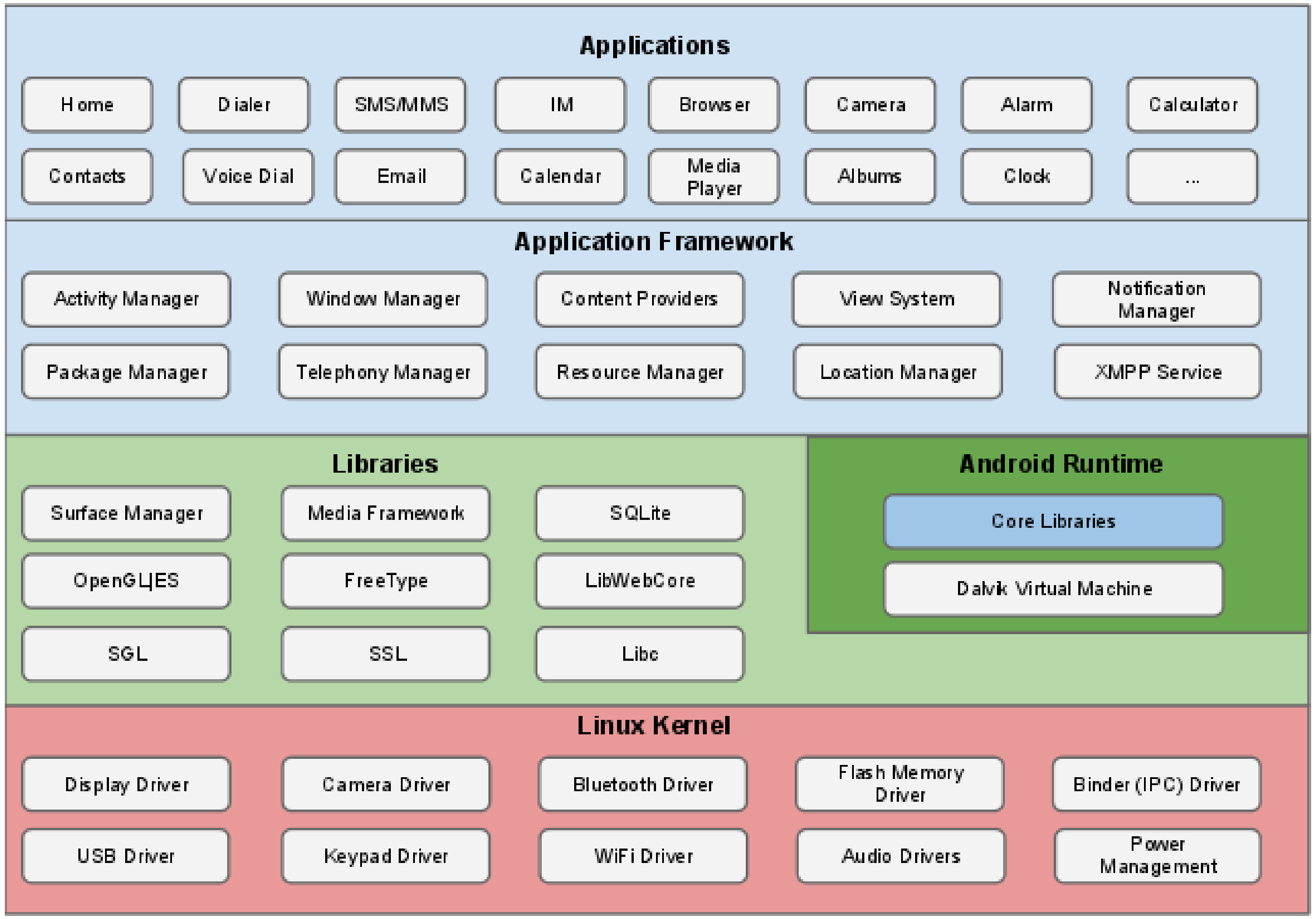
Android Permissions

Ben Holland

Android Software Stack



Source: https://source.android.com/devices/tech/security/

Application Sandbox

- Android applications run inside a mandatory sandbox
  - Private file storage
  - Restricted operations (permissions)
  - Isolated process/memory
- Secure interprocess communication (IPC)
- Application signing
  - All apps are signed by developer private key
  - Applications signed with same private key share permissions
  - Attack: find popular open source app and look in project history for accidently committed private keys

Android Manifest (AndroidManifest.xml)

- Names the application (Java) package, which acts as unique identifier
- Specifies top level components
  - Activities, Services, Broadcast Receivers, Content Providers
  - Component capabilities (priority, filters, exported, etc.)
- Specifies application permissions

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp" >
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    ...
</manifest>
```

Android Permissions

- Implemented using system user groups
  - Runtime security check
  - Permission restricted APIs without permissions granted throw runtime exceptions
  - How to enforce native code? i.e. Native code opens a socket to the Internet
- Permissions are categorized
  - Permission Groups
  - Protection Levels
- Permissions may overlap
  - ACCESS_FINE_LOCATION vs ACCESS_COARSE_LOCATION
- App can define custom permissions

Application Updates

- New permissions must be approved by user on update
- Old permissions do not have to be re-requested on updates!
  - Example: Facebook READ_SMS

Berkeley: Android permissions demystified

Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. 2011. Android permissions demystified. *In Proceedings of the 18th ACM conference on Computer and communications security (CCS '11).* ACM, New York, NY, USA, 627-638.

- Goal: Create mapping of APIs -> Android Permissions
- Dynamic Analysis of Android 2.2
    1. Randomly generate and call Android APIs in an app with no permissions
    2. If there is a security exception, generate and call same method in an app with the permission
    3. If API call does not throw a security exception add method to the set of permission restricted APIs for that permission

Berkeley: Android permissions demystified

- Limitations?
  - ~80% coverage of APIs
  - Difficult and elaborate experiment setup
  - Hard to repeat for new Android versions

- Advantages?
  - High confidence in results gathered for observed mappings

Berkeley: Android permissions demystified

- Discovered 6 incorrectly documented API permissions
  - Unknown whether the documentation or implementation is wrong
- Discovered non-existent permission in documentation
  - ACCESS_COARSE_UPDATES is not real, but some developers requested permission in apps anyway (makin’ copy-pasta)
- Some permissions are clear subsets of others
  - BLUETOOTH is subset of BLUETOOTH_ADMIN
- Some permissions are never checked
  - BRICK was never implemented in vanilla Android
  - Some manufacture specific flavors of Android modify permissions

Berkeley: Android permissions demystified

- Used mapping + static analysis to examine *principle of least privilege* in 940 apps

- Over-privileged Applications
    - Applications that request more permissions than they use
    - 35.8% of apps were over-privileged
- Under-privileged Applications
    - Applications that do not request enough permissions for their functionality

- Estimated 7% false positive rate
    - Java Reflection (61% of apps used reflection)
    - Native Code
    - Runtime.exec

Toronto: Analyzing the Android Permission Specification

- Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang and David Lie. PScout: Analyzing the Android Permission Specification. *In the Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS 2012).* October 2012.

- Goal: Generate API -> Permission mapping statically
- Static analysis of Android (2.2.3, 2.3.6, 3.2.2, 4.0.1, 4.1.1)
  1. Take Android OS source as input
  2. Generate program call graph
  3. Map explicit calls to checkPermission from API method
  4. Map permission flows through Intents (IPC)
  5. Map permission flows through Content Providers
  6. Perform feasibility checks

	Android Version			
	2.2	**2.3**	**3.2**	**4.0**
# LOC in Android framework	2.4M	2.5M	2.7M	3.4M
# of classes	8,845	9,430	12,015	14,383
# of methods (including inherited methods)	316,719	339,769	519,462	673,706
# of call graph edges	1,074,365	1,088,698	1,693,298	2,242,526
# of permission mappings for all APIs	17,218	17,586	22,901	29,208
# of permission mappings for documented APIs only	467	438	468	723
# of explicit permission checks	229	217	239	286
# of intent action strings requiring permissions	53	60	60	72
# of intents ops. w/ permissions	42	49	44	50
# of content provider URI strings requiring permissions	50	66	59	74
# of content provider ops. /w permissions	916	973	990	1417
KLOC/Permission checks	2.1	2.0	2.1	1.9
# of permissions	76	77	75	79
# of permissions required only by undocumented APIs	20	20	17	17
% of total permissions required only by undocumented APIs	26%	26%	23%	22%

Table 1: Summary of Android Framework statistics and permission mappings extracted by PScout. LOC data is generated using SLOCCount by David A. Wheeler.

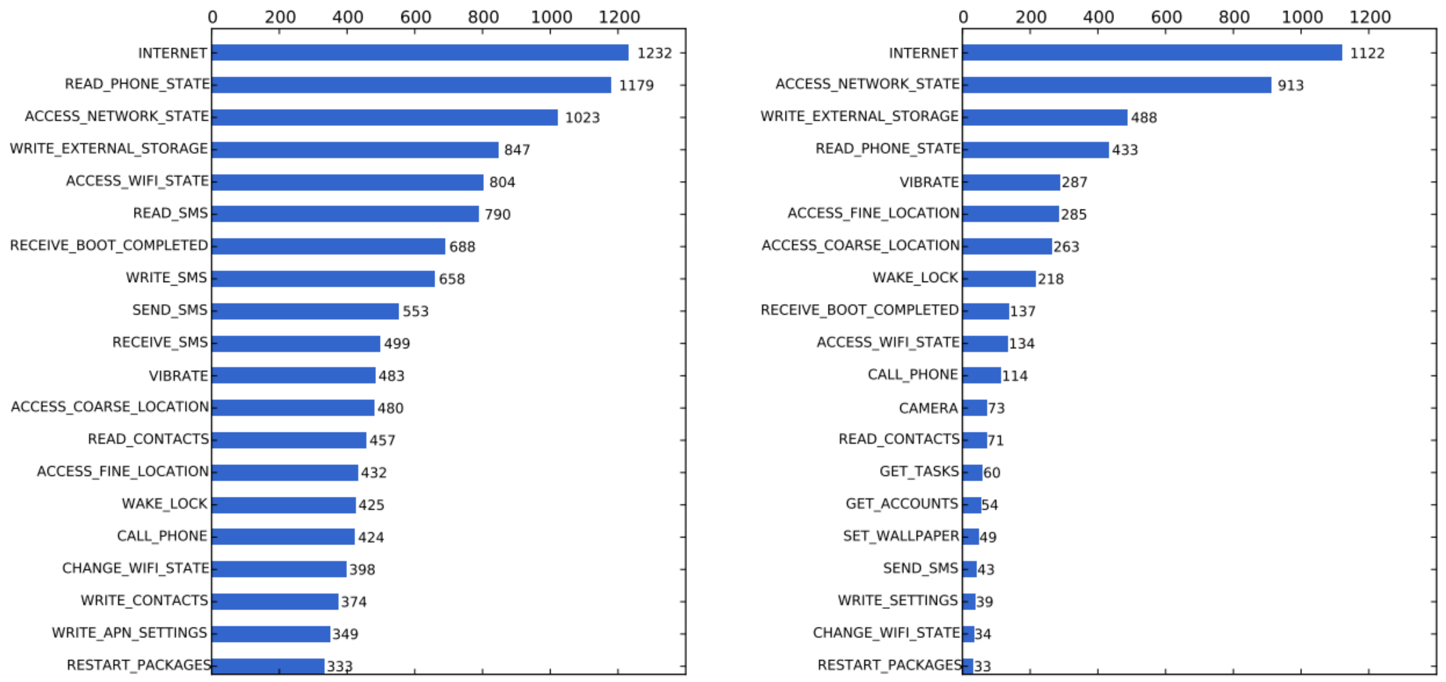
Toronto: Analyzing the Android Permission Specification

- Limitations?
    - Higher potential for false positives

- Advantages?
    - More complete mapping
    - Easy to repeat for new versions of Android
    - Includes undocumented (private) APIs
    - Includes undocumented (internal) permissions

Android Malware Overview

- Yajin Zhou, Xuxian Jiang. Dissecting Android Malware: Characterization and Evolution. http://www.csc.ncsu.edu/faculty/jiang/pubs/OAKLAND12.pdf. 2012.



(a) Top 20 Permissions Requested By 1260 Malware Samples (b) Top 20 Permissions Requested by 1260 Top Free (Benign) Apps on the Offical Android Market

Figure 5. The Comparison of Top 20 Requested Permissions by Malicious and Benign Apps

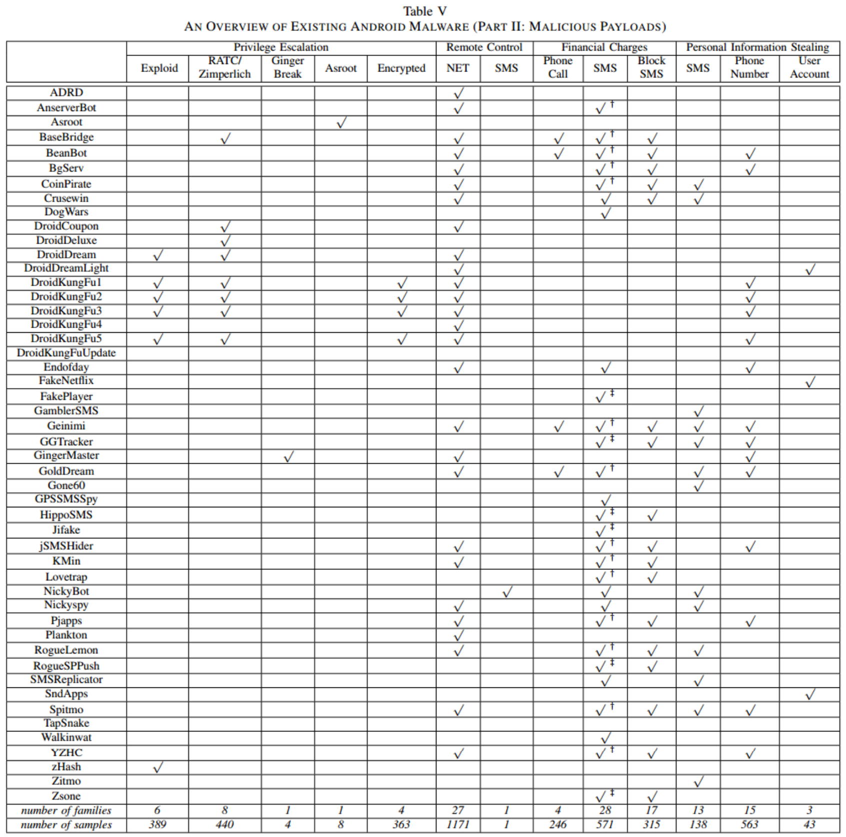
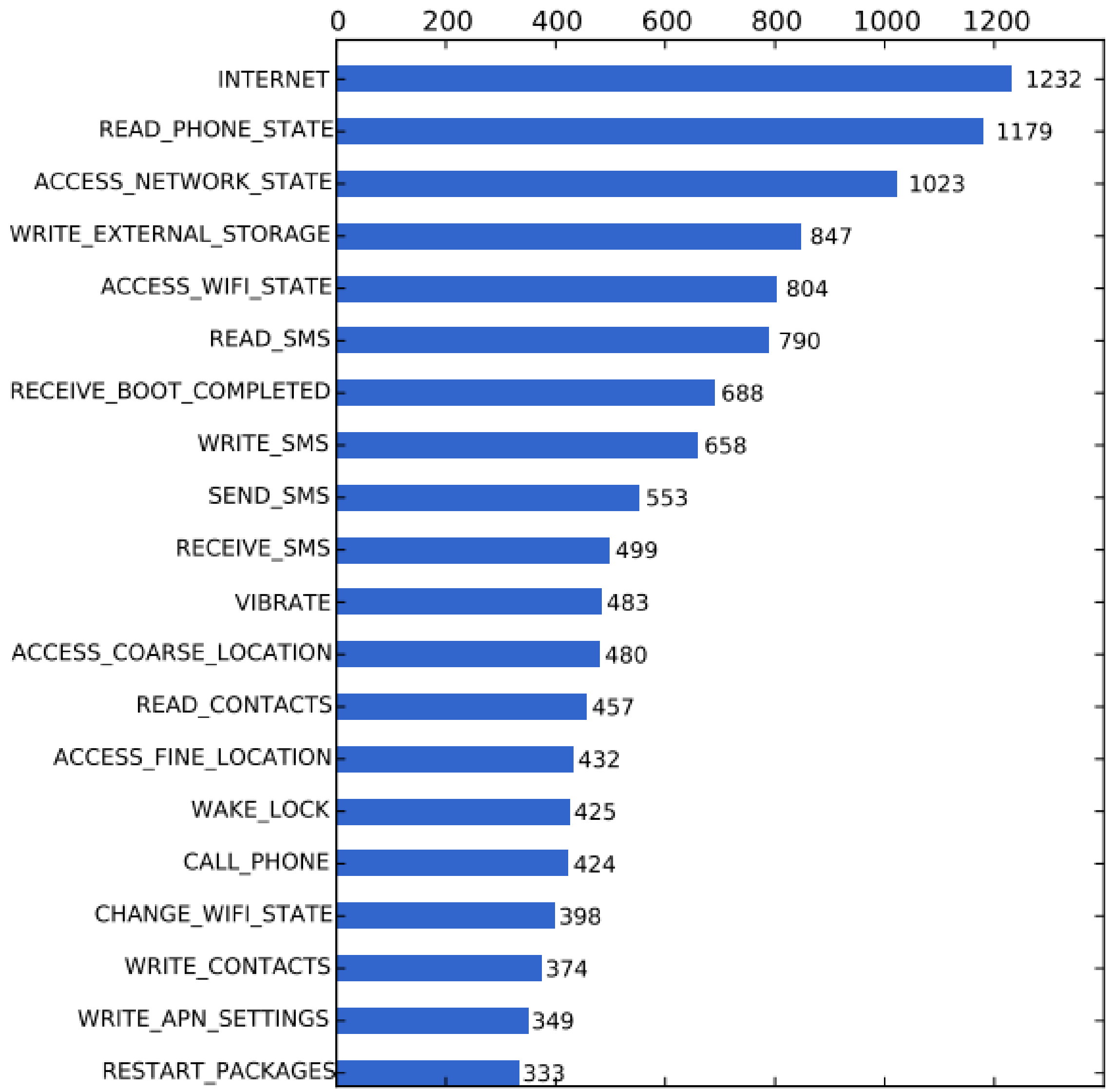
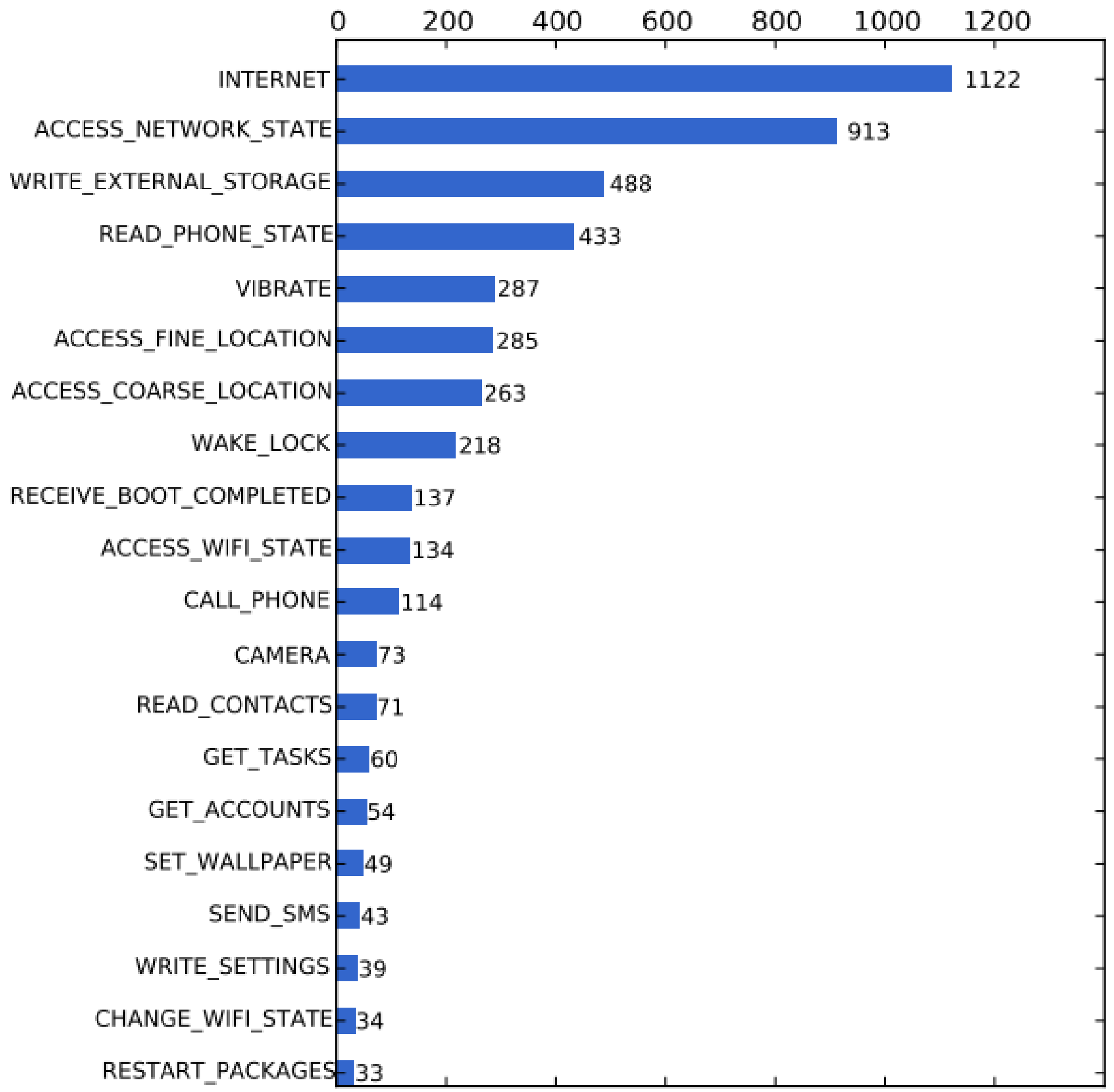


Table V
AN OVERVIEW OF EXISTING ANDROID MALWARE (PART II: MALICIOUS PAYLOADS)

	Privilege Escalation					Remote Control		Financial Charges			Personal Information Stealing		
	Exploid	RATC/ Zimperlich	Ginger Break	Asroot	Encrypted	NET	SMS	Phone Call	SMS	Block SMS	SMS	Phone Number	User Account
ADRD						√							
AnserverBot						√			√ †				
Asroot				√									
BaseBridge		√				√		√	√ †	√			
BeanBot						√		√	√ †	√		√	
BgServ						√			√ †	√		√	
CoinPirate						√			√ †	√	√		
Crusewin						√			√	√	√		
DogWars									√				
DroidCoupon		√				√							
DroidDeluxe		√											
DroidDream	√	√				√							
DroidDreamLight						√							√
DroidKungFu1	√	√			√	√						√	
DroidKungFu2	√	√			√	√						√	
DroidKungFu3	√	√			√	√						√	
DroidKungFu4						√							
DroidKungFu5	√	√			√	√						√	
DroidKungFuUpdate													
Endofday						√			√			√	
FakeNetflix													√
FakePlayer									√ ‡				
GamblerSMS											√		
Geinimi						√		√	√ †	√	√	√	
GGTracker									√ ‡	√	√	√	
GingerMaster			√			√						√	
GoldDream						√		√	√ †		√	√	
Gone60											√		
GPSSMSSpy									√				
HippoSMS									√ ‡	√			
Jifake									√ ‡				
jSMSHider						√			√ †	√		√	
KMin						√			√ †	√			
Lovetrap									√ †	√			
NickyBot							√		√		√		
Nickyspy						√			√		√		
Pjapps						√			√ †	√		√	
Plankton						√							
RogueLemon						√			√ †	√	√		
RogueSPPush									√ ‡	√			
SMSReplicator									√		√		
SndApps													√
Spitmo						√			√ †	√	√	√	
TapSnake													
Walkinwat									√				
YZHC						√			√ †	√		√	
zHash	√												
Zitmo											√		
Zsone									√ ‡	√			
number of families	*6*	*8*	*1*	*1*	*4*	*27*	*1*	*4*	*28*	*17*	*13*	*15*	*3*
number of samples	*389*	*440*	*4*	*8*	*363*	*1171*	*1*	*246*	*571*	*315*	*138*	*563*	*43*



(a) Top 20 Permissions Requested By 1260 Malware Samples



(b) Top 20 Permissions Requested by 1260 Top Free (Benign) Apps on the Offical Android Market

Figure 5. The Comparison of Top 20 Requested Permissions by Malicious and Benign Apps

Table V

AN OVERVIEW OF EXISTING ANDROID MALWARE (PART II: MALICIOUS PAYLOADS)

	Privilege Escalation					Remote Control		Financial Charges			Personal Information Stealing		
	Exploit	RATC/ Zimperlich	Ginger Break	Asroot	Encrypted	NET	SMS	Phone Call	SMS	Block SMS	SMS	Phone Number	User Account
ADRD						√							
AnserverBot						√			√†				
Asroot				√									
BaseBridge		√				√		√	√†	√			
BeanBot						√		√	√†	√		√	
BgServ						√			√†	√		√	
CoinPirate						√			√†	√	√		
Crusewin						√			√	√	√		
DogWars									√				
DroidCoupon		√				√							
DroidDeluxe		√											
DroidDream	√	√				√							
DroidDreamLight						√							√
DroidKungFu1	√	√			√	√						√	
DroidKungFu2	√	√			√	√						√	
DroidKungFu3	√	√			√	√						√	
DroidKungFu4						√							
DroidKungFu5	√	√			√	√						√	
DroidKungFuUpdate													
Endofday						√			√			√	
FakeNetflix													√
FakePlayer									√‡				
GamblerSMS											√		
Geinimi						√		√	√†	√	√	√	
GGTracker									√‡	√	√	√	
GingerMaster			√			√						√	
GoldDream						√		√	√†		√	√	
Gone60											√		
GPSSMSSpy									√				
HippoSMS									√‡	√			
Jifake									√‡				
jSMSHider						√			√†	√		√	
KMin						√			√†	√			
Lovetrap									√†	√			
NickyBot							√		√		√		
Nickyspy						√			√		√		
Pjapps						√			√†	√		√	
Plankton						√							
RogueLemon						√			√†	√	√		
RogueSPPush									√‡	√			
SMSReplicator									√		√		
SndApps													√
Spitmo						√			√†	√	√	√	
TapSnake													
Walkinwat									√				
YZHC						√			√†	√		√	
zHash	√												
Zitmo											√		
Zsone									√‡	√			
number of families	*6*	*8*	*1*	*1*	*4*	*27*	*1*	*4*	*28*	*17*	*13*	*15*	*3*
number of samples	*389*	*440*	*4*	*8*	*363*	*1171*	*1*	*246*	*571*	*315*	*138*	*563*	*43*

Permission Abuse

- How do we know if an app is abusing permissions?
- https://developer.android.com/reference/android/Manifest.permission.html
    - 146 (documented) permissions as of Android API 19

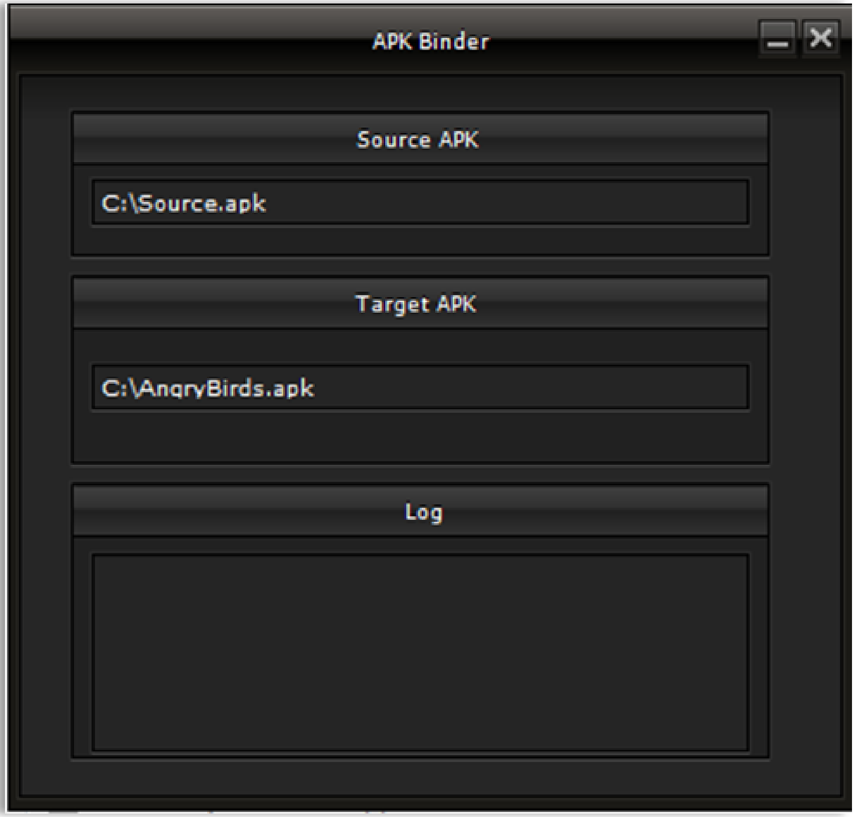
Behavior	**App Purpose**	**Classification**
Send location to Internet	Phone locator	Benign
Send location to Internet	Podcast player	Malicious
Selectively block SMS messages	Ad blocker	Benign
Selectively block SMS messages	Navigation	Malicious

Dendroid Source Leak

- Android malware source with C&C server
- https://github.com/lululeta2014/DendroidSource ($300)

•Media volume up/down
•Ringer volume up/down
•Screen On
•Record Calls
•Block SMS
•Record Audio
•Take Video
•Take Photo
•Send Text
•Send Contacts
•Get user accounts

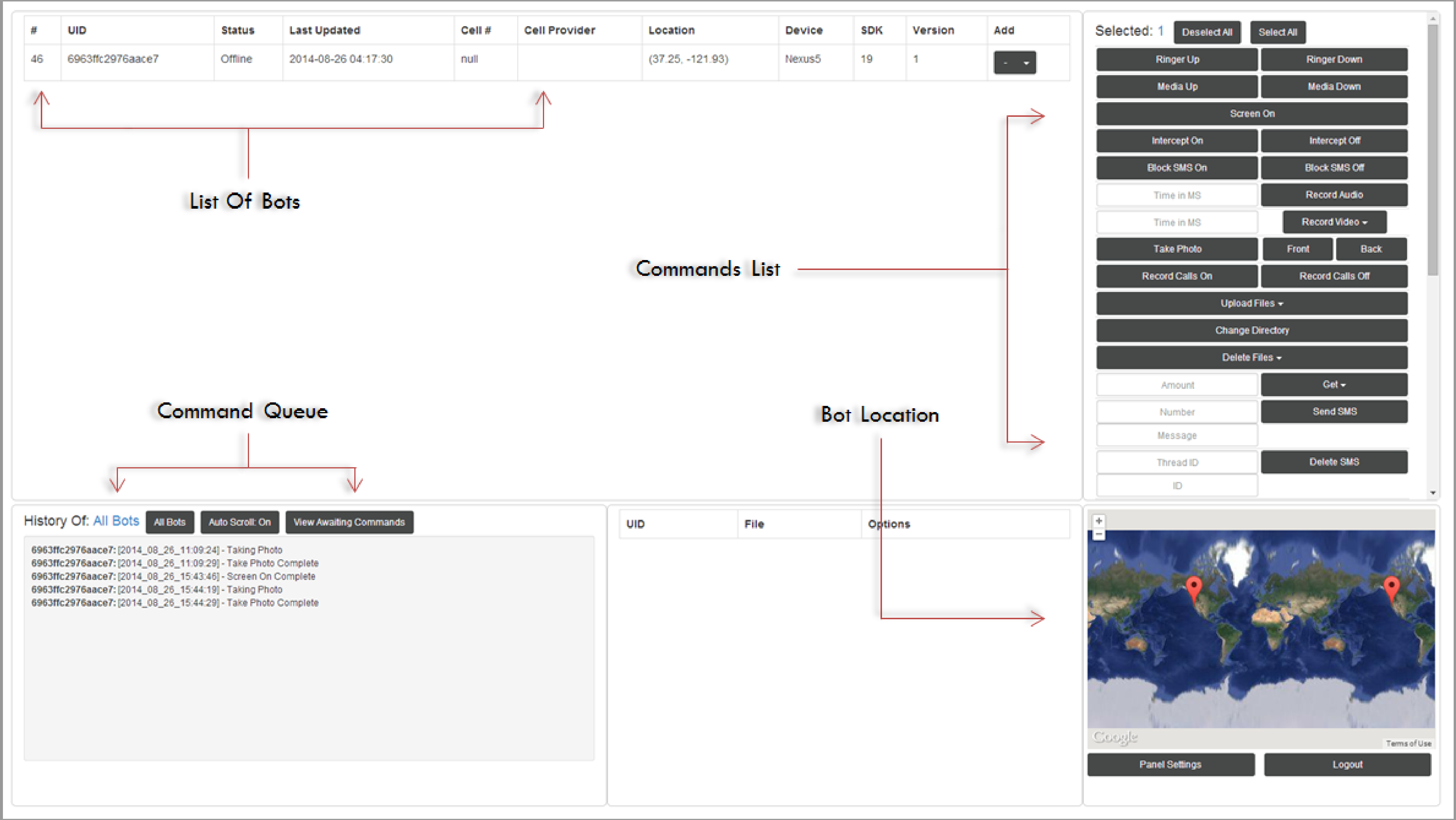
•Call Number
•Delete Call Logs
•Open Webpage
•Update the app
•Delete Files (audio, video, pictures, calls)
•Get Browser History
•Get Browser Bookmarks
•Get Call History
•Open Dialog Box
•Get Inbox SMS
•HTTP flood



C:\Source.apk

C:\AngryBirds.apk

Source: https://www.mysonicwall.com/sonicalert/searchresults.aspx?ev=article&id=718



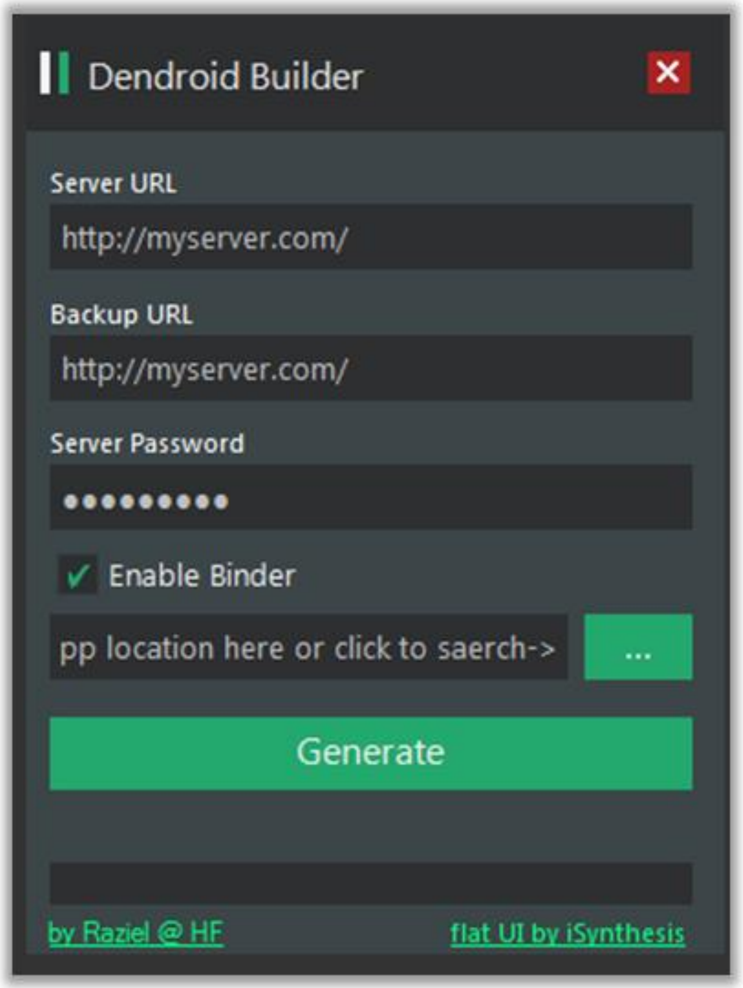
#	UID	Status	Last Updated	Cell #	Cell Provider	Location	Device	SDK	Version	Add
46	6963ffc2976aace7	Offline	2014-08-26 04:17:30	null		(37.25, -121.93)	Nexus5	19	1	-

History Of: All Bots | All Bots | Auto Scroll: On | View Awaiting Commands

```
6963ffc2976aace7: [2014_08_26_11:09:24] - Taking Photo
6963ffc2976aace7: [2014_08_26_11:09:29] - Take Photo Complete
6963ffc2976aace7: [2014_08_26_15:43:46] - Screen On Complete
6963ffc2976aace7: [2014_08_26_15:44:19] - Taking Photo
6963ffc2976aace7: [2014_08_26_15:44:29] - Take Photo Complete
```

UID	File	Options

Dendroid Source Leak (improvements)


Dendroid Builder

Server URL

http://myserver.com/

Backup URL

http://myserver.com/

Server Password

●●●●●●●●●

✓ Enable Binder

pp location here or click to saerch->

Source: https://www.mysonicwall.com/sonicalert/searchresults.aspx?ev=article&id=718

Example: Google Bouncer

- When you submit an app to the Google App store, Google runs the app in an emulator for 5 minutes and looks for malicious activity
  - How would you bypass Google Bouncer dynamic analysis?
  - What about any prior static analysis checks?

- Dissecting Android's Bouncer
  - Submitted reverse TCP shells in an Android apps to probe Google Bouncer
  - https://www.youtube.com/watch?v=ZEIED2ZLEbQ
  - https://www.youtube.com/watch?v=WBQowLKwsyE