The DroidKnight: a silent guardian for the Android kernel, hunting for rogue smartphone malware applications

M. A. Akbar, Farrukh Shahzad and Muddassar Farooq

Abstract

Smartphone devices have become an important enabler for providing m-services in a ubiquitous fashion; as a result, malware attacks on smartphone platforms have significantly escalated. Since the users are using smartphones for m-commerce based financial transactions (and also store sensitive personal data on them); therefore, they must be protected against the rising threat of m-malware. In this paper, we present a realtime malware detection framework for Android platform that performs dynamic analysis of smartphone applications and detects the malicious activities through in-execution monitoring of process control blocks (PCB) in the Android kernel. A novel scheme is utilized to mine the hidden execution patterns - from time-series PCB logs of Android applications - by using information theoretic measures, frequency component analysis and statistical analysis techniques. With the help of this novel scheme, the framework sits in the Android kernel as a loadable kernel module and is able to detect real world malware applications for Android with low false alarms. We have validated the framework using real world Android malware (from well-known malware repositories) and popular benign applications taken from Google's official app store for Android i.e. Google Play Store. By carefully designing a series of experiments, we evaluate the detection and runtime performance of DroidKnight. It is able to detect zero-day (previously unseen) malicious applications with over 90% accuracy, while keeping the false positive rate below 3%. Its runtime processing overhead is less than 4% on a low-end smartphone. Android kernel deserves such a silent guardian and a watchful protector for the Android users against mobile malware. The framework is generic and could be easily adapted to any Linux based mobile platform.

1 INTRODUCTION

Smartphones are becoming an important enabler for enhancing efficiency and effectiveness of users by providing them the ability to remain connected with the m-services in a ubiquitous fashion. This connectivity makes them an important tool of modern m-business and m-transaction frameworks. The users do internet surfing and emails and remain connected on the social networks. The findings of a recent survey are: in the first quarter of 2013, the smartphone market has increased to 50% of the total mobile phone market [1].

Most smartphones come with their dedicated operating system that allows freelance developers to develop third party useful applications that users can download. Recently, Android OS of Google, has revolutionized the smartphones by capturing more than 74% of the market. This popularity, however, has attracted the attention of intruders as well. They are writing malicious applications that users are tricked to download and install by exploiting social engineering techniques. It has been reported that more than $175,000$ Android malware existed in Sep 2012 [2]. A number of signature-based anti-malware solutions are available for Android platform

• *Farrukh Shahzad (*corresponding author – farrukhshahzad0@yahoo.com*) is with FAST National University of Computer and Emerging Sciences, A.K. Brohi Road, H-11/4, Islamabad. M. A. Akbar and Muddassar Farooq (*{ali.akbar, muddassar.farooq}@nexginrc.org*) are with the Next Generation Intelligent Networks Research Center, Institute of Space Technology, Islamabad, Pakistan.*

[3]. It is a well known fact that static signature based systems can be evaded by employing simple obfuscation techniques. In comparison, dynamic non-signature based techniques suffer from small detection accuracy, large false alarm rate and large detection delay. In case of smart phones, dynamic techniques have to overcome another challenge: limited availability of battery, memory and computing resources.

In this paper, we present an initial prototype realtime tool – The DroidKnight – that applies time series analysis on the features extracted from the Process Control Block (PCB) of a process. The tool employs information theoretic analysis on logged time-series features, followed by segmentation and frequency component analysis, to detect Android malware. To validate the proof-of-concept, we have collected 50 benign and 50 malware Android applications from Android store. The results of experiments indicate that the framework is able to detect unseen malware with more than 90% accuracy and less than 3% false alarm rate. The system degrades the performance of a low-end Android phone by 4% only.

1.1 Organization of the Paper

The rest of the paper is organized as follows. The characteristics of benign and malware datasets are explained in Section 2. In Section 3, the major components of proposed framework are presented and the working of each component is discussed. The performance evaluation of the framework is presented in Section 4. Finally, we conclude the paper with an outlook to the future work.

2 Android - Malware & Benign Datasets

The credibility and reliability of any intelligent malware detection system is directly dependent on the quality of malware dataset. We have selected well known and representative applications – 50 malware and 50 benign – to train the DroidKnight system. In Table 1, the short listed benign and malicious applications are tabulated.

2.1 Benign Dataset

The top featured applications at Google Play Store for Android are selected as benign applications[1]. A good mix of different categories – Games, Image Viewers, Sketching tools, Text Editors, Image Editors, Android Utilities such as Recorder, Dialer, Maps etc., and Misc. applications – is selected from the month of August 2012. Special efforts are made to ensure diversity and maintain balance between user-interactive and background applications.

2.2 Malware Dataset

We took malware samples for Android from a publicly available collection of Android malware site – *Contagio Mobile Malware Mini Dump*[2] – because COTS AV companies were unwilling to share their datasets. The selected malicious applications belong to Trojans, Adware, Rootkits, Bots and Backdoors categories. The category of "trojan/spyware" Android applications dominate our malware dataset. Most trojans apparently behave like benign applications and malicious activity covertly happens in a trojan code inserted in a game.

2.3 Creation of Training & Testing Datasets

In order to have rigorous and stress evaluation of the DroidKnight, it is evaluated by using *10-fold cross validation* methodology. In this methodology, ten training and testing combinations are created. The DroidKnight is trained on 90% of the samples and tested on the remaining 10% of the samples.

1. https://play.google.com/store
2. http://contagiominidump.blogspot.com/

TABLE 1
List of Benign and Malicious Apps in Collected Dataset

Benign Apps		Malicious Apps	
magic.solitaire	po.drawinggame2	DouglaLeaker-sirodougamatome-2	oid.computerlab
mes.penguinfree	q.jacobras.notes	DouglaLeaker-tubedougamatome-5	om.button.phone
my.handrite	r.background.hd	DouglaLeaker-uubedougamatome-1	ont_thouch_lite
n.droidhen.car3d	r.km.draw.sketch.txt	e.VoiceChangeIL	oregame.drakula
ndroid.bestapps	reebanana.notes	easy.jewels.Gel	per.Client.Game
ndroid.calendar	rks.ImageViewer	enkov.protector	per.mobi.eraser-3600
ndroid.settings	roid.app.camera	er.android.main	pl.byq.new
nerking.pinball	roid.breakblock	etagmedia.metro	Plankton-c.livewallpaper
nsweringMachine	rzysiek.afc.iqt	Fake-token.A–token.generator	Plankton-com.p1.chompsms
o.game.forestman	s.noteeverything	il.co.egv	Plankton-lsuebutterflies
o.perfectviewer	s.PicassoMirror	Instagram-are.app	Plankton-SawPuzzle.Tales
odle.restaurant	s.unblockmefree	isarun.Ipad2App	Plankton-SkatingMadnessP
om.dreamhawk.ea	simple.a	Loozfon-fa.lin.ero	Plankton-zoric.celebrity
om.game.BMX_Boy	space.fliqnotes	Loozfon-ll.ap.ken	reenguru.finger
ord.game.bubble	ssoft.trialdemo	m.mediawoz.gotq-2-Dup	roid.liveprintsliveprints–livewallpaper
otesclassiclite	tems.Super_Note	m.mediawoz.gotq	rootsmart-e.android.smart
otificationnote	tjsp.memowidget	m.safesys.myvpn	sileria.alsalah
otoxmayhem1lite	torultimatefree	m.space.sexypic	SMSZombie-aper.livepicker
oumiak.desknote	uicknotes.views	mannd.sdbooster	SMSZombie-com.gmdcd.pic
ovio.angrybirds	ulmach.textedit	mes.droid.mhunt	SMSZombie-com.hxmv696.pic
ox.notepad.free	ung.app.fmradio	miniarmy.engine	SMSZombie-com.ldh.no1
oy.sketchmovie1	usoft.punchmemo	nstantheartrate	SMSZombie-com.lzll.pic
p.game.vszombies	w.jsimagefinder	nwhy.WhackAMole	SMSZombie-com.xqxmn18.pic
p.voicerecorder	x.android.sketch	oid.afdvancedfm-2150	SMSZombie-om.zqbb1221.pic
ping.lifequotes	___cing.spades_ads	oid.afdvancedfm-2200	yen273.app2card

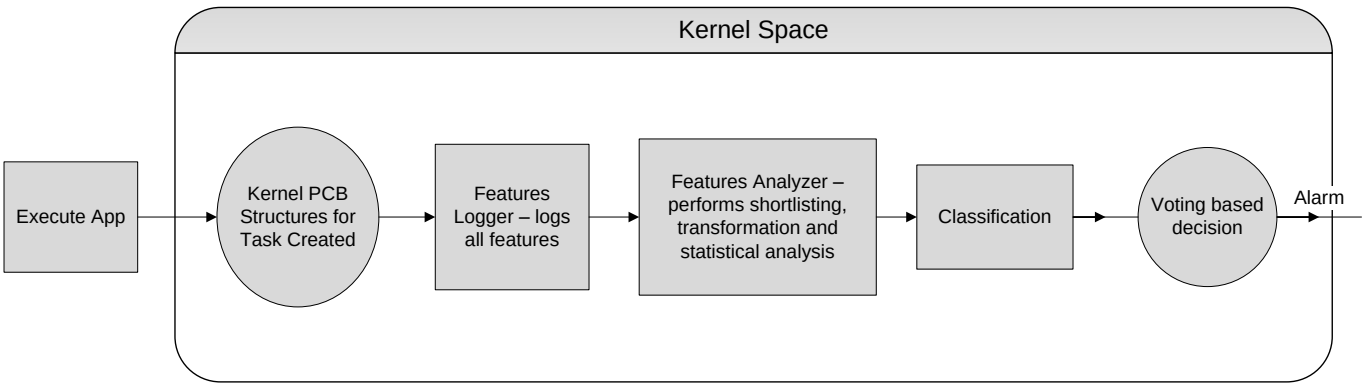


Fig. 1. Block diagram of the Malware Detection Framework's Architecture

3 MALWARE DETECTION FRAMEWORK

In this section, we present the architecture of the DroidKnight. To mine the difference in execution patterns, the framework accumulates the variance of time varying features of a process control block (typically referred to as Task Structures (`task_struct`) in Linux/Anroid).

The Binder Inter Process Communication (IPC) mechanism sends a message to the Zygote process – a special process that is an instance of Dalvik VM with core libraries loaded as read-only – when a new application is started on an Android phone. The Zygote process forks a new process to launch a new application in a new Dalvik VM without the need to copy the shared core libraries [4]. During a `fork` call, the kernel creates a child process and adds its process control block (`task_struct`) to a doubly linked circular linked list. The processes from this list are scheduled in a round robin fashion by sharing the time of a processor.

The framework runs in the kernel space as a (Loadable kernel module) LKM and root privileges are required to load and execute it. The kernel module is compiled and tested on a Samsung Galaxy Young device with Android Gingerbread distribution (Android 2.3.6, Kernel version 2.6.35.7). The architecture of DroidKnight is shown in Figure 1 and it consists of three modules: (1) features logger, (2) features analyzer and processor, and (3) classification engine. Now we explain the functionality of each component.

3.1 Features Logger

The feature logger is implemented as LKM and it periodically stores the fields of the `task_structs` of different running processes. It selects only 32 features from 99 fields available in `task_struct` of a process in Android kernel.

3.2 Features Analyzer

The features analyzer and processor removes redundant values of a feature and applies time-series Discrete Consine Transform (DCT) on the short-listed features. It also accumulates variance of the extracted features that are useful in identifying hidden patterns. It applies the following five steps to extract hidden information from the logged time series features: (1) analyze the extracted features, (2) filter relevant features only, (3) remove redundant features, (4) create windows (batches) of fixed time intervals, (5) apply different time series transforms, and (6) finally, calculate the statistical features – capable of detecting mobile malware – from the transformed dataset.

3.3 Classification Engine

After accumulating the variance of frequency components of current time-series segments that model the execution behavior of a process, the machine learning classifier is delegated the task to build benign and malware models. The next step is to select an appropriate machine learning classifier.

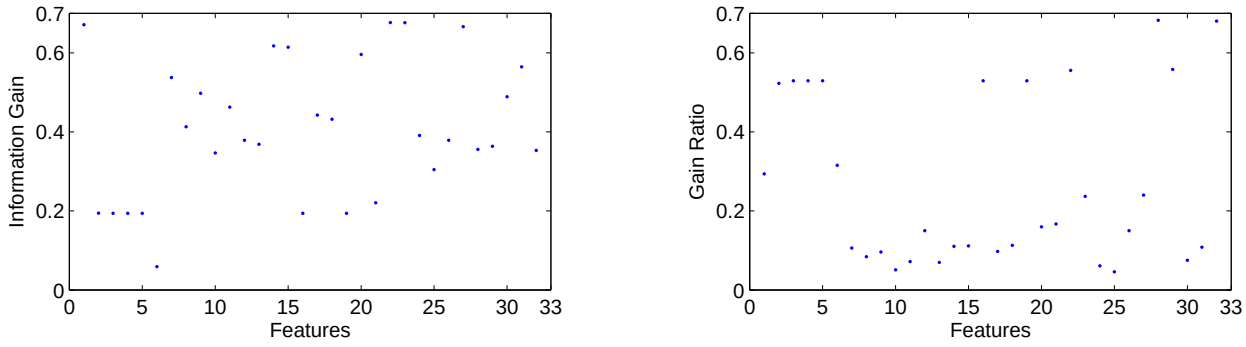
In order to systematically analyze the features' set, Information Gain (IG) and Information Gain Ratio (GR) of extracted frequency components of selected features are plotted in Figure 2. It is clear from the figure that selected features have high information gain and information gain ratio. It is a well known fact that for such a features' set, decision tree classifiers such as J48 appears to be a suitable candidate. Moreover, the authors of [5] have concluded that J48 is resilient to class noise because it avoids over fitting during learning and also prunes a decision tree for an optimum performance (a characteristic desired in resource constrained smartphones).

3.4 Voting Method & Alarm

The classifier gives *Benign|Malicious* decision, after analyzing the accumulated variance of the frequency components in each time-series segment. The majority voting method is used to classify an executing application: if more than half of segments are declared malicious in a window of consecutive segments, the application is declared as malware and is killed; otherwise, the application is allowed to execute.

4 Performance Evaluation

Now we report the accuracy of DroidKnight tested on a dataset consisting of 50 benign and 50 malicious real-world Android applications.



(a) Information Gain

(b) Information Gain Ratio

Fig. 2. Information Theoretic Analysis of Frequency Components Related to Shortlisted Features Set

4.1 Classification Performance

In a typical two-class problem, such as detecting a process as malicious or benign, the classification decision may fall into one of the following four categories: (1) true positive (TP), classification of a malicious process as malicious, (2) true negative (TN), classification of a benign process as benign, (3) false positive (FP), classification of a benign process as malicious, and (4) false negative (FN), classification of a malicious process as benign. The classification performance is measured using two standard parameters: (1) Detection Rate ($DR = \frac{TP}{TP+FN}$), (2) False Alarm Rate ($FAR = \frac{FP}{FP+TN}$).

As mentioned in Section 2, 10 different training and testing datasets are generated. The results strongly indicate that the framework can detect unseen malware applications on Android with a detection rate of more than 90% and a false alarm rate of less 3% on the average. This means that the framework detected at least $8-9$ out of every 10 randomly chosen malware applications. Similarly 2-3% benign applications (randomly chosen) might be classified as malware. This proves our thesis that accumulating the variance of the fields of a process control block and applying time series analysis on them provides the ability to distinguish a malicious application from a benign application.

In these experiments, we have used a *time resolution* (the time interval after which values of parameters in the process control block structure of a process are logged) of 10ms, and a voting window size of 30 consecutive segments. These values are empirically chosen to optimize performance in terms of DR and FAR.

4.2 Performance Overhead

DroidKnight will never be used by Android users if its running overhead hampers their usability experience. In order to analyze the performance overhead, the experiments are performed on a Samsung Galaxy Young S5360 device with an 832 MHz ARMv6 processor, 290 MB RAM and 160 MB built-in storage. (Such a low end device was deliberately selected.)

We have created a mix of customized Android applications that are CPU-intensive and I/O-bound; while others are a combination of both. The memory requirements for applications vary as well. These operations include running different algorithms for compressing files, copying a file (folder) to another location and doing text search within a file. Four different folders, containing different files (total folder size: 425 MB, 500 MB, 850 MB and 1275 MB,) have been

TABLE 2
System Performance Degradation Analysis on Android

Application	Baseline Exec. Time (s)	Exec. Time with our framework (s)	Overhead (%)
Zip (Archive)	127.2	131.5	3.27
Zip (Best Compress)	67.86	69.94	2.97
Zip (Best Speed)	50.39	52.01	3.11
Zip (Deflated)	132.12	141.81	6.83
Zip (Filtered)	147.59	154.27	4.33
File Copy	190.6	196.95	3.22
File Search	131.68	134.91	2.4
Average			**3.73**

used as input to these applications. The average performance degradation is reported in Table 2. It is apparent that an average performance overhead of (**4**%) will not degrade the usability experience significantly.

5 Related Work

To the best of our knowledge, no dynamic non-signature based technique exists in the literature that tries to apply the concept of "accumulating the variance of information in the fields of `task_struct` of an Android kernel. For completeness, we just discuss most relevant techniques.

Dini et al. have presented a multilevel anomaly detection technique for detecting Android malware (*MADAM*) [6]. The proposed framework operates in the kernel and the user space simultaneously and is capable of detecting previously unseen, zero-day malicious applications. A rich feature set is derived due to a multilevel view of the system events in both spaces. The K-nearest neighbors (KNN) algorithm is then applied during classification process. The operation of the framework can be divided into training, learning and operation phases. The machine learning classifier can adapt to new changes by incorporating new feature vectors in training and learning set at run-time. An average detection rate of 93% along with an average false positive rate of 5% is reported for a very small dataset.

Android Application SandBox framework is proposed in [7] for malicious software detection on Android. This framework first performs static analysis of user applications. The applications are then transmitted to a remote server that executes them in the sandbox. Then it performs clustering and does analysis of the generated logs to detect malicious patterns. The dataset used for testing Sandbox is small and analysis of the time and memory overhead is also not included. Another framework that uses a similar concept of decoupled security is an API based malware detection system *Paranoid Android* [8].

Burguera et.al have developed a dynamic framework called *Crowdroid* which recognizes Trojan-like malware. It takes into account the fact that genuine and trojan affected applications differ in types and number of system calls during the execution of an action that requires a user's interaction. The authors have reported results on a small dataset. The false-alarm rate is significantly high (20%). The authors have not discussed the robustness of their features set [9].

Andromaly is another IDS which monitors both the system and user behavior by observing several parameters, spanning from sensors activities to CPU usage [10]. The authors have developed four custom malicious applications to evaluate the ability to detect anomalies. They have created four different training/testing scenarios and reported good detection accuracy. Andromaly degrades performance of a smartphone by 10% with the malware detection time of 5 sec. Also, it is not evaluated over real world Android malware dataset.

To conclude, DroidKnight achieves not only high detection rate (more than 90%) but relatively small FAR (below 3%) with only 4% deterioration in performance.

6 Conclusion & Future Work

The major contribution of this paper is a dynamic malware detection tool (for Android smart phones) – DroidKnight– which has the capability to detect mobile malware while it is still executing. The proposed framework is lightweight with an average performance overhead of less than (**4**%). The framework is evaluated on a real world dataset consisting of 50 benign and 50 malware Android applications. The reported experiments show that the framework is able to achieve more than 90% accuracy and less than 3% false alarm rate. The framework can work with a number of existing solutions to provide a robust solution against rootkits.

In future, the plan is to build a formal mathematical model of "accumulative variance" and use it to further enhance the accuracy of DroidKnight and lower its FAR. Another direction for work is: to evaluate it on a larger dataset that consists of more than 100 benign and 100 malware applications. This will be the subject of forthcoming applications.

Acknowledgments

The work presented in this paper is supported by the National ICT R&D Fund, Ministry of Information Technology, Government of Pakistan. The information, data, comments, and views detailed herein may not necessarily reflect the endorsements of views of the National ICT R&D Fund.

References

[1] Gartner, "Gartner says asia/pacific led worldwide mobile phone sales to growth in first quarter of 2013," *http://www.gartner.com/newsroom/id/2482816 [last-viewed-June-3-2013]*, 2013.

[2] Trend-Micro, "3q 2012 security roundup: Android under siege: Popularity comes at a price," *Research and Analysis*, 2012.

[3] S. HILL, "Top 3 android security apps, do they protect you?" *http://www.digitaltrends.com/mobile/top-android-security-apps/(last-viewed-on-December-10-2012)*, 2012.

[4] D. Ehringer, "The dalvik virtual machine architecture," *Techn. report (March 2010)*, 2010.

[5] I. Witten and E. Frank, *Data mining: Practical machine learning tools and techniques, second edition*. Morgan Kaufmann, 2005.

[6] G. Dini, F. Martinelli, A. Saracino, and D. Sgandurra, "Madam: a multi-level anomaly detector for android malware," in *International Conference on Mathematical Methods, Models and Architectures for Computer Network Security*, 2012, pp. 240–253.

[7] T. Blasing, L. Batyuk, A. Schmidt, S. Camtepe, and S. Albayrak, "An android application sandbox system for suspicious software detection," in *International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2010, pp. 55–62.

[8] G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid android: Versatile protection for smartphones," in *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 2010, pp. 347–356.

[9] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2011, pp. 15–26.

[10] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "Andromaly: a behavioral malware detection framework for android devices," *Journal of Intelligent Information Systems*, pp. 1–30, 2011.