

A Critical Approach for Intruder Detection in Mobile Devices

⁽¹⁾ **R.Surendiran,**
Research Scholar,
Dept of MCA, Computer Center,
Madurai Kamaraj University, Madurai.

⁽²⁾ **Dr.K.Alagarsamy,**
Associate Professor
Dept of MCA, Computer Center,
Madurai Kamaraj University, Madurai.

Abstract :

The paper studies and proposed an new effective methodology for the security threats involved in using modern smart phone for accessing Internet related services and various sources. Different types of materialised mobile malware are discussed in our previous works in order to classify them for further analysis. This paper discuss growing interest of malicious codes and money making apps in mobile phone users. Security Threats of mobile phones are explored in order to compare the involved security risks in mobile environment. Analysing various security problems of mobile environment are discussed for predicting the type of attacks that may cause a substantial rise of the risk level in near future. This paper is closed with a effective solution for mobile virus and intruder attacks from the third party application and resources.

Keywords: Mobile Devices, Security threats , Detection Mechanism.

I. Introduction:

In recent years computer [1] and technology has developed from centralized computing environment which supporting static applications into client server architecture that allow complex forms of distributed applications. In the Entire evolution certain forms of code have existed. Olden remote job program entry terminals used to submit programs to a centralized server and the latest versions like Java applets downloaded from web servers into web browsers. A new advanced architecture is now under way that goes one step further, allowing complete mobility of cooperating applications among supporting platforms to form a large scale, loosely coupled distributed system.

The mobile is the most important social classes of teenagers, who favor SMS ,facebook,chats,messengers ,etc. Mobile devices used as efficient and trendy form of personal communication. During the year of 2007 to 2010 , More than 95% of teenagers sends sms and messenger then they talk on their mobile phones.

In this research approach provides an efficient security mechanism for smart phones with usage of kernal environments because most of the core processing happening inside the kernal part . If we implement the security mechanism in kernal level we can reduce most of the vulnerability attacks.

This paper contain following parts. At first we have given small introduction. In the second part we have discussed about proposed approach ,overall architecture ,design of algorithm and basic working methodology. In the

third section we have evaluated the methodology and given the statistical analysis. In the fourth part discussing the conclusion and finally go for the references.

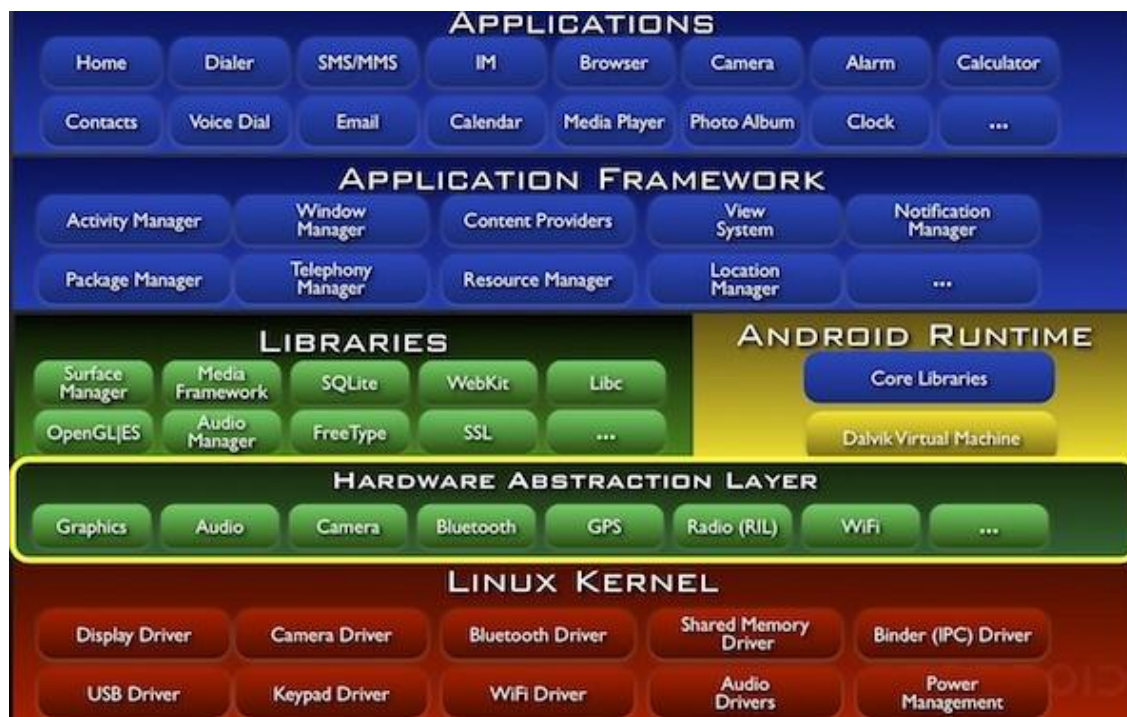


Figure 1: Overall Android Platform

II. Proposed Approach:

This research focusing security mechanism in two different category one is well known thread and another one is unknown thread.

While designing efficient security algorithm we have to concentrate on three important security principles namely

- **Accuracy :**
It is a very important requirement for detect the fraudulent application or apps from the android market.
- **Scalability :**
It is also very important requirement for efficient algorithm design. According to this scenario how far or how much application will extent for this algorithm .
- **Efficiency :**
Efficiency always shows the performance of algorithm or design. According to this scenario how far it will detect the malicious application.

In General every application contains lot of features and functionalities . There are numerous applications available in the market for different purpose and different usages like official, entertainment, games,etc. if we take all the features to determine the malicious app, It takes much time to pre process,scan and predict the malicious one. So we are taking only the important features in the application.

Following diagram shows general architecture of our drivers and behaviour.

A simplified architecture depicting the role of a device driver

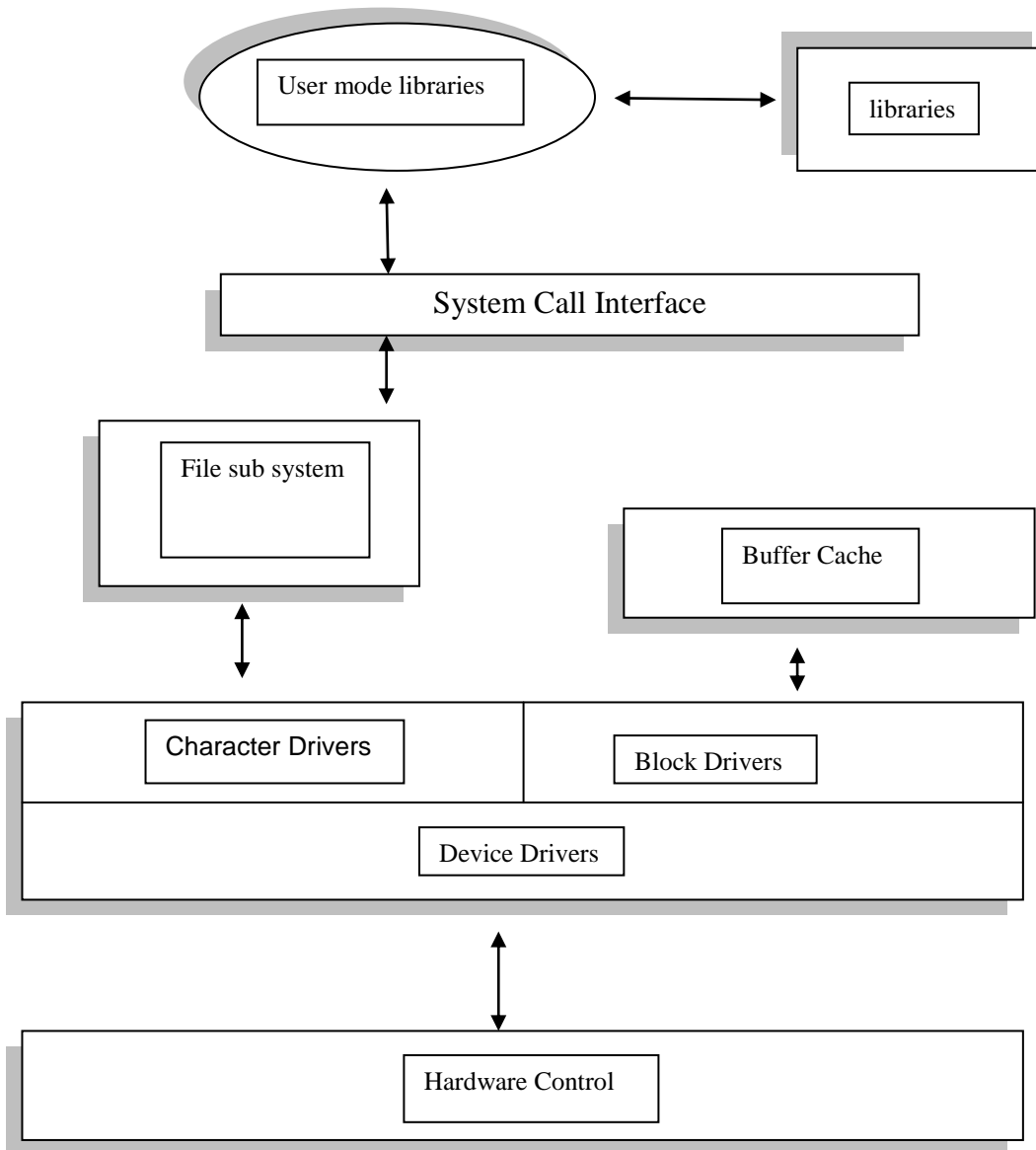


Figure 2: General architecture of drivers and behaviour.

Overall Design Algorithm:

- Collect the application from the Existing Markets from various vendors
- Save into our local drives or specific location
- Apply our Known threats methodology
- Apply our unknown threats methodology
- Predict the results

Collection of application has been stored into the local repository from the various vendors like android, apple, symbian, windows ,etc. This process is fully automated otherwise it will take much time to collect the application. Since every application softwares and supporting files are stored in their official server due to that we have to store in our local repository like hard disk or any special embedded disk. Now we are ready to process the application first we will see the known malware detection and next we will discuss the unknown threats.

Finding Known Threats:

In this methodology there will be a two step one is elimination round for reduce the unnecessary application from the large number of application in the local repository which are collected from the various vendors and markets. Elimination round is very important because to reduce the time of scanning from the large scanning.

In every android application contain a manifest file which persists some basic information , permission, metadata for authentication purpose unlike the malicious application. Malware application doesn't contain these kind of information which are more useful for remove the unrelated applications from the collection and also it will more useful for time consuming and increase the efficiency of algorithm.

For example we use short message service (SMS) because everyone knows the SMS. For understanding purpose we take SMS for illustration purpose. While sending and receiving the SMS there will be a some common communication commands will get executed in kernel level like SEND_SMS,RECEIVE_SMS. Let us take one malware persist in our application related to SMS. Malware can send and receive the N number of SMS in the background process without knowledge of victim and also it will hide the financial related information and messages. Probably 95% application get removed from this stage. In particular we have to concentrate few things in the elimination round , as per selection criteria we just choose only important permission ,system call, functionality. In our scenario we can select these commands like SEND_SMS,RECEIVE_SMS,INTERNET,BOOKMARKS. If we predict wrong commands it leads negative results.

In the second stage we go for some depth analysis in the aspect of semantic approach. This can be done in the two ways one is automated and another one is traditional approach by manual because if the first stage we have eliminated many number of application due to that we can allot much time to second stage. There are some important things has to be consider namely manifest file, byte code, structure of android.

Manifest file contain the semantic information related to the application which will be very useful for determine either applications are legitimate or fake one. Byte code contain well semantic information , which will be highly useful for understand the application since byte code understandability will come from the expert in the android field. Finally android app created based on some tree structure and it will be in compressed format. If we extract the archives we can get the original structure of application from that we can define what kind of methods system call's,packages,etc has been used in the application

Finding Unknown Threats:

In our first method we have focused on known threats . here we are concentrate on unknown threats. There are two methodologies will be there we will see one by one.

First one based on native code storage and appearance. Every android app contained classes.dex file which contain dalvik byte code for execution. Dalvik virtual machine provides dex class loader which is used to load external files by the application like java files and execution files. If app included dynamically loaded file from the server , It leads to the technical issue and security threats. If any application used DexClassLoader file more than 30% , we should consider app itself suspicious.

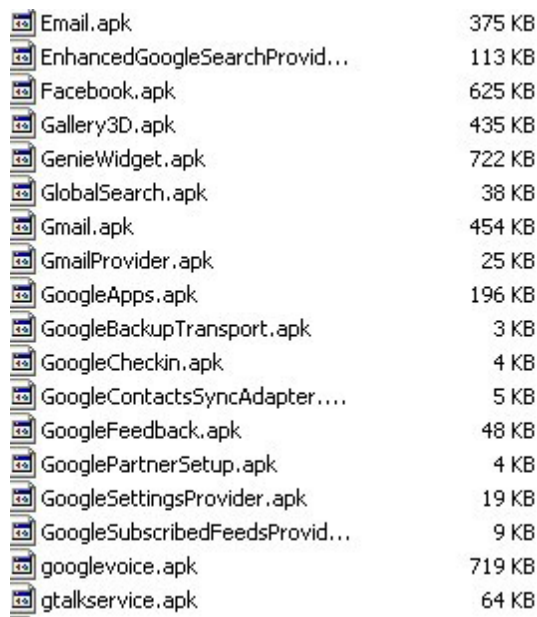
Most of the android application designed in java. Even though some of the application using native code for performance measurement. Each application will run in the separate user id and unix system calls can have ability to access the native code directly. If any malicious app call the native code, it can drop the viruses in the root itself. There are some default location in android to place the native code based on that concept we can predict the viruses.

In the first step we were identified some unwanted codings running in the application and next step monitoring the dynamic execution of unwanted codings. From the application code it can inherit some other new codes. In our automated monitor scan maintain all the system calls and arguments. Arguments will list out all the semantic information. With the help of native code dynamic monitor collects all the system call probably system call will maintain by linux. We couldn't inspect all the logs instead we can trace only important system call. From that collection we can find out suspicious runtime behaviours.

III. Result Analysis and Discussion:

This research approach implemented in android 2.2 simulator . we have taken nearly 50000 application from the various market especially we have focused android market for our implementation purpose. Collecting application itself tedious process which makes unsensible sometimes.

Here the snap shot of multiple application running in single android os



Email.apk	375 KB
EnhancedGoogleSearchProvid...	113 KB
Facebook.apk	625 KB
Gallery3D.apk	435 KB
GenieWidget.apk	722 KB
GlobalSearch.apk	38 KB
Gmail.apk	454 KB
GmailProvider.apk	25 KB
GoogleApps.apk	196 KB
GoogleBackupTransport.apk	3 KB
GoogleCheckin.apk	4 KB
GoogleContactsSyncAdapter....	5 KB
GoogleFeedback.apk	48 KB
GooglePartnerSetup.apk	4 KB
GoogleSettingsProvider.apk	19 KB
GoogleSubscribedFeedsProvid...	9 KB
googlevoice.apk	719 KB
gtalkservice.apk	64 KB

Figure 3: Multiple Process Running Mode

Applications are collected from the various markets like google play(Official site), mmoovv, eoemarket, gfan and alcateclub .

Market Name	google play	mmoovv	eoemarket	gfan	alcateclub .
count of App	24725	6320	6227	6037	6709

Table 1: Collection of Application

Around 50,000 application were collected from the five different markets.

We have taken five known samples for known malware detection algorithm

Malicious App	Year	Hash Code	Functionality
Bgserv	2012	ea97576befac2b142144ce30c2326ed61d696b87e665498b878bf92ce25440db	Trojan with bot-like capabilities
DroidDream	2012	63f26345ba76ef5e033ef6e5cceed30d763a1ab4e4a21373a24a921227a6f7a4	Root exploits with Exploit, Rageagainstthecage
DroidDreamLight	2012	3ae28cbf5a92e8e7a06db4b9ab0a55ab5e4fd0a680ffc8d84d16b8c0e5404140	Trojan with information stealing capabilities
jSMShider	2012	a3c0aacb35c86b4468e85bf9e226955389b416fb0f505d661716b8da02f92a2	Trojan that targets custom firmware devices
Zsone	2012	d204007a13994cb1ad68d8d2b89cc9a8a673481ff15028e05f6fe778a773e0f9	Trojan that sends premium-rate SMS messages

Table 2: Existing Threats and functionality

Let’s we discuss about the known malware detection performance. According to our study we have find the following facts

	SEND_SMS	RECEIVED_SMS	BOTH
Apps	1513	2230	828
Percentage	3.03%	4.46%	1.65%

Table 3: SMS command Utilization

The above table shows SEND_SMS command eliminate 97.07% apps need not to be consider for further evaluation and RECEIVED_SMS command eliminate 96.54% application. Combining both it is eliminate 98.35% . only remaining application has to forward the second step

In the second step SEND_SMS command will follow some specific series but app will encode in some other series for example SEND_SMS series 1034567854, 1034578651, 1034554673, 1034533834, etc.. But malicious application will follow different series it will differ application to application.

Next thing is unknown malware detection performance evaluation. First we will see the native code not present in the proper location

	Application with Native code	Native code in Asset Directory	Native Code in RES directory
Apps	2270	112	62
Percentage	4.54%	0.22%	0.12%

Table 4: Native Code Usage

Essential commands used in the above malwares

Malicious App	Commands	Count
Bgserv	INTERNET, RECEIVE SMS, SEND SMS	553
DroidDream	CHANGE WIFI STATE	1012
DroidDreamLight	INTERNET, READ PHONE STATE	17517
jSMShider	INSTALL PACKAGES	311
Zsone	RECEIVE SMS, SEND SMS	1016

Table 5: Malwares & Permission

The Number of infected application detected by our algorithm

Malware	google play	mooovv	eoemarket	gfan	alcateclub	Total	Peculiarity
Bgserv	0	0	0	0	2	2	2
DroidDream	0	5	3	6	0	14	12
DroidDreamLight	18	0	0	0	0	18	14
jSMShider	0	3	0	4	4	11	10
Zsone	20	15	16	21	32	104	78
Total	38	23	19	31	38	149	116

Table 6: Infected application from known Threats

Unknown Malware detected:

Malware	google play	mooovv	eoemarket	gfan	alcateclub	Total	Peculiarity
Plankton	14	0	0	0	1	15	13
DroidKungfu	1	8	7	6	8	29	22
Total	15	8	7	6	9	44	35

Table 7: Infected application from Unknown Threats

Total malware Detected:

Malware	google play	mooovv	eoemarket	gfan	alcateclub	Total	Peculiarity
Known	38	23	19	31	38	149	116
Unknown	15	8	7	6	9	44	35
Total	53	31	26	37	47	193	151

Table 8: Total Infected application

IV. Conclusion

In this article we have discussed various methodology to detect the malicious application in the android market and some other third party market also. This research clearly discuss the malicious application behaviour and execution process and also it was classified into two major group namely known threats and unknown threats. We have discuss with various examples and given the exact out come of this research. In the result and analysis section describes the working procedure ,evaluated data, out come of the research and prediction. From that we can determined the effectiveness of the algorithm and procedure. It's our belief , this research will be more useful for research scholar and research community for the next level of the advancement.

V. References

- [1] NIST Special Publication 800-19 – Mobile Agent Security Wayne Jansen, Tom Karygiannis National Institute of Standards and Technology Computer Security Division Gaithersburg, MD 20899.
- [2] W. Zhou, Y. Zhou, X. Jiang, and P. Ning. DroidMOSS: Detecting Repackaged Smartphone Applications in Third-Party AndroidMarketplaces. In Proceedings of the 2nd ACMConference on Data and Application Security and Privacy, CODASPY' 12, 2012.
- [3] Amazon Appstore. <http://www.amazon.com/mobile-apps/b?ie=UTF8&node=2350149011>.
- [4] Android.Bgserv Found on Fake Google Security Patch.<http://www.symantec.com/connect/blogs/androidbgservfound-fake-google-security-patch>.
- [5] W. Enck, M. Ongtang, and P. McDaniel. On Lightweight Mobile Phone Application Certification. In Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS '09, 2009.
- [6] Android.Pjapps Technical Details. http://www.symantec.com/security_response/writeup.jsp?docid=2011-022303-3344-99&tabid=2.
- [7] Bo Li and Eul Gyu Im: Smartphone, promising battlefield for hackers, Journal of Security Engineering , vol: 8 no: 1, 2011, pages 89-110
- [8] Exploit Root Exploit. <http://c-skills.blogspot.com/2010/07/exploit-works-on-droid-x.html>.
- [9] Mmoovv. <http://android.mmoovv.com/web/index.html>.
- [10] 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 Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS '11, 2011.
- [11] Rageagainstthecage Root Exploit. <http://c-skills.blogspot.com/2010/08/droid2.html>.
- [12] A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy. A Crawler-based Study of Spyware on the Web. In Proceedings of the 13th Annual Symposium on Network and Distributed System Security, NDSS '06, 2006.
- [13] Security Alert: DroidDream Malware Found in Official Android Market. <http://blog.mylookout.com/2011/03/securityalert-malware-found-in-official-android-marketdroiddream/>.
- [14] J. Andrus, C. Dall, A. Van't Hof, O. Laadan, and J. Nieh.Cells: A Virtual Mobile Smartphone Architecture. In Proceedings of the 23rd ACM Symposium on Operating Systems Principles, SOSP '11, 2011.
- [15] Security Alert: HongTouTou, New Android Trojan, Found in China. <http://blog.mylookout.com/2011/02/security-alert-hongtoutou-new-android-trojan-found-in-china/>.
- [16] Security Alert: Zsone Trojan found in Android Market. <http://blog.mylookout.com/2011/05/security-alert-zsonetrojan-found-in-android-market/>.
- [17] Zimperlich sources. <http://c-skills.blogspot.com/2011/02/zimperlich-sources.html>.
- [18] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner. Analyzing Inter-Application Communication in Android. In Proceedings of the 9th Annual Symposium on Network and Distributed System Security, MobiSys 2011, 2011.
- [19] A. R. Beresford, A. Rice, N. Skehin, and R. Sohan. Mock-Droid: Trading Privacy for Application Functionality on Smartphones. In Proceedings of the 12th International Workshop on Mobile Computing System and Applications, HotMobile '11, 2011.
- [20] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani. Crowdroid:Behavior-Based Malware Detection System for Android. In Proceedings of the 1st Workshop on Securityand Privacy in Smartphones and Mobile Devices, CCSSPSM'11, 2011.
- [21] B. Dixon, Y. Jiang, A. Jaiantilal, and S. Mishra. Location Based Power Analysis to Detect Malicious Code in Smartphones. In Proceedings of the 1st Workshop on Security and Privacy in Smartphones and Mobile Devices, CCSSPSM' 11, 2011.

- [22] Alcatelclub. <http://www.alcatelclub.com/>.
- [23] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. PiOS: Detecting Privacy Leaks in iOS Applications. In Proceedings of the 18th Annual Symposium on Network and Distributed System Security, NDSS '11, 2011.
- [24] Security Alert: Malware Found Targeting Custom ROMs (jSMShider). <http://blog.mylookout.com/2011/06/securityalert-malware-found-targeting-custom-roms-jsmshider/>.
- [25] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An Information-Flow Tracking System for Realtime PrivacyMonitoring on Smartphones. In Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation, USENIXOSDI '10, 2010.
- [26] A. Bose, X. Hu, K. G. Shin, and T. Park. Behavioral Detection of Malware on Mobile Handsets. In Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services, MobiSys '08, 2008.
- [27] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A Study of Android Application Security. In Proceedings of the 20th USENIX Security Symposium, USENIX Security '11, 2011.
- [28] Security Alert: zHash, A Binary that can Root Android Phones, Found in Chinese App Markets and Android Market. <http://blog.mylookout.com/2011/03/security-alertzhash-a-binary-that-can-root-android-phones-found-inchinese-app-markets-and-android-market/>.
- [29] A. Fuchs, A. Chaudhuri, and J. Foster. SCanDroid: Automated Security Certification of Android Applications. <http://www.cs.umd.edu/~avik/projects/scandroidascaa>.
- [30] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android Permissions Demystied. In Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS '11, 2011.
- [31] L. Liu, G. Yan, X. Zhang, and S. Chen. VirusMeter: Preventing Your Cellphone from Spies. In Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, RAID'09, 2009.
- [32] A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin. Permission Re-Delegation: Attacks and Defenses. In Proceedings of the 20th USENIX Security Symposium, USENIX Security '11, 2011.
- [33] M. Grace, Y. Zhou, Z. Wang, and X. Jiang. Systematic Detection of Capability Leaks in Stock Android Smartphones. In Proceedings of the 19th Annual Symposium on Network and Distributed System Security, NDSS '12, 2012.
- [34] M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu, and D. S. Wallach. QUIRE: Lightweight Provenance for Smart Phone Operating Systems. In Proceedings of the 20th USENIX Security Symposium, USENIX Security '11, 2011.
- [35] H. Kim, J. Smith, and K. G. Shin. Detecting Energy-Greedy Anomalies and Mobile Malware Variants. In Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services, MobiSys '08, 2008.
- [36] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel. Semantically Rich Application-Centric Security in Android. In Proceedings of the 2009 Annual Computer Security Applications Conference, ACSAC '09, 2009.
- [37] Android. Basebridge Technical Details. http://www.symantec.com/security_response/writeup.jsp?docid=2011-060915-4938-99&tabid=2.
- [38] L. Xie, X. Zhang, J.-P. Seifert, and S. Zhu. pBMDs: A Behavior-based Malware Detection System for Cellphone Devices. In Proceedings of the 3rd ACM conference on Wireless Network Security, WiSec '10, 2010.
- [39] Security Alert: Geinimi, Sophisticated New Android Trojan Found in Wild. <http://blog.mylookout.com/2010/12/geinimitrojan/>.
- [40] AdTOUCH. <http://www.adtouchnetwork.com/adtouch/sdk/SDK.html>.
- [41] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh. Taming Information-Stealing Smartphone Applications (on Android). In Proceeding of the 4th International Conference on Trust and Trustworthy Computing, TRUST '11, 2011.
- [42] Update: Security Alert: DroidDreamLight, New Malware from the Developers of DroidDream. <http://blog.mylookout.com/2011/05/security-alert-droiddreamlight-new-malwarefrom-the-developers-of-droiddream/>.
- [43] S. Cooperation, "Symantec Internet Security Threat Report Volume XVI," *Whitepaper*, vol. 16, Apr 2011.
- [44] Kaspersky Lab, "Popular Porn Sites Distribute a New Trojan Targeting Android Smartphones," 2010. [Online]. Available: <http://www.kaspersky.com/news?id=207576175>
- [45] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "Survey of Mobile Malware in the Wild," 2011. [Online]. Available: <http://www.eecs.berkeley.edu/~afelt/malware.html>
- [46] McAfee Labs, "2011 Threats Predictions," 2010. [Online]. Available: <http://www.mcafee.com/us/resources/reports/rp-threat-predictions-2011.pdf>