InstDroid: A Light Weight Instant Malware Detector for Android Operating Systems

Saba Arshad
Department of Computer Science
COMSATS Institute of Information Technology
Islamabad, Pakistan

Munam Ali Shah
Department of Computer Science,
COMSATS Institute of Information Technology
Islamabad, Pakistan

Rabia Chaudhary
Department of Computer Science,
Bahria University
Islamabad, Pakistan

Neshmia Hafeez
Department of Computer Science,
COMSATS Institute of Information Technology
Islamabad, Pakistan

Muhammad Kamran Abbasi
Department of Distance Continuing & Computer Education
University of Sindh
Hyderabad, Pakistan

***Abstract*—With the increasing popularity of Android operating system, its security concerns have also been raised to a new horizon in past few years. Different researchers have introduced different approaches in order to mitigate the malware attacks on Android devices and they succeed to provide security up to some extent but these antimalware techniques are still resource inefficient and takes longer time to detect the malicious behavior of applications. In this paper, basic security mechanisms, provided by Google Android, and their limitations are discussed. Also, the existing antimalware techniques which lie under the basic detection approaches are discussed and their limitations are also highlighted. This research proposes a light weight instant malware detector, named as InstDroid, for Android devices that can identify the malicious applications immediately. Through experiments, it is shown that InstDroid is an instant malware detector that provides instant security at low resource consumption, power and memory, in comparison to other well-known commercial antimalware applications.**

***Keywords*—Android; static; resource efficient; power consumption; memory; detection rate; accuracy**

I. Introduction

Smart phones have become a necessary part of everyday life. From businessman to a common person, everyone uses smart phones to perform different tasks depending upon their needs. Android devices provides attractive and easy to use features to the users due to which they are known as most popularly used devices from previous few years [1]. Android phones store the critical data related to the personal as well as professional life of a person. This data can be in the form of important transaction details, pictures, SMS and official encrypted files. It is important to ensure the security of such data in smart phones. Large number of malwares had been designed to infect and intrude into the smart phones in order to exploit the privacy of the user [2]. The mobile malware designers exploit the vulnerabilities that exist in the Android operating system. Android operating system is an open source platform that allows the installation of third party applications from App-stores other than Google play store for example PandaApp [3] and GetJar [4]. This openness becomes the opportunity for malware developers to harm the user's data and is the reason for several issues such as invalid access from one resourceful application to the other (information leakage), permission escalation, repackaging application to infuse malicious code and Denial of Service (DoS) attacks.

In order to mitigate these issues, researchers have developed lot of detection systems by using different approaches to ensure the security up to some extent. The basic approaches used by malware detection approaches includes static analysis and dynamic analysis. Static analysis techniques monitor the behavior of application without running the application on device. It scans all the code of application without running the application due to which it is not able to detect the runtime malicious behavior of applications. In dynamic analysis technique, run time behavior of application is monitored by executing the application on emulator or real device for a specific time period. These analysis techniques enable the antimalware systems to identify the malicious applications and protect the Android devices.

Android smartphone devices are usually resource constrained. They have limited battery power and storage. Due to this reason, detailed static and dynamic analysis cannot be performed on Android devices. In order to overcome this limitation, researchers have developed cloud based malware detection systems. Although these security systems shift the workload from mobile device to cloud server, but the service becomes expensive and network dependent. If the detailed analysis at server takes longer time, it is possible that during

this time period, the malicious application might get the control over device and compromise the device. An efficient and very light weight system is the necessity of time which can provide protection to Android devices against known malware types and their variants at the instant when the application is installed on the device at very low resource consumption.

In this research, InstDroid, a light weight malware detection system, is proposed that can provide instant detection of malicious applications as soon the user will install the application. It immediately identifies the malicious applications through quick scan and notifies the user about it. The heavyweight Android malware tools consume a lot of power and memory while the smart phones are constrained by resources. InstDroid is able to detect the malware using very negligible amount of hardware resources of Android devices, thus not affecting the performance of the device.

Rest of the paper is organized as follows: Section II discuses about basic security mechanisms provided by Google Android to the Android devices and user's data. Basic approaches for malware detection, static and dynamic analysis, and deployment systems are discussed in Section III. Section IV describes about the proposed malware detection system, InstDroid. The experimental results are explained in Section V and Section VI concludes the paper and future work is also discussed in this section.

II. Basic Security Mechanisms & Their Limitations

This section discusses the basic security mechanisms provided by Google Android and their limitations. These security mechanisms include permission framework, application sandboxing and Bouncer, shown in Fig. 1.

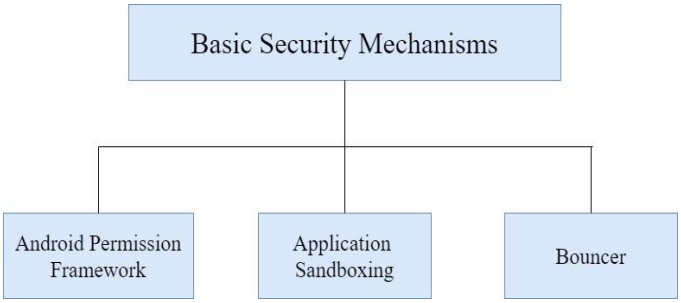

Fig. 1. Basic security mechanisms provided by Google Android.

A. Android Permission Framework

By default, an Android application has no permissions linked with it until the application requires special resources in order to operate. Different permissions have different purposes associated with them but they are used in order to limit the access of the application to the critical resources of device such as camera, SMS storage and Bluetooth permissions, etc. After careful inspection of these permissions, it is up to the user whether he wants to install the application or not [12]. There are four major categories of permissions: Normal, Dangerous, Signature and SignatureOrSystem [22]. Normal permissions are low level permissions that allows the (requesting) application to access the restricted application level features with only minimum level risk attached to other applications, the system, or the user. Dangerous permissions are high risk level permissions and can be consequently used to harm the user's device and data. Signature and SignatureOrSystem permissions are only used by the system applications or the applications which are added by the manufacturer. Any user application requesting such permissions can be malicious. Although, permission system provides information to users about applications behavior up to some extent but due to lack of technical knowledge about these permissions and their use, by the applications, users usually ignore the permissions and simply install the applications. This makes Android permission mechanism completely ineffective to provide security against the access of unnecessary resources by newly installed application, which might be malicious.

B. Application Sandboxing

Android uses application sandboxing mechanism which separates the application associated data and code implementation from other applications. Each Android application runs within its separate space or sandbox, having no conflict with other applications or interaction, unless a particular application has been assigned special privileges to communicate with other applications. For better protection of Android application's data, Android kernel executes the Linux Discretionary Access Control (DAC) to efficiently manage and protect the device from getting misused. Each application process is protected with an assigned unique ID (UID) within its isolated sandbox [13]. The isolated application communicates with each other through a method known as Inter-Component Communication (ICC) or Binder. Android middleware allows the ICC between different components of the application. The ICC very smoothly takes care of transferring the request from user to the destination applications. After that applications can access the components or services of other applications as a service [12]. This ICC process is used by malware applications too in order to control the other applications and perform malicious activities on the device. Privilege escalation or permission escalation attacks were actually possible because of the loopholes that exist within the Android operating system, in order to get access to the assets that are hidden or protected from the user of application. This series of attacks can result into the leakage of fatal information because of the unauthorized access of resources to the application than the intended access of resources. Android applications might have such components that have been added into it through external resources. In this case these exported components can be misused in order to get the access to critical permissions [11].

C. Bouncer

Bouncer is a malware detection tool deployed at Google Play Store for the analysis of all the applications available at Google Play Store. The main purpose of the bouncer is to provide a security check looking for malicious software containing malware, spyware, and Trojans. This kind of applications can be used to intrude the privacy of the user, selling it to the blackmailers or using it for more harmful purposes. Bouncer keeps on analyzing the applications continuously. If any application is detected as malware, it is instantly removed from the Play Store. Although, Bouncer performs its job very well but still there exist some malware

applications on Google Play Store that remains undetected by Bouncer, reported in a research [5].

III. Malware Detection Approaches

In spite of the security mechanisms provided by Google Android, malware attacks are increasing every year [6]. Lot of research has been done to protect the Android devices from malware attacks. Major approaches used for the malware analysis includes static analysis and dynamic analysis.

A. Static Analysis

Static analysis techniques monitor the behavior of application without running the application on device. Kirin [7], Drebin [8] and RiskRanker [9] are well known examples of antimalware techniques which performs static analysis to explore the static features of Android malware. It scans all the code of application but cannot detect dynamic loading of malware code. Also, the encrypted malicious code remains undetected. In [10] authors have categorized static analysis based malware detection techniques as signature based malware detection, permission-based malware detection, and dalvik byte code malware detection. The signature-based detection technique extracts the signatures of the applications and then matches it with the database of known malware signatures [9]. *AndroSimilar* [11] and *DroidAnalytics* [12] are signature based detection systems.

Permission based detection is a light weight malware detection method which also falls under the category of static analysis. In [13], authors have proposed the system which performs analysis on permissions declared in the Android manifest file and then analyzes if the application is over privileged or not. In the manifest file of the application, they extract three major features i.e. permissions, intent filters, process number and a total number of predefined permissions. On basis of these features, they compare it with the list of already known keywords. They tested 365 samples on the total to determine the efficiency of the proposed system. The proposed system almost provides 90% detection rate. In [14], [15] and [16], authors have also used permission based detection method.

Dalvik byte code analysis performs the instruction level code analysis to find out the malicious behavior of the applications. But it occupies more storage space due to the instruction level analysis of the code and hence consuming more power resources, therefore making it less likely to be more productive on resource constrained devices like smart phones [17]-[19].

B. Dynamic Analysis

Dynamic analysis technique provides run-time monitoring of the applications. *TaintDroid* [20], *DroidRanger* [5] and *DroidScope* [21], use the dynamic analysis to monitor the run-time behavior of the application. Dynamic analysis can detect the dynamic malicious payloads.

DroidDolphin [22] uses dynamic analysis that takes support of GUI-based testing, big data and machine learning for the detection of Android malwares. API calls are monitored by *API Monitor* [23] during execution of apk. Logs are collected by installing instrumented apk file on virtual device of Android. Sandboxing is done through *DroidBox* [24] for having dynamic logs. Testing tool, *Monkeyrunner*, is combined with *APE* [25], that is used for GUI based event simulation. Events are represented by n-grams and features are given as input to *Support Vector Machine* [26] algorithm that classifies the applications. Emulation and testing phases become complex for future testing because of large data set.

CopperDroid is presented in [27] that works on top of QEMU and performs dynamic analysis. Behaviors are analyzed by system calls tracking and centric analysis. The *CopperDroid* analyzes malware by information extraction from manifest file. The *CopperDroid* was evaluated for two sets of malwares and there is no static analysis involved.

Although dynamic analysis overcomes the limitations of static analysis, but it can only analyze the code which executes during monitoring interval and is not able to detect malicious code which does not execute during monitoring period.

C. Cloud Based Detection

These analysis approaches, static and dynamic, can be used at either mobile device or at cloud for detection of malwares. As mobile devices are resource constrained due to which malware detection systems cannot perform detailed and effective analysis on mobile devices. To develop an effective and accurate malware detection system, researchers have deployed the analysis and detection mechanism at clouds.

A cloud based intrusion detection and response framework was developed and discussed in [28], that analyzes behavior of a device and in case of unusual events, it performs different appropriate actions. This framework can work with minimum resources and can produce real and accurate detection and responses for registered devices. A key point of this architecture is to copy user inputs in real time. Proxy settings are configured by installing a software and proxy server replicates the conversation between internet and device and sends it to emulated environment for malware detection and analysis. A light weight agent is also involved for gathering info, sending it to emulated environment and waiting for responses and actions. Proposed framework was deployed to Android-equipped HTC Droid Incredible devices but attack graph does not automatically take actions in an emulated phone environment, like computer systems.

TABLE I. Cloud-Based Android Malware Detection Techniques

Ref.	Year	Implementation	Limitations
[28]	2011	Working prototype	Android-equipped HTC Droid Incredible devices and attack graph does not work for emulated devices
[29]	2014	Framework	Need device user, app store and security professionals' association
[30]	2012	Security system	Cloud can be crashed because of single component failure
[31]	2012	Architecture	Needs number of detection engines
[32]	2014	Security Mechanism	Mobile interference is less due to of cloud services
[33]	2015	Experimental	Requires different configurations

TABLE II. RESOURCE UTILIZATION ANALYSIS FOR ANDROID MALWARE DETECTION

Ref.	Year	Implementation	Evaluated Parameters	Limitations
[34]	2013	Prototypes	Battery level, FPR, cutoff drop value	Cut off values may affect the results
[35]	2016	Experimental	Accuracy, FPR, FNR	Specific pattern for resource utilization was not considered
[36]	2014	Prototype	FP percentage	Need user efforts and time to create profiles

In [29], authors proposed a cloud based detection and prevention approach. When a user makes request for any application, the request is sent to known libraries. If the application is found in libraries then it is declared as safe or malicious, on the basis of classification of that application. If application is not found in libraries then application is declared as unknown and send to malware detector that downloads the application. The malware detector performs both static and dynamic analysis and declares the application as safe or malicious for users on the basis of classification results. All these operations are performed at cloud, that keeps resources of mobile devices conserved. Mobile devices just deal with libraries for finding application classification, as safe or malicious. The major limitation of this technique is that it is highly dependent on the Internet services and cloud system. If any component at cloud fails to perform its operations, security will not be provided. This approach requires mobile users, app stores and IT security professional's association.

Qian *et al.* [30] proposed a cloud based security system which provides security to Android devices by detecting malwares, pours out harmful application and provides data backup facility. Android devices have an agent/client that communicates with the cloud. Connection between client and server should be fair enough for sending malicious applications to cloud. Authors presented agent and server modules to elaborate the system clearly. Different features were implemented that provide security. VPN builds connection between device and cloud for user safety. A transparent proxy is used to communicate data between internet and proxy server that provides security to users. Malicious applications can also send information to suspicious addresses. Push function is used to discard illegal packets that are sent to devices. Management server has facility to detect malicious applications by running different algorithms that may be available in market or may use static, dynamic zero-day analysis programs in an emulated environment or can be executed on the PC. Backing up of data is also maintained at cloud. Proposed system uses limited device resources but the service might be expensive for the users.

The security system proposed in [31], contains a host that works with the cloud provided services and it has a vast range of signature database. Different detection modules can be made run simultaneously. Virtualization helps a lot to detect malware and large number of users can be scaled over the network. Proposed system provides services such as creating a clone of the device and a proxy in cloud is used for identifying memory, system calls invoked on run time. Different open source antiviruses are used to detect malwares. Host agent is a process that is installed on the device. It performs inspection on files and compares the files against a cache of files. If file is absent in cache it is sent to the cloud for further analysis and recovery actions are taken accordingly. After analysis, it is placed on local and cloud caches. This approach needs number of detection engines to provide large detection exposure.

According to the research performed in [32], proposed system consists of three modules. First module classifies applications as light, heavy, medium, very light and very heavy, based on the signatures, permissions and services etc. Second module has local server that creates all user's feedback. Package name for feedback, date of report, IMEI number for report receiving and report that has '1' and '0' values for good and bad applications. In third module, filters are applied to applications for permission set and the generated report is sent to server. Algorithm is used for malware detection and works on confidence index. If confidence index is greater than 50 %, there is possibility of malware if not then application is considered to be safe. Mobile resource consumption is less due to the use of cloud services.

Table 1 shows cloud-based detection for malicious applications in Android. Cloud-based detection requires internet availability, detection engines, files uploading on cloud which consumes large amount of power. Major limitations of such techniques include that any component failure at cloud may affect the whole detection system. Mobile or host device have to wait for the cloud response in order to provide security on Android devices.

D. Resource Utilization Based Detection

Although cloud based detection systems allow deep analysis of applications but at the cost of heavy servers and they are dependent on cloud server's response. Also, the power consumption at mobile device increases if the device is at large distance from the server and communicates with cloud server for detection purpose. Many researchers have developed malware detection systems to overcome the power consumption limitations of cloud based detection systems.

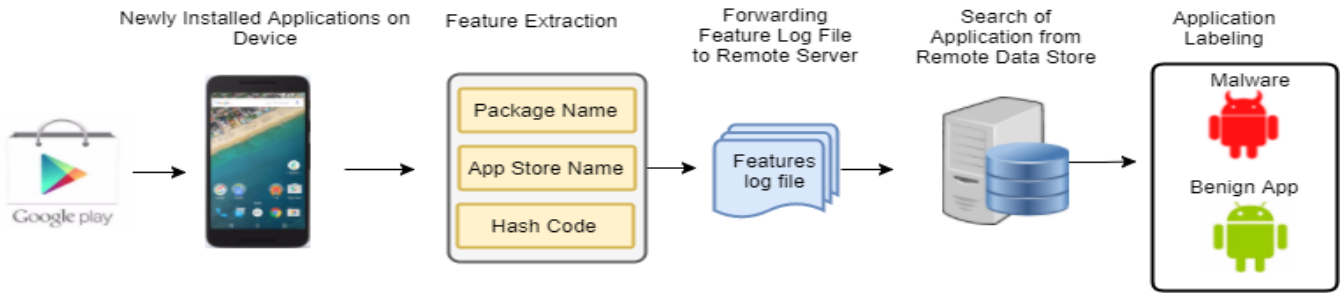

Fig. 2. Workflow diagram of InstDroid.

In [33], authors have observed effectiveness of two techniques for malware detection. Prototypes were developed for Android platform. Techniques include normal and location specific power profiles for phones. Experiments were performed to detect malware and minimizing power consumption. Authors used SMS spam and user tracking simulators for the evaluation of techniques. Normal power profile technique takes power utilization as a time function. Normal battery consumption rate is measured initially after which the system starts monitoring the power drainage pattern. Location power profile works over an extended time, based on the location i.e. whether playing games at home or using browser at airport etc. A program was written by authors to measure power utilization for working models. For first discussed technique cut off value may affect results of prototype. For second discussed technique, anomalies were predicted just for two locations.

Canfora *et al.* [34] proposed a malware detection technique that detects presence of malicious applications by analyzing the device resources such as memory, CPU, and network. Proposed methodology has three components: numerical feature set related to application behavior, a procedure in which applications are executed in a balanced environment and performs data collection, method for analyzing the collected data. Monkey tool was used as a debugger. Data is analyzed by using machine learning techniques.

Three different detection techniques are mentioned in [35] that are used in Android malware detection for testing and data collection. These techniques include location based detection, time based detection and a hybrid, combination of both. The basic idea of these techniques is to investigate the usage of battery profiles to detect malwares. Battery usage will be more in case of malware attack. In first technique, profiles are created for normal battery usage, based on the user location, because battery usage may vary depending upon location. Second technology creates profile, based on time in which user uses the Android device. Third technology involves hypothesis that user uses Android device differently at different locations in different timings. SMS spam and location tracking simulations are performed by authors. Data collection and location based detection is done by standalone prototype. Data needs to be segmented after assortment correspondent to fall in battery level between two data points and average rate of charge per second. Standard deviation is calculated for each segment by standalone project. Abnormal battery usage is observed when a new segment is created for a location. Segments are also monitored for hours but during period of 6 hours, segments produce better detection results. When both these techniques are combined, false positive rate is reduced. A program is written to measure battery usage of the prototype by authors. Random values for location and time data segments were taken and tested for two simulators. Profile creation for specific location involves user presence at that location at different time.

Table 2 shows different techniques that are developed for enhancing the resource efficiency in terms of power. Keeping in view all the limitations of malware Antimalware techniques, discussed in literature, an instant malware detection system is proposed that can provide instant security against known malware families and their known variants, at low resource consumption.

IV. INSTDROID: THE MODEL

This research proposes a light weight and instant malware detection system for Android devices. This instant malware detector immediately detects the malwares and provides instant protection to Android devices from known malware types. This light weight Android security system consumes very negligible amount of hardware resources of resource constrained Android devices. Fig. 2 depicts the workflow of Instant malware detection system. When an Android user installs any application, InstDroid instantly initiates the detection mechanism and secures the Android devices.

A. Features

Features used for the detection of malicious applications are:

1) *Hash Code:* Hash code generated for application.
2) *Package Name:* Package name of application.
3) *Application Store Name:* Name of market from which the application is installed.

B. Working

Initially, when a user installs the application from Application store, InstDroid gets activated. It generates the hash code of application and extracts the features from the application code statically. Features extracted from the application includes package name and name of application store from which application is downloaded. These features are then forwarded to the remote server which is responsible for making decision about the application's behavior. Remote server contains the database of malware applications. When it receives the application's hash code, package name and App-store name from InstDroid client application, it immediately

looks into the malware database. An application is declared as malicious if one of the two conditions occurs:

a) Any record in the database contains the same package name and App-store name, sent by InstDroid client application.

b) Any record in the database contains the same hash code send by InstDroid client application.

If the application package name and App-store name or hash code is not found in the remote server's database then the application is declared as legitimate.

Once the application is declared as legitimate or malicious, the decision is forwarded to the InstDroid client application which informs user about the application's behavior immediately. Fig. 2 describes the work flow of the proposed system.

V. Evaluation

This section provides the experimental results which we have performed for evaluation of InstDroid. We have used Drebin's dataset of malicious application for identification of malware applications, as this dataset is claimed to be the largest dataset of malware applications.

A. Power Consumption

In the first experiment we have measured the power consumed by InstDroid and compared it with the real antimalware applications such as 360 Security [36], Avira Antivirus [37] and Avast Antivirus [38]. These antivirus applications are commercially available in Google official marketplace.

In most of the detection systems, the security service keeps on running in the background all the time which consequently affects the performance of the device and causes the resource drainage. InstDroid is a light weight detection system which is developed to overcome the limitations of the existing systems. It gets activated only when any application is installed on the device, performs detection mechanism and then stop running in the background. This is how the power consumption at real Android device is very low in comparison to the other malware detectors. Fig. 3 depicts the comparison between InstDroid and other antimalware applications. It can be observed that InstDroid consumes significantly low power in comparison to other devices.

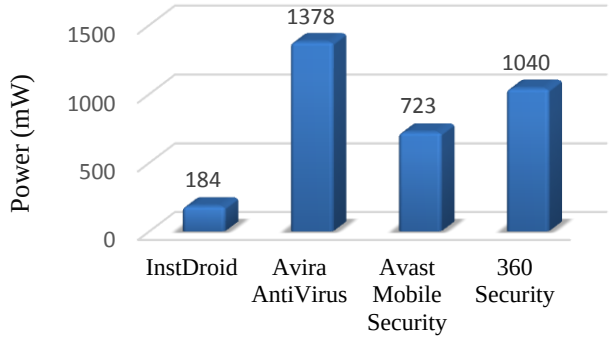

Fig. 3. Comparison of power consumed by different antimalwares.

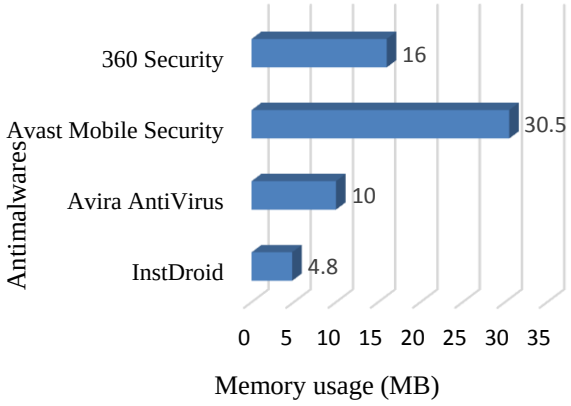

Fig. 4. Comparison of memory usage by different antimalwares.

B. Memory Consumption

The memory consumption and CPU usage of any application is directly proportional to the performance of the device. The large sized antivirus tools provide the efficient scanning of the applications on the cost of reduced performance and battery derail age of the device. The proposed system provides a very light weight mechanism for detecting the malicious properties as it requires very low amount of storage space to perform malware detection. Due to this low resource usage feature of InstDroid, performance of the device is not affected.

In this experiment, InstDroid is evaluated on the basis of memory consumption and the results are compared with the other well-known antimalware Android applications. Fig. 4 depicts the comparison of memory consumption by different antimalware systems. It can be seen that InstDroid is more resource efficient than the other antimalware tools.

C. Detection Time

Time taken by the antimalware system is also an important parameter for the evaluation. In this experiment, InstDroid is evaluated on the basis of detection time, time taken by the security system to detect the malicious behavior of application. Total time taken by the InstDroid to complete the detection process is compared with other antimalware applications. Fig. 5 shows the comparison of detection time between different anti-malwares. It can be seen that InstDroid is faster than all the other applications, just like its name – an instant malware detector.

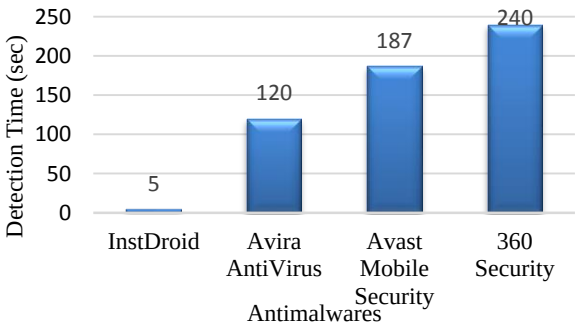

Fig. 5. Comparison of detection time between different antimalwares.

D. Detection Accuracy

In this experiment, the detection accuracy of antimalware system is measured and is compared with other commercial antimalware applications. This experiment is performed on 100 different malware applications and the detection accuracy of antimalware systems is observed, depicted in Fig. 6. Experimental results show that InstDroid achieves highest accuracy.

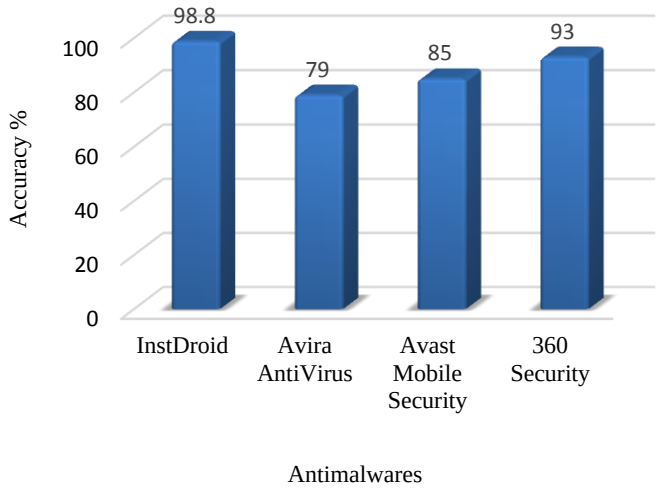

Fig. 6. Comparison of detection time between different antimalwares.

VI. Conclusion and Future Work

With the increasing popularity of Android operating system, its security concerns have also been raised to a new horizon in past few years. Different researchers have introduced different approaches in order to mitigate the malware attacks on Android devices and they succeed to provide security up to some extent but they are still resource inefficient and takes longer time to detect the malicious behavior of applications. If any malware gets installed on the device, it is possible that it effects the device before the antimalware tool knows about the malicious behavior of application. InstDroid is the instant malware detection system which becomes active at the instant when application is installed on the device and in no time, it notifies about the application's classification to the user. It is a light weight malware detector that barely occupies the space of few megabytes and consumes significantly low power in comparison to other antimalware applications.

In future, we aim to enhance the dataset of malware applications so that InstDroid can detect the new malware families and their variants immediately. InstDroid can be integrated with other antimalware systems in a modular form, for instant detection of all the known malwares and their variants. As an example, different malware types and attacks are usually recorded in different countries. For such case, InstDroid can be used with addition of cache mechanism. In such a scheme, the data set of malwares, specific to the country, can be stored in cache for quick detection. This will provide instant detection of malwares and protection against them at low resource consumption.

References

[1] "Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016," 2017. [Online]. Available: http://www.gartner.com/newsroom/id/3609817. [Accessed: 28-Apr-2017].

[2] "Mind the (Security) Gaps: The 1H 2015 Mobile Threat Landscape - Security News - Trend Micro USA." [Online]. Available: http://www.trendmicro.com/vinfo/us/security/news/mobile-safety/mind-the-security-gaps-1h-2015-mobile-threat-landscape. [Accessed: 08-Dec-2015].

[3] "Android.PandaApp.com | Free Your Mobile Life!" [Online]. Available: http://android.pandaapp.com/. [Accessed: 01-Aug-2017].

[4] "GetJar - Download Free Apps, Games and Themes APK." [Online]. Available: https://www.getjar.com/. [Accessed: 01-Aug-2017].

[5] 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 Proceedings of the 19th Annual Network and Distributed System Security Symposium, 2012, no. 2, pp. 5–8.

[6] "Trend Micro Q2 Security Roundup Report | Androidheadlines.com." [Online]. Available: http://www.androidheadlines.com/2015/08/trend-micro-q2-security-roundup-report.html. [Accessed: 08-Dec-2015].

[7] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in of the 16th ACM conference on …, 2009, pp. 235–245.

[8] D. Arp, M. Spreitzenbarth, H. Malte, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in Symposium on Network and Distributed System Security (NDSS), 2014, pp. 23–26.

[9] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "RiskRanker : Scalable and Accurate Zero-day Android Malware Detection Categories and Subject Descriptors," in Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, 2011, pp. 281–293.

[10] S. Arshad, M. Ahmed, M. A. Shah, and A. Khan, "Android Malware Detection & Protection : A Survey," Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 2, pp. 463–475, 2016.

[11] P. Faruki, V. Ganmoor, V. Laxmi, M. S. Gaur, and A. Bharmal, "AndroSimilar: Robust Statistical Feature Signature for Android Malware Detection," in Proceedings of the 6th International Conference on Security of Information and Networks - SIN '13, 2013, pp. 152–159.

[12] M. Zheng, M. Sun, and J. C. S. Lui, "DroidAnalytics : A Signature Based Analytic System to Collect , Extract , Analyze and Associate Android Malware," 2013.

[13] R. Sato, D. Chiba, and S. Goto, "Detecting Android malware by analyzing manifest files," Proc. Asia-Pacific Adv. Netw., vol. 36, pp. 23–31, 2013.

[14] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. Álvarez, "PUMA: Permission usage to detect malware in Android," Adv. Intell. Syst. Comput., vol. 189 AISC, pp. 289–298, 2013.

[15] M. Qiao, A. H. Sung, and Q. Liu, "Merging Permission and API Features for Android Malware Detection," in 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), 2016, pp. 566–571.

[16] K. A. Talha, D. I. Alper, and C. Aydin, "APK Auditor: Permission-based Android malware detection system," Digit. Investig., vol. 13, pp. 1–14, 2015.

[17] Y. Aafer, W. Du, and H. Yin, "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android," in Security and Privacy in Communication Networks, Springer, Cham, 2013, pp. 86–103.

[18] E. R. Wognsen, H. S. Karlsen, M. C. Olesen, and R. R. Hansen, "Formalisation and analysis of Dalvik bytecode," Sci. Comput. Program., vol. 92, no. December 2012, pp. 25–55, 2014.

[19] W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party Android marketplaces," in Proceedings of the second ACM conference on Data and Application Security and Privacy - CODASKY '12, 2012, pp. 317–326.

[20] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: An Information-Flow

Tracking System for Realtime Privacy Monitoring on Smartphones," ACM Trans. Comput. Syst., vol. 32, no. 2, pp. 1–29, Jun. 2014.

[21] L. Yan and H. Yin, "Droidscope: seamlessly reconstructing the os and dalvik semantic views for dynamic Android malware analysis," in Proceedings of the 21st USENIX Security Symposium, 2012, p. 29.

[22] W.-C. Wu and S.-H. Hung, "DroidDolphin: A Dynamic Android Malware Detection Framework Using Big Data and Machine Learning," in Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems, 2014, pp. 247–252.

[23] "API Monitor: Spy on API Calls and COM Interfaces (Freeware 32-bit and 64-bit Versions!) | rohitab.com." [Online]. Available: http://www.rohitab.com/apimonitor. [Accessed: 22-Aug-2016].

[24] "DroidBox." [Online]. Available: https://github.com/pjlantz/droidbox. [Accessed: 22-Aug-2016].

[25] S. Chang, "Ape: A smart automatic testing environment for Android malware," Comput. Sci. Inf. Eng. Natl. Taiwan Univ. Taiwan, 2013.

[26] A. Andrew, "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods," in Kybernetes, 2013.

[27] K. Tam, S. Khan, A. Fattori, and L. Cavallaro, "CopperDroid: Automatic Reconstruction of Android Malware Behaviors.," NDSS, 2015.

[28] A. Houmansadr, S. A. Zonouz, and R. Berthier, "A cloud-based intrusion detection and response system for mobile phones," in 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), 2011, pp. 31–32.

[29] N. Penning, M. Hoffman, J. Nikolai, and Y. Wang, "Mobile malware security challeges and cloud-based detection," in 2014 International Conference on Collaboration Technologies and Systems (CTS), 2014, pp. 181–188.

[30] H. Qian and Q. Wen, "A cloud-based system for enhancing security of Android devices," in 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, 2012, pp. 245–249.

[31] R. S. Khune and J. Thangakumar, "A cloud-based intrusion detection system for Android smartphones," in 2012 International Conference on Radar, Communication and Computing (ICRCC), 2012, pp. 180–184.

[32] M. Patil and M. Shelke, "Revisiting Defense against Malwares in Android using Cloud Services," Int. J. Appl. or Innov. Eng. Manag., vol. 3, no. 3, 2014.

[33] B. Dixon and S. Mishra, "Power Based Malicious Code Detection Techniques for Smartphones," in 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2013, pp. 142–149.

[34] G. Canfora, E. Medvet, F. Mercaldo, and C. A. Visaggio, "Acquiring and Analyzing App Metrics for Effective Mobile Malware Detection," in Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics - IWSPA '16, 2016, pp. 50–57.

[35] B. Dixon, S. Mishra, and J. Pepin, "Time and Location Power Based Malicious Code Detection Techniques for Smartphones," in 2014 IEEE 13th International Symposium on Network Computing and Applications, 2014, pp. 261–268.

[36] "360 Security - Antivirus Boost - Android Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.qihoo.security. [Accessed: 16-May-2017].

[37] "Avira Antivirus Security - Android Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.avira.android. [Accessed: 16-May-2017].

[38] "Mobile Security & Antivirus - Android Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.avast.android.mobilesecurity. [Accessed: 16-May-2017].