A Comparison of Svm With Deep Learning Models for Large-Scale Intents Analysis

Toqeer Ali[1], Salman Jan[2], Safiullah Faizullah[3], Shahrulniza Musa[4]

[1] Islamic University of Madinah Madinah, Saudi Arabia
[2]University of Kuala Lumpur
[3]Islamic University of Madinah
[4]University Kuala Lumpur

Abstract
Android has been effectively adopted as an open source operating system over the smart devices since it offers customers a wide range of applications. The statistics regarding number of active applications in Google Play Store show overwhelming increase. Until December 2017, the number of available applications in the Google Play Store was 3.5 million while 50.6 million number of active applications are predicted by 2020. However, there are reports of intruded applications which violates user's privacy. It is essential to devise effective techniques to analyze and detect threats. to ensure integrity of data and applications, security experts presented various approaches including use sequences of permissions required by applications similarly system calls generated by applications are measured. This study proposes to consider intents initiated by applications as a parameter to verify malignant behavior of applications. To meet the purpose, a dataset containing 60,000 applications is generated which includes 20,000 malicious while 40,000 benign applications. The dataset is utilized to train proposed deep machine learning models including SVM and Generative Adversarial Networks (GANs). The results show reasonable malicious detection rate using intents on GANs. We believe that the proposed model is appropriate solution for ensuring security of Android applications.

Key words:
Smartphone Security, Android intents based analysis, intrusion detection, dynamic behavior analysis

1. Introduction

Android operating system is developed by google based on linux kernel for mostly smart phones. The smartphone market has raised considerably and android operating system has become de-facto OS for managing resources over these devices. Currently, android has become a market leader in smartphone operating systems [1]. The ultimate vast adoption of android has attracted the attention of malware developers. Experts estimated $600 billion as a global cybercrime cost in 2018. There are mobile malicious campaigns which generate millions of dollars revenues. Mobile malware has evolved at a high pace. There is ad click frauds, banking trojans, and many other malware continuously targeting mobile users. Thousands of malignant applications have been found on the Google Play Store, as legitimate apps. The malignant applications carry a recent detected malware called Dresscode while infiltrates networks and to hide security sensitive information. Moreover, it adds Botnet to infected devices to carry out denial of service (DDoS) attacks and to take part in spam email campaigns.

If a device infected with Dresscode meets a network where the router has a weak password, it can crack the password and then infect other devices on the network, including IoT connected home devices. [2]. Dresscode portrait itself as layout themes, utility apps, games etc. Malware researchers have witnessed a continuous increasing growth in android malware from 2012 until 2018. They have found that in Google play store, the perceived malignant applications are more than 130,000 among which no less than 40 are zero-day malware and roughly 35, 000 go undetected by a large portion of the malware detection run by VirusTotal [3]. In 2012, the Eurograbber assault took an expected e36 million in Europe [4]. Android owners have been warned that as many as 21.1 million devices may have been infected by malware from apps that were on the Google Play store. There are almost 30000 new applications including third party applications which are added to google play store. The rapid development of applications has make it difficult for the researchers to verify the health of all applications added to google play store. Google introduced Bouncer [5] to automate antivirus systems for Google's app store. But it has been observed that it can be bypassed and has low detection rate. To ensure that platforms are protected against the malware, researchers proposed solutions which are classified into static malware analysis [6], [7] dynamic analysis [8], [1]. Static analysis does not essentially execute programs to know they are malicious or not. Theses involves dissecting the application and reverse engineering it to observe its functions and to identify malicious code. On the other hand, dynamic malware analysis involves running the application in a closed environment like a sandbox wherein system calls initiated by applications are recorded and analyzed to determine the health of applications. The provided solutions are however, not able to deal with zero-day malware as the existing solutions are based on signatures, while in the absence of these signatures, the newly born malware go undetermined. Similarly, many

Manuscript received July 5, 2018
Manuscript revised July 20, 2018

machine learning techniques are used to classify benign and malicious applications on Android plat- form. For example, DREBIN [9] one of the successful work done in the recent past to classify the behavior of application on certain matrix, such as, permissions required by an application, API calls between the applications and middleware etc. They gathered a large dataset of 52 GB of behavior of 16 million benign applications' behavior and around 4,000 malware samples. However, our work is different from DREBIN in certain ways i.e., we are considering the dynamic behavior of the applications to classify intents while the DREBIN conducts static analysis of applications. In addition, we considered large-scale classification models that work on very large datasets. In contrast DREBIN worked on traditional classification models, such as, SVMs which cannot be scaled to very large datasets. TaintDroid [10] is another quite famous work done on behavior tracking at various levels of the Android software stack. TaintDroid use machine learning model to identify the bad and good behavior. TaintDroid works on out-of-the-box analysis technique to measure the run-time behavior of an app. However, one of our concerns regarding this kind of approach is its feasibility in the new Android version. Because of the many architectural changes in Android, such as its permission models, Delvik replacement with Android Run Time(ARM) etc., this technique is no longer workable. Similarly, this work is classifying the behavior on traditional machine learning model. We propose to utilize android intents as a parameter to determine the trustworthiness of applications. Applications during execution generate intents to perform various actions. The intents generated by benign and malicious applications constitutes classes and can be recorded in log files to train machine learning deep models on both classes of intents. The intents from applications can be collected through running application in an isolated environment e.g. a sandboxed environment which helps in following the strides of vindictive applications. We utilized Intents as a tool for our model. We extricated particular features amid the execution of an application that can foresee the reliability of application. Rest of paper is organized as follows. Section II explains architectural details of Android OS including system libraries, android runtime, application framework, applications, and android intents in general. provide background related to this paper. The target architecture, its evaluation and implementation details are elaborated in section 6.

2. Architectural Details of Android Operating System

This section provides introduction to Android's framework and its components as presented in Figure 1.

2.1 System Libraries

To perform various tasks in Android, system libraries are utilized that are indeed low-level codes written in either C or C ++. These define functionalities for the tasks. The same are presented to Android runtime (ART) and application framework through the use library Application Programing Interfaces. Direct access to native libraries is provided through Java Native Interface (JNI) bridge.

2.2 Android Runtime

All applications and daemons are executed using Android Runtime. The Dalwik VM and ART were specifically developed for the Android wherein the ART runs Dalvik executables and bytecode specification. In addition, it isolates processes from each one.

2.3 Application Framework

It adds to runtime environment in which android applications can be managed and executed. It can be considered as middleware. The key capacity of application system is to give a level one foundation to applications as various Android classes. Parts of this layer are for the most part executed as administrations or daemons that keep running out of sight. Some of real parts are depicted underneath. 1) Activity Manager: The activity manager behaves as daemon to watch active applications. It supervises services and manages them accordingly. In case of out of memory or unresponsive applications, the in-execution processes are killed by the activity manager. 2) Content Providers: Various applications require access to shared data. The purpose of content providers is to provide authorized access to contents required to applications e.g. the contact list of individuals is stored in content providers and are presented to applications upon request. 3) Telephony Manager: Telephony Manager manages over- all events related to telephone. Keeps International Mobile Equipment Identity (IMEI) number, a unique 15-digit code number possessed by all smart phones. similarly, it manages telephone calls made or received out of device. 4) Location Manager: Location manager gets device location through global positioning systems sensors. Consequently, applications get updates based on geographical location.

2.4 Android Applications

The device features are utilized through applications which adds features users are interested in. There are built in applications in the device which includes a browser, calculator, dialer Gmail and many others. while users may install third party applications in the device to increase the functionality of the device and to carry out functions which are or not offered by built in applications. Examples

of such applications include MS office, apps for video players, Sygic and many others.

2.5 Android's Intents

To put basically, intents are demand from applications for playing out a specific activity. Google characterizes it as; an Intent is theoretical depiction of an activity to be performed. It can be utilized to dispatch an action and to send it to any intrigued receiver part, and to begin service or tie any service for communication with services running in the background. Intents facilitates in carrying out late runtime binding among the parts of various applications. It launches activities, carries out communication among inter-activities and indeed serve as glue between them [11]. An intent is generated by the activity that intends to communicate with some other activity.

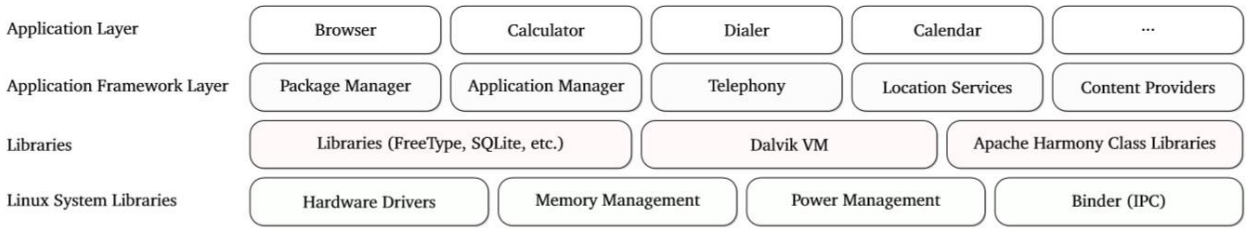

Fig. 1 Android Architecture

In that case, the intent encapsulates information which the other activity requires Intent Resolver or Intent Filter is utilized for resolving intents. The resolver searches up for the applications that have been enrolled before to deal with the particular intent upon a request is made by applications. Any application willing to serve an intent describes its willingness using the <IntentFilter> tag in the manifest le.

2.6 Intent Filter

An intent filter is an applications willingness to serve an intent. Intent filters are associated with individual components of applications. An intent-filter association function Af: C ➔2F maps each component of an application to a set of intent filters where F is the set of intent filters and I $\subseteq$ F. If a component c $\in$ C has an intent filter f, we write f $\in$ Af(c).

The flickr service exposes the action string edu.apex.android.intents.fks intent in its intent filter. In MS word program, the documents do possess hyperlinks to various other sources. Upon clicking on the hyperlink, a request for that resources is generated in the form of intents which is later on broad-casted. The Intent filter captures the same and handles it through appropriate application. The suitability of application for handling intent is checked in registry files wherein all applications for handling intents are listed. In the case, we browser gets started.

3. Malware

Malware is a malicious application that performs actions without the consent of users on the owner's devices. Malware writers uses this phenomenon to steal owners important in- formation and utilize resources and to gain monetary benefits. Due to these facts, the number of malware writers are increasing day by day [12]. For readers clarity, we describe various types of malware in the following subsection.

3.1 Types of malware

There are two Android malware types that traps mobile clients and intrude applications over android devices. Both types are provided hereunder. 1) Fake Installers, SMS Trojans, Spywares and Botnets: There is a major part of applications over the web which are SMS trojans or fake installers. These types of malware purport themselves are benign and bamboozle applications and the devices and thus behave abnormally [13]

Spywares and botnets hides data over the devices and communicates the same to remote computers. The latest types of these malware do set specific port for listening commands from the remote attacker and provides access to resources which are sensitive in nature. The attacker thus misuses the resources through performing various actions through these ports enabling the devices compromised.

3.2 Malware Distribution

Malware distributions take place in various ways. It can occur through downloading pirating applications specifically the third-party applications that contain trojans. If the devices are not managed properly or if the required updates are set to off, the devices become more prone to be attacked [14].

4. Machine Learning Vs Deep Learning

A set of automated techniques and methods for obtaining knowledge in the form of patterns and then using those patterns for processing future data including prediction . Machine learning is classified into to types i.e. supervised and unsupervised [15]. Unsupervised Learning lacks target field in its data i.e. no training data is provided [16]. The data is investigated for possible detection of structures for better representation of the same data. [15]. The data is investigated for possible detection of structures for better representation of the same data. There are many unsupervised models including GANs, Deep Generative Adversarial Networks (GAN) are utilized for identifying benign and intruded behaviors. GANs are proven effective in the fields of computer vision and rapid use in many domains. The strengths of GANs as reported in the literature have made many deep learning scientists to explore the utilization of GANs in their respective fields. GANs are proved best in recognizing patterns of similar appearances. The results of utilizing GANs in identifying benign and malicious sequences is reported in the paper. Beside utilization of unsupervised learning, we used super- vised learning in this study which utilizes datasets consisting both features and target objects [17]. This study utilized logistic regression and support vector machines [18], [19] for regression and classification. In [Ref to conference], we were having limited dataset and the machines used were not computationally efficient. Due to the fact, machine learning traditional algorithms were employed. Since the size of dataset has now been increased, we made use of GANs as well. Generative Adversarial Net- works require computational efficient machines so for the purpose GPU machines were opted. The results demonstrate the difference between traditional machine learning and deep learning output in correctly recognizing patterns in the domain of malware analysis.

5. Target Framework

Our proposed solution is based on intents that is a core of Android application interaction. Formally, an intent is 4 tuple entity represented through (α, β, γ, σ) such that α denote the action that is carried out, β represent data part, γ presents the category while σ : name val represents function mapping names to corresponding values. The set of intents is denoted as I. We will be using intents and action interchangeably throughout the paper. An action can be called either with activity, service or broadcast. However, we are not considering service based actions because this research solely works on applications. A service can generate intents in the background and that is basically goes out of the scope of this research. There are times when service can come with a client/server interface, in that case, we will consider the actions related to services as an Android application.

5.1 Behavior of Android Application

Application's behavior is indeed the sequences of in- tents/actions that is generated by a particular application. Intents are of two types one is called implicit and the other is explicit. Figure 2 explains an example of implicit intent. Application 1 is generating an action that in turn creates implicit intent. The action is captured by Android framework that tries assign the action to appropriate app that may further create another intent.

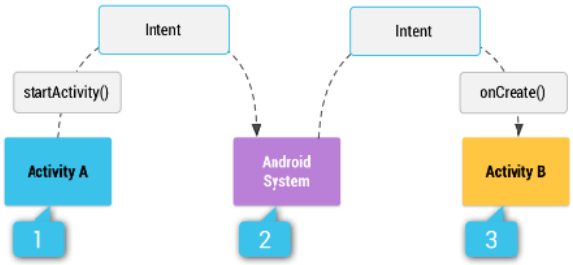

Fig. 2 Behavior capture in Android Application (Implicit Intent)

5.2 Scope of the Research

Many approaches are presented in literature for An- droid malware analysis. Researchers have provided solutions that can hardly be distinguished, following brief deviate our studies from earlier studies carried out in this domain. Researchers provide static analysis, in which the applications are not executed to analyze behavior. Static analysis includes the DREBIN, a famous work that considers permissions required by applications and API calls made between applications [20], [21], [7]
Dynamic Analysis approach, which requires the application to run to analyze the behavior, includes TaintDroid [10]. Taindroid identifies the bad and good behavior similarly, SCANDROID [22] is an incremental approach to analyze behaviors over devices. The problem with these techniques is that they use traditional Machine Learning model that cannot handle big data and have scalability issues. They do face problem network activities as well Similarly, some presented solutions that considers

system's information and flow monitoring. Few of them monitors traces as left by applications that goes in many levels of the Android stacks. That includes, method-level, message-level, variable- level, and file-level trace tracking. This research provides solution in classification of behavior into good and bad behaviors based on Android Intents. It is different from the previous studies in a sense that earlier studies worked at either kernel level to capture system calls or reverse engineering was used to obtain application's behavior. Moreover, models of Deep Learning including Generative Adversarial Networks are also trained on the benign and intruded behaviors of applications as recorded in the CSV files. It is observed during evaluation of the framework that these models outperform rest of machine learning models in terms of efficient classification of good and bad behavior of applications.

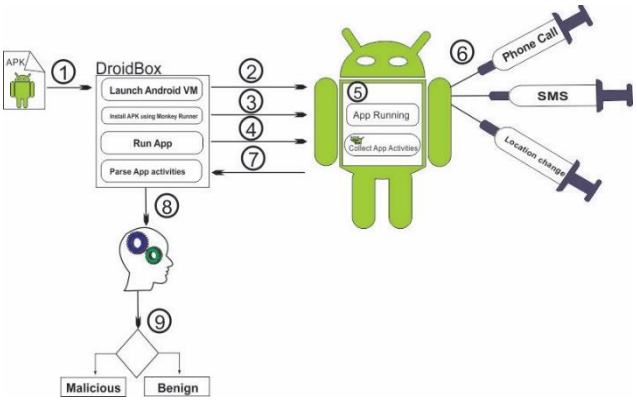

Fig. 3 Intent Based Monitoring System Design

Our proposed framework is indeed an extension to the work published presented in conference [45]. which make use of machine learning techniques like SVM. It has been observed through employing SVM in the previous research work that it works well with dataset of intents that was indeed short in size. Since the dataset was not covering data about large number of applications, we continuously worked on our dataset and enhanced its size to incorporate large number of applications and their associated behaviors. However, with the increase in size of the dataset, the traditional machine learning models are no longer applicable. The idea of Neural networks presented back in 1990's was not very successful at that time due to limited resources and less computational power [23]. However, the emergence of supercomputers and huge processing power, the deep learning concepts resurged. The GPUs can efficiently process big data that was not possible earlier. Today, even the google and other famous organizations are utilizing these models. The deep learning latest models also termed as deep neural architectures includes Fully-Connected Neural-Networks (FCNN) [24], CNNs [25], DBNs [26], DAE [27], RNNs [28], LSTM [29] are best to work with images. These models are trained for analyzing applications behaviors too. These models are efficient to detect malware with low false positive, but they are best to work with static analysis and are not best with dynamic analysis. For our solution, we prefer to use Generative Adversarial Network (GAN) [30], [31], [32], which are unsupervised machine learning, and is best working for big data. It is reported during review of literature that GANs work well when we don't have labeled data. Since the malware samples are increasing drastically likewise, there are increasing number of android user's behaviors and applications that constitute huge volumes of data to be analyzed and mostly these are unlabeled. The traditional machine learning algorithms are not capable to analyze these volumes of data. Since, our large of dataset does not have labels, we opted GANs and surprisingly, the results are quite adequate and show the strengths of GANs over the previous machine learning models. The results section demonstrates both supervised and unsupervised machine learning and their results on the dataset of applications. In the past many parameters of applications are considered for analysis of malware. Security measured were taken on program levels, functions level, security sensitive functions were analyzed, others opted data structures that hold information regarding processes. Many researchers utilized system calls sequences on desktop and smart devices as well as behavioral analysis and permissions used over desktop and smart devices [33], [34], [35], [36]. Programs and functions exhibit various behaviors during execution which constitute behavioral profile on windows, Unix and Linux Operating Systems. Many researches have considered these parameters as notion of analysis [37], [38], [39], [40]. Since Android has its own architecture as depicted in figure 1 and the same architecture frequently adopts changes consequently, the dex format, permission model and the runtime environment are changed, however, the intents, as generated by applications remains the same. The system calls sequences that are dealt at comparatively very lower level in the Android Architecture has not remained good choice for analysis of malware over android device. The notion of selection of Android intents as behavioral analysis is opted as its dealt on top layer of Android architecture i.e. the application layer and even if the architecture is changed again, the intents as generated by the applications shall remain intact. As an extension in size of our database, we downloaded 60, 000 applications, containing 40000 benign and 20000 malicious applications, from Google Play Store and various other sources used for malicious applications. To run these programs and to record their possible intents, we downloaded.

Android Source Code, the Oreo Version 8.01 and build the same on a machine with reasonable memory and processing speed. Hooks were placed in Android OS to

capture sequences of intents as generated by various applications. The generated sequences of intents from the good and bad applications were recorded in log files. The dataset constituted 6 Gb of data in size. These intents dataset were preprocessed for training GANs and other machine learning models. Moreover, the dataset was split into a ratio of 60 percent for training set and 40 percent for testing set. On a standard GPU, the different traditional machine learning algorithms along with deep learning models were employed. we observed that GANs outperformed rest of other machine learning algorithms as our earlier machine learning algorithms were not generating up to that extension. The results section elaborates how machine learning traditional models and the deep generative adversarial networks efficiently classifies the good and bad behavior of applications.

6. Evaluations

In this section, we report results of employing various machine learning models on our dataset to affirm which model is more accurate in terms of correctly classifying behavior of applications. We consider a number of model evaluation tools including confusion matrix, classification accuracy, F1 score and ROC curve. The results of employing traditional machine learning and deep generative models are provided in this section. 1) Confusion Matrix: The correct and incorrect classification of applications is reflected through confusion matrix . Our confusion matrix reports that GANs are better than others. The details are listed in table I.

Table I: Confusion Matrix

Algo	TP	TN	FP	FN
SVM Poly	331	16	109	202
SVM Linear	326	21	40	271
Log Reg	325	22	38	273
SVM RBF	321	26	50	261
GAN	335	14	112	207

Where
True Positive (TP): Model predicts true while its True
False Positive (FP): Model Predicts true while its False
False Negative (FN): Model Predicts Negative and its False
True Negative (TN): Model Predicts Negative and it is True
2) Accuracy in classification: It presents how correct the model classifies given input. It provides percentage of correct classification [41]. It is calculated through the following expression:

Model's Accuracy = $\frac{TP+TN}{TP+TN+FN+FP}$

The classification accuracy is presented in the beneath table II.

Table II: Classification Accuracy

Logis Reg:	0.908814589666
SVM RBF:	0.884498480243
SVM Linear:	0.907294832827
SVM Poly:	0.810030395137
GAN:	0.947864987621

6.1 F1 Score

The F1-score is listed in table III

Table III: F1 Score

Logis Reg:	0.91549296	0.9009901
SVM RBF:	0.914446	0.89883914
SVM Linear:	0.914446	0.89883914
SVM Poly:	0.841169	0.7637051
GAN:	0.837799	0.7587562

6.2 ROC Curve

Receiver Operating Characteristic (ROC) curve is the graphical representation of the performance of any binary machine learning classification algorithm on multiple thresholds [44]. Below are the ROC curves of our model.

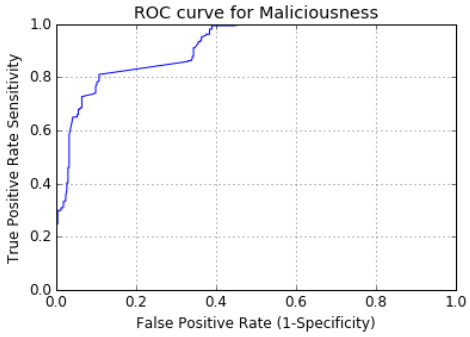

Fig. 4 ROC Curve of SVM Poly

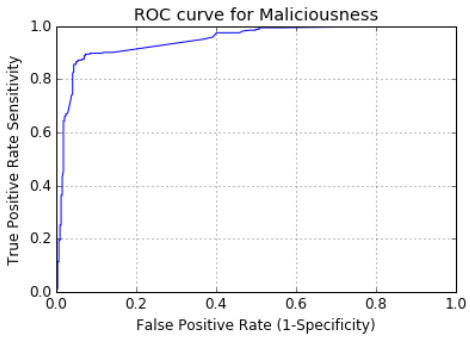

Fig. 5 ROC Curve of SVM Linear

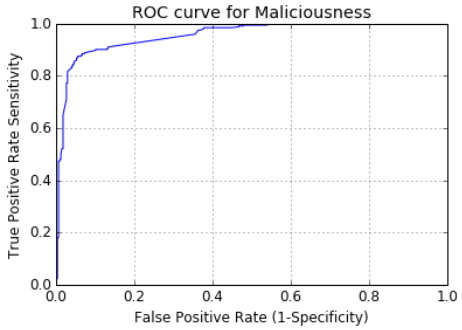

Fig. 6 ROC Curve of Logistic Regression

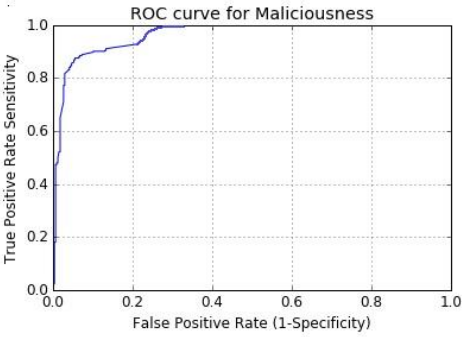

Fig. 7 ROC Curve of GAN

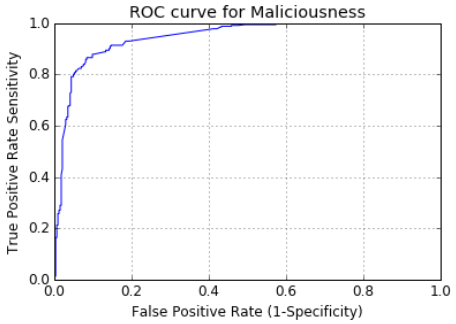

Fig. 8 ROC Curve of SVM RBF

6.3 AUC Score

After employing various models, the following AUC is recorded:

Table IV: Auc Score

Logis Reg:	0.954497437846
SVM RBF:	0.945754607708
SVM Linear:	0.945416384814
SVM Poly:	0.914503738985
GAN:	0.92679213678

6.4 Histogram

The histograms after implementation of various models are provided below:

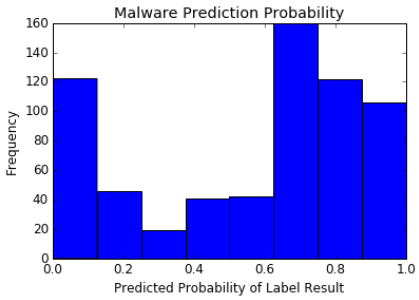

Fig. 9 Histogram of SVM Poly

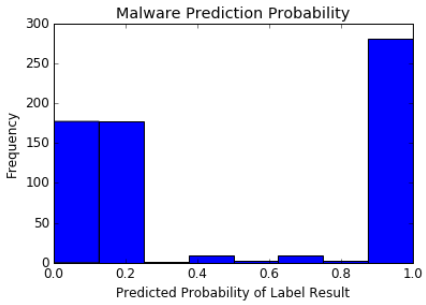

Fig. 10 Histogram of SVM Linear

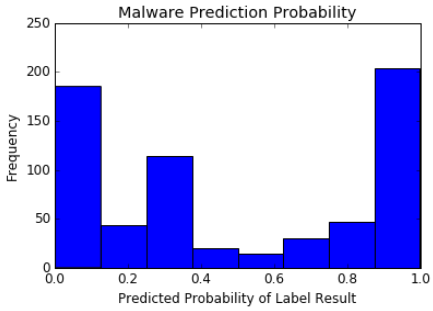

Fig. 11 Histogram of Logistic Regression

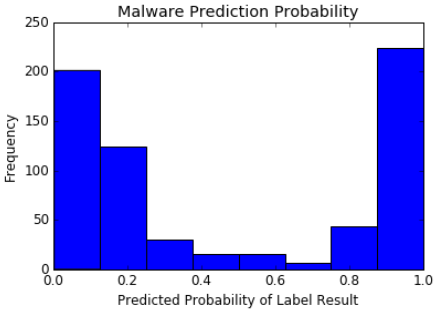

Fig. 12 Histogram of SVM RBF

7. Conclusion

The study provides a novel approach for detection of malware for Android operating systems based on intent based dynamic behavior capture. The presented approach successfully distinguishes between benign and intruded intent sequences. The fact is ascertained through experimental results which confirms the effectiveness of framework in detection of malware. Moreover, the documented results suggest that selection of Generative Adversarial Networks produces more better results. GANs generated 0.94 accuracy which are better than those of SVM and logistic regression for learning features and patterns within android intents for classification of behaviors into benign and malicious of various applications.

References

[1] 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.

[2] S. provides security products, m. solutions to protect small, and m. enterprise businesses from advanced threats, "Symantec employee," https://malaysia.norton.com/internetsecurity-emerging-threats-hundreds-of-android-apps-containing-dresscode-malware-hiding-in-google-play-store. html.

[3] V. is a free service that analyzes suspicious files, URLs, and facilitates the quick detection of viruses, "An online antivirus," https://https://www. virustotal.com.

[4] E. Kalige, D. Burkey, and I. Director, "A case study of eurograbber: How 36 million euros was stolen via malware," Versafe (White paper), 2012.

[5] J. Oberheide and C. Miller, "Dissecting the android bouncer," Summer- Con2012, New York, 2012.

[6] Y. Feng, S. Anand, I. Dillig, and A. Aiken, "Apposcopy: Semantics- based detection of android malware through static analysis," in Pro- ceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2014, pp. 576–587.

[7] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, "Droidmat: Android malware detection through manifest and api calls tracing," in Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on. IEEE, 2012, pp. 62–69.

[8] L.-K. Yan and H. Yin, "Droidscope: Seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis." in USENIX security symposium, 2012, pp. 569–584.

[9] D. Arp, M. Spreitzenbarth, M. H¨ubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket," in Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS), 2014.

[10] 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 Transactions on Computer Systems (TOCS), vol. 32, no. 2, p. 5, 2014.

[11] G. Inc., "What is android," 2017.

[12] i. a. i. G DATA Software AG, with its head office in Bochum and quickly expanding software house focusing on antivirus security solutions., "Gdata mobile malware report," https://public.gdatasoftware.com/Presse/Publikationen/Malware Reports/G DATA MobileMWR Q1 2015 US.pdf.

[13] V. Vanitha, "Android malware analysis: A survey," 2017.

[14] C. V. ANAND, S. NAWAZ, and R. RAMACHANDRA, "The spread of malicious software in large scale networks."

[15] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to mcmc for machine learning," Machine learning, vol. 50, no. 1-2, pp. 5–43, 2003.

[16] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," in The elements of statistical learning. Springer, 2009, pp. 485–585.

[17] C. E. Rasmussen, "Gaussian processes for machine learning," 2006.

[18] S. Menard, Applied logistic regression analysis. Sage, 2002, no. 106.

[19] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," Journal of machine learning research, vol. 2, no. Nov, pp. 45–66, 2001.

[20] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011, pp. 627–638.

[21] M. Nauman, T. A. Tanveer, S. Khan, and T. A. Syed, "Deep neural archi- tectures for large scale android malware analysis," Cluster Computing, pp. 1–20, 2017.

[22] A. P. Fuchs, A. Chaudhuri, and J. S. Foster, "Scandroid: Automated security certification of android," Tech. Rep., 2009.

[23] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jes´us, Neural network design. Pws Pub. Boston, 1996, vol. 20.

[24] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural networks, vol. 2, no. 5, pp. 359–366, 1989.

[25] Y. LeCun, Y. Bengio et al., "Convolutional networks for images, speech, and time series," The handbook of brain theory and neural networks, vol. 3361, no. 10, p. 1995, 1995.

[26] R. Salakhutdinov and I. Murray, "On the quantitative analysis of deep belief networks," in Proceedings of the 25th international conference on Machine learning. ACM, 2008, pp. 872–879.

[27] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," science, vol. 313, no. 5786, pp. 504–507, 2006.

[28] Z. Yang, Z. Hu, Y. Deng, C. Dyer, and A. Smola, "Neural ma- chine translation with recurrent attention modeling," arXiv preprint arXiv:1607.05108, 2016.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing systems, 2014, pp. 2672– 2680.

[31] Z. C. Lipton, "Deep Convolutional Generative Adversarial Networks," available at:

https://github.com/zackchase/mxnet-the-straight-dope/blob/ master/chapter14 generative-adversarial-networks/dcgan.ipynb.

[32] J. Burns, “Exploratory Android Surgery,” in Black Hat Technical Se- curity Conference USA, 2009, available at: https://www.blackhat.com/ html/bh-usa-09/bh-usa-09-archives.html.

[33] R. Tian, R. Islam, L. Batten, and S. Versteeg, “Differentiating malware from cleanware using behavioural analysis,” in Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on. IEEE, 2010, pp. 23–30.

[34] T. Ali, M. Nauman, and X. Zhang, On Leveraging Stochastic Models for Remote Attestation. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 290–301. [Online]. Available: http://dx.doi.org/ 10.1007/978-3-642-25283-9 19

[35] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, “Opcode se- quences as representation of executables for data-mining-based unknown malware detection,” Information Sciences, vol. 231, pp. 64–82, 2013.

[36] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, “Classification of malware based on integrated static and dynamic features,” Journal of Network and Computer Applications, vol. 36, no. 2, pp. 646–656, 2013.

[37] K. A. Asmitha and P. Vinod, “A machine learning approach for linux malware detection,” in 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Feb 2014, pp. 825–830.

[38] M. Nauman, N. Azam, and J. Yao, “A three-way decision making approach to malware analysis using probabilistic rough sets,” Information Sciences, vol. 374, pp. 193 – 209, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025516308969

[39] Christopher Olah, “Understanding LSTM Networks,” http://colah.github. io/posts/2015-08-Understanding-LSTMs/, 2015.

[40] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, “Deep learning for classification of malware system call sequences,” in Australasian Joint Conference on Artificial Intelligence. Springer, 2016, pp. 137–149.

[41] P. Poˇ r´ızka, J. Klus, A. Hrdliˇcka, J. Vr´abel, P. ˇSkarkov´a, D. Prochazka, J. Novotn`y, K. Novotn`y, and J. Kaiser, “Impact of laser-induced break- down spectroscopy data normalization on multivariate classification accuracy,” Journal of Analytical Atomic Spectrometry, 2017.

[42] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg, “Adaptive name matching in information integration,” IEEE Intelligent Systems, vol. 18, no. 5, pp. 16–23, 2003.

[43] M. Buckland and F. Gey, “The relationship between recall and preci- sion,” Journal of the American society for information science, vol. 45, no. 1, p. 12, 1994.

[44] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in Proceedings of the 23rd international conference on Machine learning. ACM, 2006, pp. 233–240.

[45] M. W. Afridi, T. Ali, T. Alghamdi, T. Ali, and M. Yasar, “Android application behavioral analysis through intent monitoring,” in Digital Forensic and Security (ISDFS), 2018 6th International Symposium on. IEEE, 2018, pp. 1–8.