger seeks to detect malicious apps in this set
- Two schemes for malware detection are implemented:
  - Permission-based behavioral footprinting
  - Heuristics-based filtering
- DroidRanger detected 211 malicious apps
- DroidRanger detected 2 zero-day malware apps (in both Google marketplace and others)
- This project highlights the need for better policing of offical and alternative marketplaces

Bibliography

- W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri.
  A study of android application security.
  In *USENIX security symposium*, 2011.

- A.P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner.
  A survey of mobile malware in the wild.
  In *ACM workshop on Security and privacy in smartphones and mobile devices*, pages 3–14. ACM, 2011.

- P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall.
  These aren't the droids you're looking for: retrofitting android to protect data from imperious applications.
  In *CCS*, pages 639–652. ACM, 2011.

- Y. Zhou, Z. Wang, W. Zhou, and X. Jiang.
  Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets.
  In *Network and Distributed System Security Symposium*, 2012.