Towards an Autonomous Host-based Intrusion Detection System for Android Mobile Devices

José Ribeiro[1, 2], Georgios Mantas[1, 3], Firooz B. Saghezchi[1], Jonathan Rodriguez[1], Simon J. Shepherd[2], and Raed A. Abd-Alhameed[2]

[1] Instituto de Telecomunicações, Aveiro, Portugal
[2] Engineering & Informatics, University of Bradford, UK
[3] Faculty of Engineering and Science, University of Greenwich, UK
{jcarlosvgr, gimantas, firooz, jonathan}@av.it.pt
{s.j.shepherd, R.A.A.Abd}@bradford.ac.uk

Abstract. In the 5G era, mobile devices are expected to play a pivotal role in our daily life. They will provide a wide range of appealing features to enable users to access a rich set of high quality personalized services. However, at the same time, mobile devices (e.g., smartphones) will be one of the most attractive targets for future attackers in the upcoming 5G communications systems. Therefore, security mechanisms such as mobile Intrusion Detection Systems (IDSs) are essential to protect mobile devices from a plethora of known and unknown security breaches and to ensure user privacy. However, despite the fact that a lot of research effort has been placed on IDSs for mobile devices during the last decade, autonomous host-based IDS solutions for 5G mobile devices are still required to protect them in a more efficient and effective manner. Towards this direction, we propose an autonomous host-based IDS for Android mobile devices applying Machine Learning (ML) methods to inspect different features representing how the device's resources (e.g., CPU, memory, etc.) are being used. The simulation results demonstrate a promising detection accuracy of above 85%, reaching up to 99.99%.

Keywords: Mobile Intrusion Detection System, Android, Security, 5G Communications, Machine Learning, Malware Detection, Host-based IDS.

1 Introduction

Nowadays, the growing popularity of mobile devices (e.g., smartphones) along with the increased data transmission capabilities of future 5G networks, the wide adoption of open operating systems and the fact that mobile devices support a large variety of connectivity options (e.g.,3G/4G, Bluetooth) are factors that render the mobile devices a prime target for cyber-criminals. Apart from the traditional SMS/MMS-based Denial of Service (DoS) attacks, the future mobile devices will also be exposed to more sophisticated attacks originated from mobile malwares (e.g., viruses) that target both the device itself and the 5G network. Moreover, the open operating systems will allow users to install applications on their devices, not only from trusted, but also

from untrusted sources (i.e., third-party markets). Consequently, mobile malwares can be included in applications looking like innocent free software packages (e.g., games), that can be downloaded and installed on users' mobile devices (e.g., smartphones), exposing them to many threats. In particular, mobile malwares can be designed to enable attackers to exploit the stored personal data on the device or to launch attacks (e.g., DoS attacks) against other entities, such as other user mobile devices, the mobile access networks, the mobile operator's core network and other external networks connected to the mobile core network [1], [2], [3], [4]. Thus, security mechanisms such as mobile Intrusion Detection Systems (IDSs) are essential to protect mobile devices from many known and unknown security threats and to ensure user privacy.

During the last decade, a lot of research effort has been placed on IDSs for Android mobile devices, as Android is the most popular mobile device OS in the market, so it remains the main target for mobile threat actors [5], [6], [7]. Additionally, the emergence of cloud computing has led a lot of IDS solutions to be cloud-based, since they take advantage of the effectiveness that the centralized data collection and processing provide [8], [9], [10]. However, this trend is characterized by two main constrains. First, it needs a continuous connectivity of the mobile device (e.g., smartphone) to a remote central server. Although 5G aims to provide ubiquitous coverage and full connectivity, it is yet possible, even in the 5G era, for the mobile devices to suffer from the channel fading or the network outage. In addition, the second constraint is the risk of sensitive information leakage that can occur (e.g., via IDS alerts sent out from the device) and lead to compromising user privacy. Hence, it is fundamental to investigate the design and development of more autonomous host-based IDSs to protect future Android mobile devices from a plethora of known and unknown security threats and to ensure user privacy in a more efficient and effective manner.

Therefore, in this paper, we propose an autonomous host-based IDS for Android mobile devices (e.g., smartphones) that overcomes the limitation of continuous connectivity to a central server and addresses the risk of data leakage due to communication of the IDS with the remote central server. The proposed IDS is based on dynamic analysis of device behaviour for detecting suspicious behaviour on Android mobile devices. In other words, the detection takes place through analysis of deviations in device's behaviour which is described through a vector of features. The proposed IDS continuously monitors a specific set of features of the mobile device at the device level to define its run-time behaviour and apply Machine Learning (ML) algorithms to classify it as benign or malicious. It is worthwhile to mention that the monitoring process (i.e., real-time data acquisition) does not require root access, and thus the proposed IDS is able to run directly on un-rooted Android devices. In particular, the proposed IDS was implemented as a regular Android application running on an un-rooted Samsung Galaxy (J1 model: SM-J100H) smartphone running Android KitKat (version 4.4.4). Finally, to the best of our knowledge, publicly available datasets including benign and abnormal behaviour of Android mobile devices do not exist. Thus, in order to evaluate the proposed IDS, we generated our own two datasets: a) benign activity dataset; and b) abnormal activity dataset. The evaluation results demonstrate that the proposed IDS has a low impact on the data collection process in terms of CPU consumption, memory and battery usage.

Following the introduction, this paper is organized as follows. In section 2, we describe the architecture of our proposed IDS and its different components. In Section 3, we introduce different features that we use for building ML models. In Section 4, we discuss how we construct our own datasets and present the evaluation results. Finally, Section 5 concludes the paper and provides some hints for the future work.

2 Proposed Host-based IDS for Android Mobile Devices

The proposed Host-based Intrusion Detection System (HIDS) employs ML algorithms including One Rule (OneR), Decision Tree (DT), Naïve Bayes (NB), Bayesian Network (BN), Logistic Regression (LR), Support Vector Machine (SVM) or k-Nearest Neighbour (k-NN) to identify suspicious behaviour on the Android device by analysing the system log files and then it calculates the probability of intrusion. To this end, we identified the features that effectively characterize the impact of mobile malware on the Android device and maximize the effectiveness of ML techniques for detection of suspicious activity. These features are monitored in real-time by the IDS in order to collect the required data for suspicious behaviour detection.

2.1 Overall Architecture of the Proposed Host-based IDS

The architecture of our proposed host-based IDS is composed of the following components as shown in Fig. 1: a) real-time data acquisition, b) real-time dataset generation, c) feature normalization, d) classifier, e) intrusion probability assessment, and f) alert manager. In the following, we briefly explain these components.

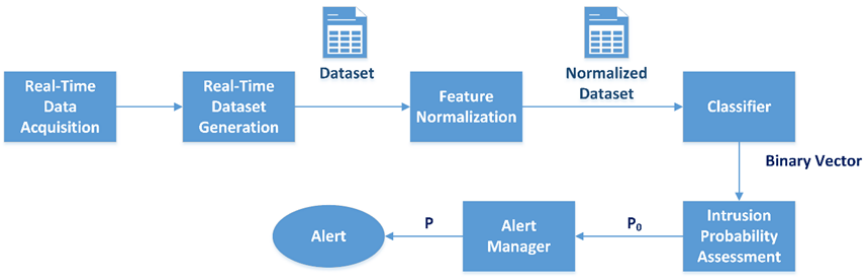

Fig. 1. Architecture of the proposed HIDS

2.2 Real-Time Data Acquisition

The *Real-Time Data Acquisition* component is responsible for collecting real-time information about the following features: total CPU usage, memory consumption, outgoing/incoming network traffic, battery level/voltage/temperature, number or running processes/services, and a binary indicator representing whether the screen is on or off during a data acquisition period.

2.3 Real-Time Dataset Generation

The *Real-Time Dataset Generation* module is responsible for constructing the training and/or testing datasets in real-time. The collected real-time information is saved in csv (comma-separated values) files. Each file contains the data collected during a *data acquisition interval*, which can be adjusted from several minutes up to one hour (see Fig. 2). Each entry (row) represents a sample (training example) and each column represents a feature. Data collection can be performed periodically every hour, every two hours, or so during a day. The data collected during each *data acquisition period* is saved in a separate csv file.

2.4 Feature Normalisation

The *Feature Normalisation* component receives the raw data from the *Real-Time Dataset Generation* component and normalises it as follows: for each column (representing one feature), it first subtracts the mean value of the column from each element of the column and then divides the result by the standard deviation of the column. This operation is repeated for all columns and the output is again saved in a new csv file. That is, each column of the new csv file has mean 0 and standard deviation 1.

2.5 Classifier

The *Classifier* module makes use of ML algorithms, namely OneR, DT, NB, BN, LR, SVM (with the polynomial kernel with exponent equal to 1) or k-NN in order to classify each entry of the normalized dataset. It is worth mentioning that OneR is a classifier which simply has only one rule for classification; it checks the feature that yields the best classification performance. For DT algorithm, we consider at least ten objects per each leaf, and for k-NN, we consider k=1; that is, each new example is assigned to the class of its nearest neighbour example amongst all previously classified examples. Particularly, the output for each entry is classified as either benign (represented by the binary value 0) or malicious (represented by 1). Therefore, the output of the Classifier is a binary vector whose length is equal to the number of the entries in the normalized dataset. This binary vector is the input to the *Intrusion Probability Assessment* component.

2.6 Intrusion Probability Assessment

The *Intrusion Probability Assessment* calculates the probability of intrusion for a given *data acquisition period*. Denoting the output (binary) vector of the *Classifier* in Fig. 1 as $y \epsilon \mathbb{R}^{m \times 1}$, the probability of intrusion in data acquisition period k is calculated as follows:

$$P_0(k) = \frac{\sum_{i=1}^{m} y_i}{m} A \qquad (1)$$

where A denotes the accuracy of the Classifier, which is defined as follows:

$$A = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

where:

- TP (True Positives): the number of positive entries (malicious behaviour) that are correctly classified,
- TN (True Negatives): the number of negative entries (normal behaviour) that are correctly classified,
- FP (False Positives): the number of negative entries (normal behaviour) that are wrongly classified as positive (malicious behaviour), and
- FN (False Negatives): the number of positive entries (malicious behaviour) that are wrongly classified as negative (normal behaviour).

Furthermore, we define additional three metrics that later on in Subsection 4.2 will be used for evaluating the performance of the ML algorithms that we consider for the Classifier in Fig. 1, namely Precision, Recall and F-Measure, as follows.

Precision: the ratio of the total generated alerts by the IDS, either correct or false, that are really originated from malicious incidents:

$$P = \frac{TP}{TP+FP} \quad (3)$$

Recall: the ratio of the total positive incidents that are successfully detected by the IDS:

$$R = \frac{TP}{TP+FN} \quad (4)$$

F-Measure: a combination of precision and recall defined specifically as their harmonic mean.

$$F = 2\frac{P\times R}{P+R} \quad (5)$$

2.7 Alert Manager

The overall probability of intrusion given the probability of intrusion for the current and the past monitoring periods is calculated by the *Alert Manager* component. We assume that the incident of intrusion is independent from one monitoring period to another and calculate the overall probability of intrusion as follows:

$$P(k) = 1 - \prod_{i=1}^{\alpha-1}\big(1 - P_0(k-i)\big), \quad (6)$$

where α is the number of consecutive alerts that the alert manager receives up to k^{th} *data acquisition period*. For instance, if the *Alert Manager* receives three consecutive alerts and the probability of intrusion for each alert is 0.87, then the overall probability of intrusion would be $P = 0.998$. In case the overall probability exceeds a threshold, the HIDS sends an alert to the user (i.e., notification message).

3 Feature Extraction

To detect suspicious behaviour on Android mobile devices (e.g., smartphones), the proposed IDS needs to analyse different kinds of features. For this reason, the proposed IDS continuously monitors the following features of the mobile phone at the device level: the total CPU usage, memory consumption, outgoing/incoming network traffic, battery level/voltage/temperature, number or running processes/services, and a binary indicator representing whether the screen is on or off during each data acquisition period. The complete list of the monitored features is reported in Table 1.

Table 1. Monitored features for malware detection.

Feature	Description
Total CPU usage	Overall CPU consumption
Memory usage	Overall memory usage
Memory available	Mem Free + Cached
Memory Free	Memory not used
Cached	Memory used as cache
Total Rx bytes	Received bytes
Total packets Rx	Received packets
Total Tx bytes	Transmitted bytes
Total packets Tx	Transmitted packets
Batt Level	Battery level percentage
Batt Voltage	Battery voltage
Batt temp	Battery temperature (ºC)
Running Processes	Total number of running processes
Running Services	Total number of running services
Time Display On	Total seconds of the display is on.
Display On/Off	Display is: on = 1 ; off = 0

4 Evaluation

To evaluate the performance of the proposed IDS, we implemented it as a regular Android application on an un-rooted Samsung Galaxy (J1 model: SM-J100H) smartphone running Android KitKat (version 4.4.4). In particular, we used the Android Studio platform to develop the proposed IDS, as it contains specific tools for developing mobile Android applications [10]. However, to the best of our knowledge, publicly available datasets representing benign and abnormal behaviour of Android mobile devices do not exist. Thus, we generated our own two datasets: a) the benign activity dataset; and b) the abnormal activity dataset to evaluate the proposed IDS.

4.1 Dataset Generation

We defined the *data acquisition period*, *the data acquisition interval* and the *sampling period*, as illustrated in Fig. 2, for the purpose of dataset generation.

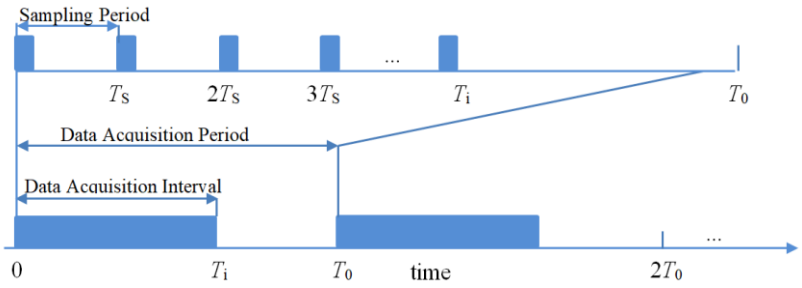

Fig. 2. Data acquisition period, data acquisition interval, and sampling period for dataset generation.

To create our datasets, we set these parameters as follows:

a) *Data Acquisition Period*, $T_0 = 1\ h$
b) *Data Acquisition Interval*, $T_i = 20\ min$
c) *Sampling Period*, $T_s = 2\ sec$

The process starts by collecting data from the device for the benign behaviour dataset. To generate the benign behaviour dataset, we run a game (Mind games) while listening an online radio station (radioonline.com.pt). Then, the device was infected with a malware and we run the same game as before while listening the online station in order to generate the malicious behaviour dataset. The process was repeated for each of the five malwares listed in Table 2; the table also provides additional information about the type of misbehaviour that each malware manifests. The device was cleaned after each operation so that only one malware was running at a time.

Table 2. Android Malwares used for testing the proposed IDS.

Malware	Type of misbehaviour	Package name
Adobe Flash Player	CPU consumption, Admin. rights, Activate Wifi, Fake Google store.	com.paranbijuv.aijuy
Adobe Flash Player	CPU consumption, Admin. rights, Activate Wifi, Fake Google store, lock the screen.	com.android.locker
Secrettalk_Device	Admin. rights, CPU consumption.	com.android.secrettalk
Google Installer	AndroidXbot, Admin. rights, CPU consumption.	org.luckybird.core
Radardroid2Map	Used to mine and generate bit coins.	com.ventel.android.radardroid2

The collected data is saved in csv files. Each file contains the data collected during a *data acquisition interval*. Each entry (row) represents a training example and each column represents a training feature. For the *data acquisition interval* which is equal to 20 min, the csv file contains 600 samples, as the sampling period is set to 2 sec. Thus, since the *data acquisition period* was equal to 1h, 24 csv files were created for the benign behaviour during one day. On the other hand, as we infected our smartphone with five different malwares, each at a time, 120 (24x5) csv files were created for the malicious behaviour during a day.

4.2 Evaluation Results

For performance evaluation, we construct two datasets using the data collected from both benign and infected versions of a mobile device discussed in the previous section. We refer to these new datasets as dataset 1 and dataset 2. Each dataset contains 12000 training examples, 6000 benign and 6000 malicious, uniformly and independently chosen from the collected benign and malicious data. In particular, the 6000 malicious examples in each dataset is uniformly and randomly selected from five malwares listed in Table 2. That is, each dataset contains 1200 examples from each malware.

Table 3. Evaluation results for 10-fold cross validation over training dataset 1.

Algorithm	Accuracy	Precision	Recall	F-Measure
OneR	0.9895	0.9913	0.9877	0.9895
DT	0.9992	0.9993	0.9990	0.9992
NB	0.9987	0.9973	1	0.9987
BN	0.9993	0.9987	1	0.9993
LR	0.9988	0.9992	0.9985	0.9988
SVM	0.9994	0.9993	0.9995	0.9994
k-NN	**0.9999**	**1**	**0.9998**	**0.9999**

Table 4. Evaluation results for training on dataset 1 and testing against dataset 2.

Algorithm	Accuracy	Precision	Recall	F-Measure
OneR	0.5563	0.5301	0.9893	0.6904
DT	0.5903	0.5496	0.9992	0.7092
NB	0.7152	0.6371	1	0.7783
BN	0.5483	0.5254	1	0.6889
LR	0.5608	0.5324	0.9983	0.6945
SVM	**0.8447**	**0.7632**	**0.9995**	**0.8655**
k-NN	0.8406	0.7582	1	0.8625

For numeric evaluation, we conduct two experiments. In the first experiment, we train and test over the same dataset, namely dataset 1, using 10-fold cross validation,

whereas in the second experiment, we train the algorithm using dataset 1 and test it on examples from dataset 2. The main rational behind the second experiment was to inspect the generalisation capability of the constructed ML model for the IDS. Table 3 summarises the results for the first experiment. As can be seen in the table, all algorithms show impressive performance, leading to over 99% accuracy, precisions, recall, and F-measure. This shows that the ML algorithm correctly classifies most of the training instances except few FPs or FNs. Furthermore, surprisingly, the simple k-NN algorithm yields the best performance. However, it is worth mentioning that unlike other learning algorithms where the training is the most computationally intensive part and the testing is just a simple calculation, the k-NN algorithm essentially has no training phase and testing a new example is computationally expensive, as we have to search for the nearest neighbour amongst all previously classified examples.

On the other hand, Table 4 summarizes the results for the second experiment, i.e., training on dataset 1 and testing against dataset 2. Although the results of the first experiment were impressive, the results of the second experiment show that the ML method for IDS still has limitations in terms of generalisation. Noticeably, all algorithms lead to above 99% of recall, while showing lower values for the precision. This implies that most of the detection errors are due to FPs, and there are occasional FNs. Furthermore, the SVM algorithm demonstrates the best generalization performance among all applied classification algorithms, where its accuracy reaches up to 84%. Finally, similar to the first experiment, the k-NN algorithm demonstrates an impressive generalisation performance. As seen in Table 4, its performance is comparable with the one of the SVM algorithm, with 84 per cent of detection accuracy.

The results reveal that ML methods for IDS achieve a satisfactory performance, but they still lead to a high number of FPs, which can render the IDS into an inefficient and troublesome tool since when receiving an intrusion alert, the user has no idea if it is originated from an intrusive event or it is just a false alarm. Therefore, additional mechanisms are needed to further inspect the alerts before notifying the user. This is what is done by the post detection processing modules (i.e., Intrusion Probability Assessment and Alert Analysis modules) of our proposed IDS architecture in Fig. 1. These modules essentially generate an alert when the overall probability of intrusion exceeds a predefined threshold, relying on how many consecutive positive outcomes (indicating a malicious incident) are observed in a row.

5 Conclusion and Future Work

In this paper, we proposed an autonomous host-based IDS for Android mobile devices. The proposed IDS is based on dynamic analysis of the device's behaviour for detecting suspicious behaviour on Android mobile devices. In other words, the detection takes place through analysis of the deviations in device's behaviour described through a vector of features. The proposed IDS continuously monitors a specific set of features of the mobile device at the device level, i.e., without individually inspecting the behaviour of each application, in order to define its run-time behaviour and apply machine learning techniques to classify it as benign or malicious. The simulation

results demonstrate a promising detection accuracy of above 85%, reaching up to 99.99%. For future work, we plan to incorporate statistical algorithms for malware detection in Android mobile devices. An interesting aspect of this approach is that it relies primarily on the benign data, for building a normal profile, and requires only few malicious examples for tuning the IDS. This is crucially important for an IDS design since constructing a training dataset with an equal number of benign and malicious examples is tedious in practice.

Acknowledgments

José Ribeiro would like to acknowledge his PhD grant funded by the Fundação para a Ciência e Tecnologia (FCT-Portugal) with reference SFRH/BD/112755/2015. This work is supported by the European Regional Development Fund (FEDER), through the Regional Operational Programme of Centre (CENTRO 2020) of the Portugal 2020 framework [Project MOBITRUST with Nr. 003343 (CENTRO-01-0247-FEDER-003343)].

References

1. L. Polla, F. Martinelli and D. Sgandurra, “A Survey on Security for Mobile Devices,” Communications Surveys & Tutorials, IEEE, vol. 15(1), pp. 446-471, 2013.
2. M. Becher, F. C. Freiling, J. Hoffmann, T. Holtz, S. Uellenbeck and C. Wolf, “Mobile security catching up? Revealing the nuts and bolts of the security of mobile devices,” *In Security and Privacy (SP)*, pp. 96-111. IEEE, 2011.
3. G. Mantas, N. Komninos, J. Rodriguez, E. Logota and H. Marques, “Security for 5G Communications,” Eds., John Wiley & sons, Ltd, Chichester, UK, 2015, pp. 207-220.
4. A. Arabo and B. Pranggono, “Mobile Malware and Smart Devices Security: Trends, Challenges and Solutions,” Control Systems and Computer (CSCS), 2013 19th International Conference, pp. (526-531). IEEE, (2013).
5. Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C. and Weiss, Y., 2012. “Andromaly”: a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems, 38(1), pp.161-190.
6. Burguera, I., Zurutuza, U. and Nadjm-Tehrani, S., 2011, October. Crowdroid: behavior-based malware detection system for android. In Proc. of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (pp. 15-26). ACM.
7. R. Xu, H. Saïdi, and R. Anderson, “Aurasium: Practical policy enforcement for Android applications,” in Proc. 21st USENIX Conf. Security Symp., 2012, USENIX Association.
8. P. Borges et al., "Towards a Hybrid Intrusion Detection System for Android-based PPDR terminals," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, pp. 1034-1039, 2017.
9. N. Ulltveit-Moe, V. A. Oleshchuk and G. M. Koien, “Location-aware mobile intrusion detection with enhanced privacy in 5G context.,” Wireless Personal Communications, vol. 57(3), pp. 317-338, 2011.
10. D. Huang, X. Zhang, M. Kang and J. Luo, “MobiCloud: building secure cloud framework for mobile computing and communication,” In Service Oriented Sys-tem Engineering (SOSE), 2010 Fifth IEEE International Symposium, pp. 27-34, (2010).