Behaviour Based Anomaly Detection for Smartphones Using Machine Learning Algorithm

Khurram Majeed[1], Dr Yanguo Jing[2], Dr Dusica Novakovic[3], and Prof Karim Ouazzane[4]

***Abstract*—**Since the first handheld cellular phone was introduced in 1973, the mobile phones have evolved into immensely popular smartphones. These devices provide all-in-one expediency by integrating traditional mobile phones with handheld computing devices making them more open and general purpose. Smartphones have become hosts for sensitive or personal data and applications. However many smartphones are prone to attacks. In the past few years, hundreds of malware have appeared to target these devices, propagating through various channels including SMS, MMS, Bluetooth, traditional IP-based applications and platform level vulnerabilities. These malware not only pose a threat to mobile system data confidentiality, availability and integrity but can also result in unwanted billing, depletion of battery power and denial-of-service attacks. Current smartphone malware detection and prevention techniques are generally limited to signature-based antivirus scanners. These can efficiently detect malware with a known signature, but they are unable to detect new and unknown malware. This shortcoming creates a window of opportunity for attackers. Recent research in smartphone security has also focused on classification of applications as either benign or malicious by using data mining and machine learning algorithms. The scope of these techniques is also limited because they rely on static analysis of application permissions and system calls etc. Consequently, it is vital to address these shortcomings. In this paper, we propose a novel, platform independent behavior-based anomaly detection framework for mobile devices. The fundamental premise of this framework is that every smartphone user has a unique usage patterns. By modeling these patterns into a profile we can uniquely identify users. To test this hypothesis, a data collection application was developed to accumulate real-life dataset consisting of application usage statistics, various system metrics and contextual information from mobile devices. Then K-Means Clustering algorithm was used to create baseline usage profile for each user. To detect accuracy of profiling, we tried to fit every user's data into profiles of other users and monitored the deviation. By monitoring the deviation between profiles, we were able to assert our hypothesis. Whilst some users were difficult to profile, a significant proportion fell within the performance expectations. Our tests show that 80% of user profiles had accuracy (measure of uniqueness) between 64% and 95%. Another novelty of our research is that it is one of the first to implement behavior based anomaly detection on mobile devices. The implications of our research are far reaching as it can also provide means of transparent, continuous user authentication on mobile devices.

***Keywords*—** Anomaly detection, behavior profiling, malware, security, smartphones.

I. Introduction

SMARTPHONES have evolved from simple mobile devices into complex yet compact minicomputers which can connect to a wide assortment of communication networks to service its users, such as: voice calling and messaging through cellular network, video conferencing through 3G, 4G and Wi-Fi, Door-to-Door navigation by Global Positioning System (GPS), multimedia sharing through Bluetooth, mobile payments by using Near Field Communication (NFC), data synchronization with personal computer, high end gaming.

While this plethora of features intends to provide convenience to users of mobile devices, there are also threats, which can make their life less comfortable. Some of these inconveniences include, but are not limited to, loss or theft of the device, service fraud, mobile malware, information disclosure, DoS (Denial-of-Service) attacks, Smishing and Vishing. In 2004, the first articles about smartphone malware were published stating that the mobile devices were the next generation of targets [1]-[2]. Statistics show that more than 1000 smartphone malware variants, such as worms, Trojan horses; other viruses and spyware have been unleashed against the mobile devices in the world since 2004. In the last year alone 143,211 new modifications of mobile malware were detected [3].

A number of security techniques have been developed to combat these threats, both on the mobile device and the service provider's network. Most commonly used host-based mobile security solutions include personal identification number (PIN) number/pattern based authentication, mobile antivirus for malware detection and firewalls to block unwanted network traffic. Although PIN number and pattern based authentication is widely used on today's smartphones, many users don't employ them properly which limits their usefulness [4]-[5]. Unwanted network traffic is generally blocked by firewalls while signature-based antivirus software solutions use a signature of known malware for their detection.

Khurram Majeed[1] is with the School of Computing, London Metropolitan University, London, UK (e-mail: k.majeed@londonmet.ac.uk).

Dr Yanguo Jing[2] is with the School of Computing, London Metropolitan University, London, UK (e-mail:y.jing@londonmet.ac.uk).

Dr Dusica Novakovic[3], is with School of Computing and Technology, University of West London.UK (e-mail: d.novakovic@uwl.ac.uk).

Prof Karim Ouazzane[4] is with the School of Computing, London Metropolitan University, London, UK (e-mail:k.ouazzane@londonmet.ac.uk).

Obtaining latest virus signatures and network traffic rules are not easy tasks. As a consequence, mobile antivirus software leave smartphone users exposed to new malware until the signature is available and the security solution provider releases a patch (zero day attacks).

Bulygin [6] researched propagation of MMS and Bluetooth worms and demonstrated that in the worst case a MMS worm targeting phone book numbers can infect more than 0.7 million devices in about three hours. Furthermore, Oberheide et al. [7] demonstrated that the average time required for a signature-based anti-virus engine to become capable of detecting new threats is 48 days. In some cases, malware instances target a specific and relatively small number of mobile devices (e.g. for extracting confidential corporate information or track owners location) and will therefore take longer to be discovered. The growth rate for the virus signature list for mobile viruses in last two years is equivalent to the growth rate of the virus signature list for personal computers in twenty years. Additionally, current mobile devices are unable to support the existing antivirus technologies available for personal computers because of limited processing power, storage space, battery life and memory [8]-[9].

So as it stands, these techniques have serious shortcomings that make them inefficient for mobile devices. This leads us to explore more sophisticated systems such as Intrusion Detection System (IDS). Historically, service providers have implemented such systems on the network-side to monitor mobile user's calling and migration activities to detect telephony service fraud. Hilas et al. used Artificial Neural Networks (ANN's) in order to detect anomalous behavior indicating a fraudulent use of the operator services [10].

Since today's smartphones have the ability to access multiple communication channels and accommodate a wide variety of service, existing network-based security mechanisms are unable to offer a comprehensive protection for mobile devices. Also, existing security solutions for mobile devices with ability to detect user related activities are either limited or application specific.

The focus of this paper is to fill this void by presenting host based anomaly detection framework for mobile devices. Our system utilizes mobile device usage behavior to profile users in order to detect anomalous behavior. The hypothesis behind this proposition is that, every user has a unique usage pattern and Machine-learning algorithms can be applied to convert these patterns into profiles. Once we have a user profile, deviations can be monitored to detect anomaly. This system has the ability to detect mobile device misuse and can also be extended to provide transparent and continuous user authentication.

The rest of the paper is organized as follows. Section 2 briefly reviews the previous work on the topic. Section 3 presents the methodology used in our work. Section 4 presents the data used as input, the profile generation process and the anomaly detection. Section 5 presents the results of our framework followed by a conclusion.

II. Previous work

Research in mobile device security has become well established paradigm in the last decade whereby a substantial amount of focus has been given to the areas of authentication, antivirus, firewalls, and IDS. Of particular interest in the context of this paper is the research carried out in behavior based IDS techniques.

Early research in the field was implemented on the mobile service provider's network to monitor user calling and migration behavior to detect telephony service fraud [11]. These techniques were based on the idea of profiling calling patterns observed over a period of time and then comparing it against subsequent usage data, with deviations above a predefined threshold resulting in an alarm. Monitoring geolocation for misuse detection is based on the premise that user's travel patterns are predictable and monitoring these can reveal anomalous behavior. A user's mobility profile can be built using his/her location information obtained from cellular networks (i.e. cell ID) or via a GPS sensor (i.e. longitude, latitude). Probability measures are then to determine likelihood of user's presence at a given location that leads to misuse detection. Anomaly detection (sudden increase in network activity in an area having normally low network activity), location-based detection (a user cannot be active in two different locations at the same time) also leads to detection of anomalous behavior of individual mobile-phone users. Other notable work by Sun et al. and Hall et al. [12]-[13] further consolidated this idea, as they were able to achieve high detection rates for service misuse and were able to detect unforeseen attacks. These techniques don't hamper the mobile device processing capabilities since the processing happens at the network service provider's side.

Various researchers have also devoted special efforts to implement IDS based on analysis of battery power consumption patterns in order to block distributed DoS attacks [14]-[15]. Taxonomy of malware behavior [16] demonstrated that by monitoring the mobile device's electrical current and evaluating its correlation with known malware patterns can facilitate attack detection. Slightly more complicated IDS designed to detect battery exhaustion attacks was presented by Nash et al. [17]. This system monitors performance, energy, and memory constraints of mobile phones.

Kim et al. [18] presented a power-based malware detection framework that monitors, detects, and analyses previously unknown energy depletion threats. A hybrid Battery Sensing Intrusion Protection System (B-SIPS) [19] that combined battery sensing with signature-based SNORT IDS demonstrated that abnormal current changes can be monitored to trigger security alerts. These malware detection solutions have been shown to work very well for malware with abnormal power signature, but these may not be able to detect a malware with a power signature similar to that of legitimate application or malware hijacking a legitimate application to infect the mobile device.

Yap & Ewe [20] used a proof-of-concept behavior checker

to identify malicious behavior in a mobile system. They used a Nokia Mobile phone running on Symbian OS. They demonstrated the detection of a simulated Trojan attempting to use the message server component without authorization to create an SMS message. A collaborative proxy-based virus detection and alert system called SmartSiren [21] can detect device or system-wide abnormal behaviors by the joint analysis of communication activity of monitored smartphones. Bose et al. [22] presented a behavioral detection framework that employed Temporal-Logic approach to detect malicious activity over time. This work gave an efficient representation of malware behavior based on a key observation that the logical ordering of applications actions over time often reveals malicious intent even when each action alone may appear harmless. Schmidt et al. [23] present an anomaly detection system that monitors smartphones running Symbian OS in order to extract features that describe the state of the device by using a Symbian monitoring client. The collected features were then forwarded to a Remote Anomaly Detection System (RADS). The gathered data was used by anomaly detection methods in order to distinguish between normal and abnormal behavior. These approaches have a common problem that they suffer from a high computational burden that is moved to an external server in most cases hence they rely on remote servers.

Despite the fact that all of the aforementioned researches have significantly contributed to the anomaly based IDS for mobile devices, several important issues remain unsolved. Currently the main disadvantage of anomaly based IDS technique is high false-positive rate (FPR) [24]. Hence it is vital to explore new techniques that increase the accuracy while reducing the FPR. Also, device side behavior-based IDS focused only on telephony, SMS or battery power consumption. Nevertheless, mobile devices have evolved a great deal and they provide services through various communication channels. Thus it is crucial that the data originating from provision of these services is also taken into account.

III. Methodology

In the view of the above discussion we believe that more sophisticated controls such as behavior-based IDS are needed to combat mobile device intrusion and misuse. To achieve this, various user actions and contextual information should be collected to create behavioral profiles and to effectively distinguish between legitimate users and intruders. Features from this collected dataset (collected from real world mobile devices) are used as input for machine learning algorithm that is implemented on a remote server. In future, the IDS can be built to directly run on the mobile device for real-time detection.

IV. Proposed Framework

In this section, we present the conceptual framework of our behavior based IDS. This framework can detect malicious activity on the mobile device in real-time by employing unsupervised machine learning technique called k-means clustering. Figure 1 shows this framework. The detailed explanation of the important components of our framework is presented in the following sections.

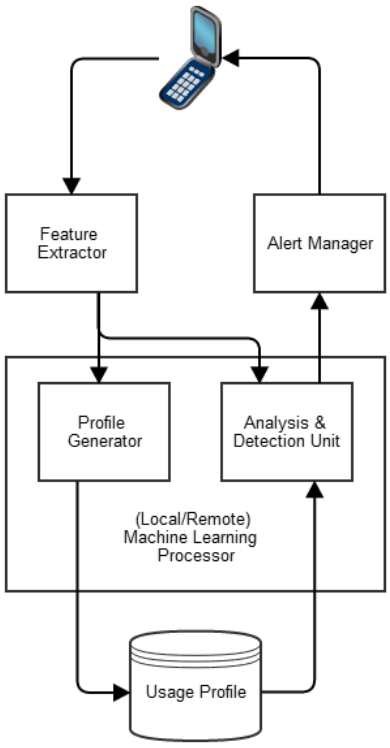

Fig. 1 Behaviour based IDS

A. Feature Extractor

The Feature Extractor is the most crucial component of the framework as it communicates with various components of the Mobile Device Operating System, including the kernel and the Application Framework Layer in order to collect raw features (system metrics). The collected features are fed to the Machine Learning Processor as an input during its two operational phases. The feature collector is implemented as an Android application that uses a background service to collect system features from a mobile device. The epoch for feature collection is set at every 5 minutes. It was assumed that the smartphone owners use their phones differently under different conditions. For example, they may place more calls or send messages out of working hours, they may prefer to play games or read during travelling etc. So in order to ensure that all possible usage scenarios were included in the usage profile, optimum number of observations had to be made. According to our tests, the feature extraction should continue for two weeks to ensure comprehensive usage profile.

Our feature extraction service is very lightweight on processor, memory and battery resources. If we used an always-running service, it would be a waste of resources for the mobile device. This type of service is also prone to getting force closed by the device user with the use of any Task Killer application. Both these factors will jeopardize the effectiveness of the framework.

To overcome this issue, we have developed a background service that employs the Android's equivalent to Cron Jobs for Linux or Scheduled Tasks for Windows. That is, using Alarm Manager with an IntentService. The idea is that the main Activity sets the time interval (5 minutes) for feature extraction. After every epoch, the AlarmManager starts the background service (feature extraction). After data collection is finished the AlarmManager is set for next interval and the service terminates.

An Android device is in active state when a user is interacting with the device. When the screen goes off after the time interval (set by the user usually from 15 seconds to 10 minutes) Android system powers down the CPU and GPU, Wi-Fi and other sensors in order to reduce the power consumption. An Android mobile device will spend more time in sleep mode than being actively used by the user. Hence, if the device is asleep during the time when next feature extraction is scheduled, no data can be extracted. To counter this issue we need to bring the device to awake state. This is achieved by creating a WakeLock that keeps the device awake for as long as all the feature extractors have finished their task.

After every epoch, following system metrics are recorded by feature extractor service and saved into SQLite database on the device:

1) CONTACT CALLS: The number of calls made to phone numbers stored in the address book.
2) CONTACT CALLS SEC: The duration (seconds) of calls made to phone-numbers stored in the address book.
3) UNKNOWN CALLS: The number of calls made to phone numbers not present in the address book.
4) UNKNOWN CALLS SEC: The duration (seconds) of calls made to phone-numbers not present in the address book.
5) SMS OUT CONTACTS: The number of SMS messages sent to phone-numbers stored in the address book.
6) SMS OUT UNKNOWN: The number of SMS messages sent to phone-numbers not present in the address book.
7) CPU USAGE: The Average CPU usage since last epoch
8) RAM USAGE: The Average RAM usage since last epoch.
9) SYS APPS: The number of applications running on the device that are pre-installed by Google.
10) USER APPS: The number of applications running on the device that are pre-installed by user.
11) SYS SERVICES: The number of services running on the device that are pre-installed by Google.
12) USER SERVICES: The number of services running on the device that are installed by the user.
13) BYTES TX: The number of bytes of data transmitted from the device since last epoch.
14) BYTES RX: The number of bytes of data received by the device since last epoch.
15) LOC LAT: The latitude of the geographic location of the device based on cell towers.
16) LOC LON: The longitude of the geographic location of the device based on cell towers.
17) TIME SLICE: This represents the portion of a week. In order to get more accurate mobile device usage, a week was divided into 28 slices. The full mapping of time of observation to TIME SLICE is shown the Table 1.
18) SCREEN ON: The status of the mobile device screen at the time of feature collection. If the user is physically interacting with the device, 1 is recorded, else 0.
19) READ AT: The time of the feature collection. After 2 weeks of feature extraction, the users are asked to send the data (the SQLite database file) via email. This option provided within the application.

TABLE I
OBSERVATION TIME TO TIME_SLICE MAPPING

Day	06:00 to 11:59	12:00 to 17:59	18:00 to 23:59	24:00 to 05:59
MONDAY	11	12	13	14
TUESDAY	21	22	23	24
WEDNESDAY	31	32	33	34
THURSDAY	41	42	43	44
FRIDAY	51	52	53	54
SATURDAY	61	62	63	64
SUNDAY	71	72	73	74

B. Machine Learning Processor

The Machine Learning Processor (hereinafter abbreviated to MLP) relies on k-means clustering technique to create usage profile from the input data. We have implemented our MLP using Mathworks Matlab 2013a, which provides feature rich Machine Learning library and tools for data analysis and visualization. The MLP has two operational modes and each mode is controlled by separate component of MLP. The processes of usage profile generation and anomaly detection are discussed in more detail in the followings sections.

TABLE II
MAPPING FEATURES FROM DATABASE TO CSV FILE

CSV	Database
CALLS_CONTACT	CALLS + UNKNOWN_CALLS
CALLS_DURATION	CONTACT_CALLS_SEC + UNKNOWN_CALLS_SEC
SMS_OUT	SMS_OUT_CONTACTS + SMS_OUT_UNKNOWN
CPU_USAGE	CPU_USAGE
RAM_USAGE	RAM_USAGE
SYS_APPS	SYS_APPS
USER_APPS	USER_APPS
SYS_SERVICES	SYS_SERVICES
USER_SERVICES	USER_SERVICES
BYTES_TX	BYTES_TX
BYTES_RX	BYTES_RX
LOC_LAT	LOC_LAT
LOC_LON	LOC_LON
SCREEN_ON	SCREEN_ON
TIME_SLICE	TIME_SLICE

C. Profile Generator

In training mode the data (extracted features) from the Feature Extractors is processed and scaled for consistency, a statistical test is performed to determine the optimum number

of clusters k and then a usage profile is generated. Each stage of profile generation is explained in more detail in the following sections.

Once the data (extracted features) from the Feature Extractors is received in the form of a database file from smartphone user, its data is exported to a comma-separated values (CSV) file. The mapping of database table to CSV file is shown in Table 2. Once the CSV file is created, its data is imported into Matlab for further processing. In order to facilitate the scaling process, the user's Geolocation (longitude and latitude) is converted into distance from Charing Cross, London. The reason for this conversion is two-fold; firstly, we needed a reference location for all users, secondly, all road distances in UK are calculated from Charing Cross. The conversion is done using the Haversine formula [25]. This formula is used to calculate the great-circle distance between two points, which is the shortest distance over the earth's surface.

In geometry, all dimensions are equally important. A distance of 5 on X dimension is the same as the distance of 5 on the Y dimension. It doesn't matter what units X and Y are measured in, as long as they are the same. Unfortunately, in commercial data mining problems including ours, usually no common scale is available because of the different units being used to measure different things. For example, in our case CPU USAGE, CALLS, USER APPS, BYTES TX cannot be converted to a common unit. On the other hand, it will be misleading that a CPU USAGE of 50% is indistinguishable from a value of 50 Bytes for BYTES TX. The best solution to overcome this problem is to use z-scores which are also known as standardizing values, as it removes the bias present due to different units. By standardizing all features so that each has a mean of zero and standard deviation of one, they contribute equally when the distance between two records is computed. Equation (1) shows the calculation of a z-score.

$$z - scores = \frac{x - \mu}{\sigma} \quad (1)$$

Here x is the value of a single feature (e.g. CPU USAGE) from list of observations μ is the mean and σ is the standard deviation for all values of that feature. Being a non-deterministic process, K-Means clustering doesn't always yield the same result. The accuracy of the clustering process (i.e. the correctness of the usage profile) depends on number of clusters (denoted by k). The selection of this number depends on the prior domain knowledge. It is also important to note that finding k in two, three, or four dimensions is much easier than in twenty. For the clustering to be accurate, records should have similar values in all dimensions. To assist in the process of selecting value of k for high dimensional data, we have used both visual and statistical techniques.

We used Multi Dimensional Scaling (MDS) to provide a visual representation of the pattern of proximities (i.e. similarities or distances) among a set of observations made on the smartphone. Although MDS made it possible to visually analyze the data to estimate the number of clusters, it was not definitive and has to be complemented by some form of statistical measure. Hence, we have used cluster silhouette test, which provides a statistical measure of cluster goodness. K-Means Clustering process is performed on usage data. The value of k which results in highest silhouette score is used for profiling.

TABLE III
SILHOUETTE SCORES

k (clusters)	Silhouette score
2	0.4140
3	0.3893
4	0.4044
5	0.4377
6	0.2675
7	0.2744
8	0.2824
9	0.2513
10	0.2608

Table III shows the silhouette score for K-Means Clustering process where with k is in the range of 2 to 10. Once best value of k is determined using the silhouette test, further processing the results of K-Means Clustering generates the usage profile of the user. Since K-Means clustering suffers from local minimization issue and to overcome this, the clustering process is replicated until a global minimum is achieved. After processing, the usage profile is saved as Matlab's native data file format. The usage profile consists of the following data:

1) TRAINING DATA: The usage data from user's smartphone that is scaled using z scores and represented as n-by-14 matrix. Here n is the number of observations.
2) MU: The mean calculated during z score calculation. It is represented as 1-by-14 vector where each column represents the mean of the corresponding variable in the training data.
3) SIGMA: The standard deviation calculated during z-score calculation. It is represented as 1-by-14 vector where each column represents the standard deviation of the corresponding variable in the training data.
4) IDX: The cluster indices of observations (data points) represented by n-by-1 vector.
5) C: The cluster centroids represented as k-by-14 matrix, where k is number of clusters. Each row of this matrix represents a cluster centroid.
6) CB: The cluster boundaries, i.e. the distance from centroid and the farthest point with the cluster. It is represented by a 1-by-k vector.
7) D: The distance from each data point (observation)

D.Anomaly Detector

Once a usage profile is generated, the MLP operates in testing mode. Whereupon, the data from the Feature Extractors is processed, scaled and then compared with the usage profile from the training phase to determine deviation. When the anomaly detector encounters a persistent deviation from the

usage profile, it notifies the Alter Manager to take an appropriate action.

E. Alert Manager

The role of alert manager is to notify the smartphone user about any anomalous activity on the mobile device. Depending upon the user's configuration various actions can be taken:

1) A warning message shown in the smartphone.
2) Prompting the user to enter preconfigured pin-code to unlock the device.

Since the MLP has been implemented and tested in Matlab, this component has not been implemented on smartphone and left as future work.

For the prototype implementation, the MLP is implemented on a remote server. It performs the following tasks:

1) Receive data (extracted features) from smartphones.
2) Perform data pre-processing for visualization and feature generation.
3) Generate a model to create usage profiles from produced features.
4) Test subsequent usage data against profiles to identify profile accuracy.

Since usage profile generation is a computationally expensive task, many low-end devices may not be able to generate usage profile. In such a situation remote IDS serves as a very good option. However, for real time profile generation and anomaly detection this processor should operate on the mobile device.

V. RESULTS

It is vital to note that the proposed framework is novel and no other comparable framework exists to our knowledge that can be used as a reference. Absence of a benchmark implies that this framework can only be validated with data collected during our testing. An Android application was developed and deployed on 10 test devices. The testers were asked to run the data collection application for two weeks and email the collected data. The data collected from the devices was processed; scaled using z-score technique and some variables were transformed. To measure the quality of clustering results, there are two kinds of validity indices: An External index is a measure of agreement between two partitions where the first partition is the a priori known clustering structure, and the second results from the clustering procedure [26]. For external indices, we evaluate the results of a clustering algorithm based on a known cluster structure of a data set (or cluster labels). Internal indices are used to measure the goodness of a clustering structure without external information [27]. For internal indices, we evaluate the results using quantities and features inherent in the data set. The optimal number of clusters is usually determined based on an internal validity index.

In case of our MLP, we don't have access to External validation criteria, i.e. the number of clusters in unknown. Hence we relied on visual (Multidimensional scaling) and statistical (Silhouette test) measures to determine the appropriate number of cluster. Table 4 presents the optimum number of clusters (from silhouette score) for each user's profile and average value of accuracy of the profiles generated by the MLP.

The accuracy value represents the uniqueness of the profiles. With 80% of our user profile having accuracy figures between 64% and 95% and total average accuracy of 74.48% for all profiles, the results of our framework are within expected range.

VI. CONCLUSION

Today's smartphones have evolved into all-in-one computing devices capable of providing a wide range of service over various communication channels. Due to immense interaction between users and these devices there is a strong need to provide anomaly based IDS. Although a significant research has been carried out in mobile device IDS, user behavior based IDS is still immature and several issues still exist. The contribution of this paper is two-fold; firstly, we have presented the rationale for the need to have behavior based IDS based on the weaknesses of existing security solutions; secondly we have presented an efficient framework that combines broad selection of usage metrics and contextual information to create an integrated user behavior profile. The ultimate goal of this technique is to construct a profile for normal user, which when compared with subsequent data will allow the detection of deviations. The results of our experimental procedure show that it is possible to achieve high accuracy with behavior based profiling. Currently, profiling is done on a proxy server, which is done to facilitate the calibration of machine learning algorithm. Based on our encouraging results, we are currently working on extending this work by implementing the MLP on mobile devices to provide real-time detection. This will allow us to further test the effectiveness of our framework in term of resource utilization and detection speed in real-time on smartphones.

REFERENCES

[1] D. Dagon, T. Martin, and T. Starner. Mobile phones as computing devices: The viruses are coming! IEEE Pervasive Computing, 3(4):11-15, Oct. 2004. http://dx.doi.org/10.1109/MPRV.2004.21

[2] M. Piercy. Embedded devices next on the virus target list. Electronic Systems and Software, 2(6):42-43, 2004. http://dx.doi.org/10.1049/ess:20040612

[3] V. Chebyshev and R. Unuchek. Mobile Malware Evolution: 2013. Technical report, Feb. 2014.

[4] N. Clarke and S. Furnell. Authentication of users on mobile telephones - a survey of attitudes and practices. Computers & Security, 24(7):519 - 527, 2005. http://dx.doi.org/10.1016/j.cose.2005.08.003

[5] S. Kurkovsky and E. Syta. Digital natives and mobile phones: A survey of practices and attitudes about privacy and security. In Technology and Society (ISTAS), 2010 IEEE International Symposium on, pages 441-449, June 2010.

[6] Y. Bulygin. Epidemics of mobile worms. 2007 IEEE International Performance Computing and Communications Conference, (2):475-478, Apr 2007.
http://dx.doi.org/10.1109/PCCC.2007.358929

[7] J. Oberheide, E. Cooke, and F. Jahanian. Cloudav: N version antivirus in the network cloud. In Proceedings of the 17th conference on Security symposium, Security Symposium 08, pages 91-106, Berkeley, CA, USA, 2008. USENIX Association.

[8] Alexander Gostev. Mobile malware evolution: An overview, part 2, Oct 2006. [Accessed: April 28, 2011].

[9] Denis Maslennikov. Mobile malware evolution: An overview, part 4, Mar 2011. [Accessed: April 28, 2011].

[10] C. S. Hilas and P. A. Mastorocostas. An application of supervised and unsupervised learning approaches to telecommunications fraud detection. Knowledge-Based Systems, 21(7):721 - 726, 2008.
http://dx.doi.org/10.1016/j.knosys.2008.03.026

[11] A. Boukerche and M. S. M. A. Notare. Behavior-based intrusion detection in mobile phone systems. J. Parallel Distributed Computing, 62(9):1476-1490, 2002.
http://dx.doi.org/10.1006/jpdc.2002.1857

[12] J. Hall, M. Barbeau, and E. Kranakis. Anomaly-based intrusion detection using mobility profiles of public transportation users. Technical report, In Proceedings of the Wireless and Mobile Computing, Networking and Communications, 2005.

[13] B. Sun, F. Yu, K. Wu, and V. C. M. Leung. Mobility-based anomaly detection in cellular mobile networks. In Proceedings of the 3rd ACM Workshop on Wireless Security, WiSe '04, pages 61-69, New York, NY, USA, 2004. ACM.

[14] R. Racic. Exploiting mms vulnerabilities to stealthily exhaust mobile phone's battery. In SecureComm 06, pages 1-10. SECURECOMM, 2006.

[15] T. Martin, M. Hsiao, D. Ha, and J. Krishnaswami. Denial of service attacks on battery powered mobile computers. Pervasive Computing and Communications, IEEE International Conference on, 0:309, 2004.

[16] G. Jacob, H. Debar, and E. Filiol. Behavioral detection of malware: from a survey towards an established taxonomy. Journal in Computer Virology, 4(3):251-266, Feb 2008.
http://dx.doi.org/10.1007/s11416-008-0086-0

[17] D. C. Nash, T. L. Martin, D. S. Ha, and M. S. Hsiao. Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices. PERCOMW 05, pages 141-145, Washington, DC, USA, 2005. IEEE Computer Society.

[18] H. Kim, J. Smith, and K. G. Shin. Detecting energy greedy anomalies and mobile malware variants. In Proceedings of the 6th international conference on Mobile systems, applications, and services, MobiSys 08, pages 239-252, New York, NY, USA, 2008. ACM.

[19] T. Buennemeyer, T. Nelson, L. Clagett, J. Dunning, R. Marchany, and J. Tront. Mobile device profiling and intrusion detection using smart batteries. In Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, page 296, Jan 2008.

[20] T. S. Yap and H. T. Ewe. A mobile phone malicious software detection model with behavior checker. HSI 05, pages 57-65, Berlin, Heidelberg, 2005. Springer-Verlag.

[21] J. Cheng, S. H. Wong, H. Yang, and S. Lu. SmartSiren: virus detection and alert for smartphones. In Proceedings of the 5th international conference on Mobile systems, applications and services, MobiSys '07, pages 258-271, New York, NY, USA, 2007. ACM.

[22] A. Bose, X. Hu, K. G. Shin, and T. Park. Behavioral detection of malware on mobile handsets. In Proceedings of the 6th international conference on Mobile systems, applications, and services, MobiSys '08, pages 225-238, New York, NY, USA, 2008. ACM. [Accessed May 4, 2011].

[23] A.-D. Schmidt, F. Peters, F. Lamour, and S. Albayrak. Monitoring smartphones for anomaly detection. In Proceedings of the 1st international conference on Mobile Wireless Middleware, Operating Systems, and Applications, MOBILWARE '08, pages 40:1-40:6, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[24] T. Alpcan, C. Bauckhage, and A.-D. Schmidt. A probabilistic diffusion scheme for anomaly detection on smartphones. In Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices, volume 6033 of Lecture Notes in Computer Science, pages 31-46. Springer Berlin Heidelberg, 2010.

[25] Wikipedia. Haversine formula | Wikipedia, the free encyclopedia, 2014. [Online; accessed 9-March-2014].

[26] S. Dudoit and J. Fridlyand. A prediction-based re sampling method for estimating the number of clusters in a dataset. Genome Biology, 3(7):1-21, 2002.
http://dx.doi.org/10.1186/gb-2002-3-7-research0036

[27] A. Thalamuthu, I. Mukhopadhyay, X. Zheng, and G. C. Tseng. Evaluation and comparison of gene clustering methods in microarray analysis. Bioinformatics, 22(19): 2405-2412, Sept. 2006.
http://dx.doi.org/10.1093/bioinformatics/btl406