Impact of Dataset Representation on Smartphone Malware Detection Performance

Abdelfattah Amamra, Chamseddine Talhi, and Jean-Marc Robert

Department of Software Engineering and Information Technologies,
École de Technologie Supérieure,
Montreal, Canada
Abdelfattah.amamra.1@ens.etsmtl.ca
{Chamseddine.Talhi,Jean-Marc.Robert}@etsmtl.ca

Abstract. Improving Smartphone anomaly-based malware detection techniques is widely studied in recent years. Previous studies explore three factors: dataset size, dataset type and normal profile model. These factors improve the performance, but increase computation complexity and the required memory space. In this paper we explore a new factor: the dataset representation. Dataset representation is the format adopted to organize and represent data. To investigate the impact of this factor, we examine four machine learning classifiers with three different dataset representations. Those dataset representations are: successive system calls, bag of system calls and patterns frequency system calls. The used dataset is a collection of system call traces of Smartphone executing Android 2.2. We analyse the performance of each classifier and deduce the influence of dataset representation on accuracy and false positive rates. The results show that the dataset representation has a potential impact on the performance of classifiers with low computational and memory cost.

Keywords: Smatphone security, system calls, intrusion detection, machine learning, Anomaly technique.

1 Introduction

Smartphones are very popular and widely used in business and personal life in recent years due to their capabilities and services. Therefore, their market is growing very fast. According to the International Data Corporation (IDC), they expect about 982 million units will be shipped by the end of 2015 [1]. This popularity of Smartphones attracts malware developers. Google recently found about 50 applications that were infected with hidden malicious code DroidDream. The applications were loaded in the official applications store (Android Market). They were available for about 4 days and between 50,000 and 200,000 copies were downloaded [1]. Thereby, the need of malware detection tool is now more curial.

Smartphone malware detection techniques are classified broadly into two main classes: signature-based and anomaly-based [2]. Signature-based techniques look for patterns that match with malware patterns in their database. Anomaly-based

C. Fernandez-Gago et al. (Eds.): IFIPTM 2013, IFIP AICT 401, pp. 166–176, 2013.

techniques maintain normal behavior profiles and any deviation from these profiles is considered malicious [2]. The main advantage of anomaly detection techniques is the ability to detect unknown malwares. Therefore, they have been actively investigated. However, anomaly-based malware detection still needs more investigation and improvement to reduce false positive and increase detection accuracy. Prior works explored actively the factors having an impact on the performance of anomaly-based detection techniques. They examined three factors: dataset size, dataset type, and the model of normal profile. From related work, we can learn that increasing the quantity of dataset or varying the data type used to characterize normal behavior profile improves the normal profile accuracy. However, the resulting profile has usually high computational and memory space overhead. Therefore, these factors are not suitable to improve performance of anomaly-based detection technique on limited resources environment, such as Smartphone. Thereby, we should explore other factors that have significant influence on the performance of anomaly-based technique as well as low computation and memory cost. In this work, we highlight the dataset representation factor. We examine the impact of this factor on the performance of anomaly-based technique on Smartphone system call traces. Dataset representation is the format of how the data is organized and represented. This format is used during the training phase as well as the detection phase. The process of preparing dataset representation has light computation cost and memory complexity which make it suitable approach for Smartphone malware detection technique.

In this study, we examine three dataset representations on Smartphone system call traces. Those representations are:

- Successive system calls representation, where the ordering information between system calls in sequence is considered. This dataset representation is used in prior work.
- Bag of system calls representation, where the successiveness of system calls in sequence is disregarded and only the frequency of each system call is preserved. It is used in prior work.
- Patterns frequency system calls representation combines features of the two previous representations. Pattern-frequency representation regards the successive order information of system calls in short pattern, and regards the frequency of each pattern in the sequence. To the best of our knowledge, this representation has not been studied in the past.

We evaluate performance of different machine learning classifiers with the above dataset representations and we deduce the influence of each dataset representation on the performance of each classifier. We select the most known and used classifiers: Support Vector Machine (SVM), Naïve Bayes (NB), Logistic Regression (LR), and the Decision Tree (DT). We experiment our approach on system call traces collected from an HTC Dream Smartphone. These system call traces represent the execution of the 100 most downloaded normal applications and the 90 available real malwares.

The rest of this paper is organized as follows: related work is introduced in section 2. Section 3 presents the dataset presentations. Section 4 presents experiments and results analysis. We end with conclusion in section 5.

2 Related Work

Smartphones malware detection techniques based on system traces such as system calls and system events studied actively in recent years. Several classifiers and datasets types are investigated to produce more accurate model. This comes with more memory and computational cost. Bose et al [3] introduced framework to detect mobile malwares. The framework based on Support Vector Machine (SVM) classifier and temporal logic of causal knowledge (TLCK) to represent the applications behaviors. The application behaviors are constructed from several data: system events, resource accesses and API calls in Symbian OS. The authors studied only 25 distinct malicious behaviors. Shabtai et al [4] proposed a framework using machine learning classifiers to detect new malwares. These classifiers are: k-means, Logistic Regression, Histograms, Decision Tree, Bayesian Networks and Naïve Bayes. The normal behavior is constructed from various system metrics like CPU consumption, number of sent packets through Wi-Fi, number of running processes and battery level. Xie et al [5] adopted Hidden Markov Model (HMM) to build normal behavior. The training dataset is a combination of user inputs and applications system calls. HMM classifier is powerful, but computationally expensive which make it not suitable to limited resources environment. Zhao et al [6] presented software behavior signature framework called RobotDroid. It runs on Android OS phones. This Framework is based on Support Vector Machine (SVM) active learning algorithm. The classifier is trained on three malwares families: Plankton, DroidDream and Geinimi. The application behaviors are defined as intent issued and system resources accessed by applications. Burguera et al [7] proposed a framework named Crowdroid running on Android OS platform. The proposed framework collects system call traces, organizes them in bag of system call representations and applies k-means algorithm to classify them into two categories: normal and malware. The shortcoming of this technique is the fact that the system always divides the system call dataset into two classes even if there is no malware. Amamra et al [8] proposed hybrid machine learning classifiers using stacking method to improve the classification process. The classifiers trained and tested with system call traces. These system calls are collected from 100 normal applications and 90 malwares. Hybrid classifiers improve accuracy performance with high computational cost. Pathak et al [9] proposed a new system calls based on power modeling framework. This framework consists of two main components: (1) Finite State Machines (FSM) to model the power states and state transitions of each component as well as the whole smartphone, (2) testing application suite which leverages the domain knowledge of system calls to systematically uncover the FSM transition rules. Buennemeyer et al [10] proposed B-SIPS (Battery-Sensing Intrusion detection Protection System). B-SIPS monitors the power consumption of Bluetooth and Wi-Fi communication channels to detect intrusion activities. The anomalous activities are detected if the power consumption level exceeds the system's dynamic threshold value. Power based malware detection technique detects only malwares have an impact on device power level, whereas many malwares have normal power level consumption.

3 System Calls Representations

Roughly speaking, a system call trace is a chunk of information. That information is formatted and organized in a dataset D as shown in Fig 1. The dataset D has a specific format that expresses how the data is organized. This format is called dataset representation and it is used during training as well as detection phases of machine learning classifiers.

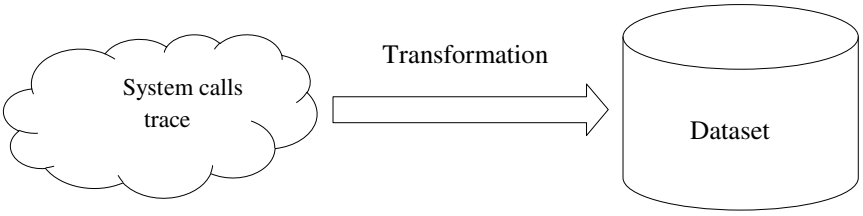

Fig. 1. System call trace transformation to Dataset

Given the dataset D, the purpose of the machine learning algorithms is to find a classifier C: D $\longrightarrow$ {normal, malicious} that maximizes accuracy and minimizes false positive rate. Dataset representation has an impact on the accuracy and false positive rate of the machine learning classifier.

For harmony definitions of the three system calls representations, we consider the set of system calls $\Sigma = \{s_1, s_2, s_3 \ldots s_m\}$, where *m* is the number of system calls of operating system. Let S_i be finite sequence of system calls and $|S_i|$ is the length of the sequence. Let Σ^* be the set of all possible finite sequences of system calls, $S_i \in \Sigma^*$. N is the number of applications (normal and malware) used in training and test phases.

3.1 Successive System Calls Representation

This representation considers the sequential order information of system calls in sequence. Formally, this representation can be defined as follow:

Dataset D is a set of finite sequences of successive system calls generated by different executed applications. The dataset D can be defined formally as: $D = \{<S_i, T_i> | S_i \in \Sigma^*, Ti \in \{\text{normal, malicious}\}\}$ where S_i is a finite sequence of system calls generated by an application i and T_i indicates whether this application is normal or malicious. Fig 2 shows an example of successive system calls representation.

syscall_983045, prctl, mmap2, gettid, brk, getpriority, futex, clock_gettime, clock_gettime, futex, open, ioctl, ioctl, mmap2, close, syscall_983042, brk, brk, futex, futex, futex, futex, mmap2, mmap2, ioctl, sigprocmask, syscall_983042, futex, futex, sigprocmask, syscall_983045, prctl, mmap2, gettid, brk, getpriority, futex, clock_gettime, clock_gettime, …, normal

Fig. 2. Successive System Calls

The memory complexity of successive system calls representation is depends on the number of sequences S_i in the database (N) and the total length of sequences. The memory complexity is $O(N^* \sum_{i=1}^{N} |S_i|)$. The total length of sequences is big number, therefore this representation is costly.

3.2 Bag of System Calls Representation

This representation disregards the ordering information of sequential system calls. Only the frequency of system calls in the sequence is maintained. Formally, the representation can be defined as follow:

Let dataset D be the set of features X. The feature X_i is defined as a list $X_i = <n_1, n_2, n_3, \ldots, n_m>$, where $m = |\Sigma|$ and n_j is the number of occurrence of a system call s_j in the sequence S_i.

Fig 3 illustrates bag system call representation of a normal application. Each number in the sequence represents the frequency of a system call in application trace. For example, the numbers 5,0,0,10,75,…. correspond to frequency of the following system calls: fstat64, setgroups32, setgid32, setuid32, getuid32 respectively. The last attribute in figure 3 indicates the sequence for normal application.

5,0,0,10,75,0,0,0,0,0,01,1,3,0,0,4,0,0,0,0,203,25,62,58,0,0,0,0,0,0,0,0,0,0,0,0,35,101,0,0,0,0,0,0,0,0,0,0,36,0,0,0,0,0,4,0,10,0,0,0,0,0,0,0,0,0,0,0,513,0,0,0,0,0,0,0,0,97,6072,1617,142,360,0,1,0,0,0,0,0,0,0,0,90,0,0,0,00,0,0,0,0,0,0,0,108,0,0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10,0,0,0,0,85,353,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,00,0,0,0,0,2575,40,0,0,normal

Fig. 3. Bag System call Representation

The memory complexity of bag of system calls representation is $O(N^*|\Sigma|)$, where N is the number of sequences and $|\Sigma|$ is number of system calls. $|\Sigma|$ is a small number. For example, Linux 2.6 has 326 system calls [11], then $|\Sigma| = 326$. The cost of this representation is much less than successive system calls representation cost because $|\Sigma|$ is very small than $\sum_{i=1}^{N} |S_i|$.

3.3 ℓ-Patterns Frequency of System Calls Representation

This representation combines features of the two previous representations. Pattern-frequency representation regards the successive order information of system calls in a short patterns, and regards the frequency of those patterns in a sequence. Formally, the representation can be defined as follow:

Let P be the set of all possible patterns of length ℓ, $P = \{p_1, p_2, p_3 \ldots p_r\}$, where $r = |\Sigma^{\ell}|$. Dataset D is set of features Y. The feature Y_i is defined as an ordered list $Y_i = <n_1, n_2, n_3, \ldots, n_r>$, where n_j is the number of occurrence of a pattern p_j in the sequence S_i.

Fig 4 illustrates 3-pattern frequency representation of a normal application. Each number represents the frequency of pattern of 3 system calls length. For example, the numbers 15,0,30,219,1,…. represents the following: the pattern: (open, fstat64,

mprotect) is repeated 15 times, the pattern: (close, close, close) is repeated 0 time, the pattern: (mmap2, mprotect, clone) is repeated 30 times, the pattern: (futex, futex, gettimeofday) is repeated 219 times, and the pattern: (access, access, mkdir) is repeated 1 time. The last attribute in figure 4 indicates the sequence for normal application.

15,0,30,219,1,0,0,0,0,0,0,10,0,113,0,0,4,0,0,0,0,2,125,622,58,0,0,0,0,20,0,0,0,0,0,0,0,
35,101,0,0,0,0,0,0,0,0,0,0,36,0,0,0,0,0,432,0,10,0,0,0,0,0,0,0,0,33,0,0,0,513,0,0,0,0,0,
0,0,0,0,6002,160,1402,360,0,1,0,0,0,0,0,0,0,0,90,0,0,0,00,0,10,0,0,0,0,0,108,0,0,0,8,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10,0,0,0,0,385,153,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0
,25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,3…,normal

Fig. 4. 3-Patterns Frequency Representation

The memory complexity of ℓ-patterns frequency of system calls representation is $O(N*|\Sigma^{\ell}|)$, where N is the number of sequences and $|\Sigma^{\ell}|$ number of possible patterns. For short patterns ($\ell = 2$ or 3) and $|\Sigma| = 326$, $|\Sigma^{\ell}|$ is not big number. This representation consumes memory little more than bag of system calls representation and much less than successive system calls representation.

4 Experiments and Results

In this section we examine and evaluate the influence of dataset representations on the performance of various machine learning classifiers. We measure the performance of the machine learning classifiers using two standard metrics: Accuracy rate and False Positive (FP) rate. Table 1 presents the confusion matrix. Confusion matrix holds information about actual and predicted classes generated by classifier. The performance of classifier is measured using the information in the matrix.

Table 1. Confusion Matrix

		Predicted Class	
		Normal	Malware
Actual Class	Normal	True Positive (TP)	False Positive (FP)
	Malware	False Negative (FN)	True Negative (TN)

In order to compare the performance of various machine learning classifiers, we employ two standard metrics: false positive rate and accuracy rate.

- False positive (FP) is the quantity of misclassifying normal behaviours as malicious.

$$\text{False Positive Rate (FP)} = \frac{\text{FP}}{\text{TP} + \text{FP}}$$

- Accuracy rate is the rate of correct predictions over the whole dataset.

$$\text{Accuracy Rate} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}$$

We experiment the different dataset representations on the following machine learning algorithms: Support Vector Machine (SVM), Naïve Bayes (NB), Logistic Regression (LR) and Decision Tree (DT). Those classifiers are obtained from the WEKA machine learning visual package [12].

The dataset represents a large set of system-call traces of benign and malicious applications. The normal applications are the top 100 popular free applications in the official Android applications store (Android market) [13]. The 90 available Android real malwares are download from [14]. All applications are installed and executed on HTC Dream device running Android OS 2.2. The number of sequences in the dataset N = 190 (100 normal applications + 90 malwares). The length of sequences varies from applications to other. The shortest sequence has 10000 system calls and the longest sequence has 60000 system calls.

Support Vector Machine (SVM) classifier classifies data by determining a set of support vectors, which are members of the set of training inputs that outline a hyper plane in the feature space [15]. Table 2 represents the support vector machine classifier accuracy and false positive rates using different data representations. Pattern-frequency representation of 2 system calls length and 3 system calls length outperforms with 100% accuracy rate and 0% false positive rate, whereas successive system calls representation has better performance than bag of system calls with 95.10% accuracy rate and 6.10% false positive rate.

Table 2. Support Vector Machine Classifier Performance

Dataset Representation	*Accuracy rate*	*FP rate*
Successive system calls	95.10%	6.10%
Bag of system calls	92.50%	8.50%
2-Pattern frequency	100%	0%
3-Pattern frequency	100%	0%

Naïve Bayes (NB) Classifier is the simplest form of Bayesian network. It is based on Bayes theorem with heavy independence assumptions [16]. Table 3 shows the performance of naïve Bayes classifier using different system calls representations. The 3-pattern and 2-pattern frequency representation has the best performance (97.10% accuracy rate and 3.10% false positive rate). Bag of system calls representation has the worst performance (91.30% accuracy and 9% false positive rate).

Table 3. Naïve Bayes Classifier Performance

Dataset Representation	*Accuracy rate*	*FP rate*
Successive system calls	94.10%	6.30%
Bag of system calls	91.30%	9%
2-Pattern frequency	97.10%	3.10%
3-Pattern frequency	97.10%	3.10%

Logistic Regression (LR) Classifier is a discriminative model. It is based strongly on the logistic function [16]. Table 4 illustrates the impact of dataset representation on the performance of logistic regression classifier. The 3-pattern frequency representation has the highest performance. It has the higher accuracy rate (100%) and the lower false positive rate (0%). The 2-pattern frequency representation still performs better than the two classical representations with 97.10% accuracy rate and 3.10% false positive rate. Bag of system calls representation has the worst performance (92.30% accuracy and 8% false positive rate).

Table 4. Logistic Regression Classifier Performance

Dataset Representation	*Accuracy rate*	*FP rate*
Successive system calls	96%	4.10%
Bag of system calls	92.30%	8%
2-Pattern frequency	97.10%	3.10%
3-Pattern frequency	100%	0%

Decision Tree (DT) Classifier is built during training phase. The nodes represent attributes, the edges represent the possible attributes values, and leafs represent the classes. The classification start by root node and move down in decision tree relative to attributes values till reach leaf [17]. Table 5 demonstrates decision tree classifier influenced by dataset representations. The classifier has better performance with pattern frequency system calls representation than other dataset representation. The 3-pattern frequency representation has the best performance with accuracy rate 98.20% and false positive rate 2.10%. Successive system calls representation has the worst performance with accuracy rate 92.10% and false positive rate 9.40%.

Table 5. Decision Tree Classifier Performance

Dataset Representation	*Accuracy rate*	*FP rate*
Successive system calls	92.10%	9.40%
Bag of system calls	96%	4.10%
2-Pattern frequency	97.10%	3.10%
3-Pattern frequency	98.20%	2.10%

In summarizing of the above the results, the dataset representation has an important impact on the performance of machine learning classifiers. With the same dataset size and type, we have different accuracy rate and false positive rate for the same classifier. The classification process is based on two classification properties: system calls successive property and system calls frequency property. The SVM classifier has low performance with bag of system calls representation with 92.50% accuracy rate and 8.50% false positive rate which means SVM classifier is more sensitive to successive

classification property than frequency classification property. Naïve Bayes classifier and Logistic Regression have the same behavior as SVM. Their performance on successive dataset representation is better than the performance on frequency dataset representation. However, Decision Tree classifier is more sensitive to frequency property than successive property; its performance with bag of system calls representation (96% accuracy and 4.10% false positive rate) is better than successive system calls representation (92.10% accuracy and 9.40% false positive rate). The pattern frequency representation has the better performance over all classifiers because this representation holds the both classification properties: frequency property and the successive property of short sequence. SVM classifier reaches its idle performance at two system calls pattern length. Three system calls pattern is the optimal length for Logistic Regression classifier. The LR classifier reaches its best performance with this length. Decision Tree and Naïve Bayes classifier may require longer patterns length to achieve their idle performance. Longer patterns have more successive and frequency information; therefore it is more discriminative in classification process. However, longer patterns require more memory and computation resources. SVM with 2-pattern frequency representation is the best combination of classifier and representation because this combination reaches the idle performance with lowest cost. Table 6 orders the combination (classifier-pattern frequency dataset representation) according to performance per cost. The worst performance of pattern frequency representations is 97.10% accuracy rate and 3.10% false positive rate. However, this performance is better than the best performance of other representations.

Table 6. Classifier-Dataset Representation Order per Performance

Classifier-Dataset Representation	Accuracy rate	FP rate
SVM (2-Pattern frequency)	100%	0%
SVM (3-Pattern frequency)	100%	0%
LR (3-Pattern frequency)	100%	0%
DT (3-Pattern frequency)	98.20%	2.10%
LR(2-Pattern frequency)	97.10%	3.10%
DT(2-Pattern frequency)	97.10%	3.10%
BN (2-Pattern frequency)	97.10%	3.10%
BN (3-Pattern frequency)	97.10%	3.10%

5 Conclusion

Dataset representation has a potential impact on the performance of machine learning classifiers. It increases the accuracy rate and decreases the false positive rate. The experiments results affirm the performances of classifiers are improved significantly with same data quantity and type, but with using a new dataset representation. The 3-pattern frequency representation has the best performance over all experimented classifiers; it reaches the idle performance of SVM classifier and logistic regression classifier (100% accuracy rate and 0% false positive rate). The cost of improving the

performance by using dataset representation factor is very low relative the other factors: normal behavior model, dataset size and dataset type. Because dataset representation keeps the same dataset type and size, the required memory space is few extra kilobytes. The process of preparing dataset representations is not computationally expensive. Bag of system calls representation has the lowest memory cost and the lowest performance. Successive system calls representation has the highest memory cost and not the highest performance. Short pattern system calls representation has a few extra memory cost than bag of system calls representation, but it improves the classifiers performance significantly. The frequency property of system calls in sequence is important in classification process and the successive property of system calls in sequence is also important in the process of classification. However, the pattern frequency property provides advantage by holding the two properties which leads to better classification process. A new factor improves the performance of machine learning classifiers with low computation and memory cost is an important step to implement lightweight machine learning algorithms on Smartphones device. We will study this approach in the future work.

References

1. Amamra, A., Talhi, C., Robert, J.-M.: Performance Evaluation of Multi-pattern Matching Algorithms on Smartphone. In: BWCCA, Victoria, BC, Canada, pp. 329–334 (2012)
2. Amamra, A., Talhi, C., Robert, J.-M.: Smartphone Malware Detection: From a Survey Towards Taxonomy. In: Malware, Fajardo, Puerto Rico, USA, pp. 89–96 (2012)
3. Bose, A., Hu, X., Shin, K.G., Park, T.: Behavioral detection of malware on mobile handsets. In: MobiSys, Breckenridge, CO, USA, pp. 225–238 (2008)
4. Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., Weiss, Y.: Andromaly: a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems 38, 161–190 (2012)
5. Xie, L., Zhang, X.: pBMDS: a behavior-based malware detection system for cellphone devices. In: Third ACM Conference on Wireless Network Security, Hoboken, NJ, USA, pp. 37–48 (2010)
6. Zhao, M.: RobotDroid: A Lightweight Malware Detection Framework on Smartphones. Journal of Networks 7(4) (2012)
7. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: Behavior-Based Malware Detection System for Android. In: Workshop on Security and Privacy in Smartphones and Mobile Devices 2011, Chicago, USA (2011)
8. Amamra, A., Talhi, C., Robert, J.-M., Hamiche, M.: Enhancing Smartphone Malware Detection Performance by Applying Machine Learning Hybrid Classifiers. In: Kim, T.-H., Ramos, C., Kim, H.-K., Kiumi, A., Mohammed, S., Ślęzak, D. (eds.) ASEA/DRBC 2012. CCIS, vol. 340, pp. 131–137. Springer, Heidelberg (2012)
9. Pathak, A., Hu, Y.C., Zhang, M., Bahl, P., Wang, Y.-M.: Fine-grained power modeling for smartphones using system call tracing. In: EuroSys 2011, Salzburg, Austria, pp. 153–168 (2011)
10. Buennemeyer, T.K., Nelson, T.M., Clagett, L.M., Dunning, J.P., Marchany, R.C., Tront, J.G.: Mobile Device Profiling and Intrusion Detection using Smart Batteries. In: HICSS 2008, Waikoloa, Hawaii, pp. 1–10 (2008)

11. Organization, L.K. Linux 2.6 System calls Table (2013), https://www.kernel.org/pub/linux/kernel/v2.6/ (cited 2013)
12. Waikato, U.o. WEKA, http://www.cs.waikato.ac.nz/ml/weka/ (cited 2012)
13. Google. Android Market, https://play.google.com/store (cited 2012)
14. Mobile, B.C. Mobile Malwares (2012), http://contagiominidump.blogspot.ca/
15. Mukkamala, S., Janoski, G., Sung, A.: Intrusion detection using neural networks and support vector machines. In: International Joint Conference on Neural Networks, New Mexico, USA, pp. 1702–1707 (2002)
16. Smola, A., Vishwanathan, S.V.N.: Introduction to Machine Learning. Cambridge University Press, United Kingdom (2008)
17. Ben Amor, N., Benferhat, S., Elouedi, Z.: Naive Bayes vs Decision Trees in Intrusion Detection Systems. In: Symposium on Applied Computing, Nicosia, Cyprus, pp. 420–424 (2004)