A Probabilistic Diffusion Scheme for Anomaly Detection on Smartphones

Tansu Alpcan[1,⋆], Christian Bauckhage[2], and Aubrey-Derrick Schmidt[3]

[1] Deutsche Telekom Labs., Technical Univ. Berlin, 10587 Berlin, Germany
alpcan@sec.t-labs.tu-berlin.de
[2] University of Bonn and the Fraunhofer IAIS
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
christian.bauckhage@iais.fraunhofer.de
[3] DAI-Labor, Technische Universität Berlin, 10587 Berlin, Germany
aubrey-derrick.schmidt@dai-labor.de

Abstract. Widespread use and general purpose computing capabilities of next generation smartphones make them the next big targets of malicious software (malware) and security attacks. Given the battery, computing power, and bandwidth limitations inherent to such mobile devices, detection of malware on them is a research challenge that requires a different approach than the ones used for desktop/laptop computing. We present a novel probabilistic diffusion scheme for detecting anomalies possibly indicating malware which is based on device usage patterns. The relationship between samples of normal behavior and their features are modeled through a bipartite graph which constitutes the basis for the stochastic diffusion process. Subsequently, we establish an indirect similarity measure among sample points. The diffusion kernel derived over the feature space together with the Kullback-Leibler divergence over the sample space provide an anomaly detection algorithm. We demonstrate its applicability in two settings using real world mobile phone data. Initial experiments indicate that the diffusion algorithm outperforms others even under limited training data availability.

Keywords: Anomaly detection, mobile security, machine learning.

1 Introduction

Next generation smartphones and mobile devices (e.g. Internet tablets) offer advanced capabilities beyond a typical mobile phone and often provide general-purpose computing functionality. Although some of the smartphone manufacturers are ahead of traditional PC systems in terms of security through advanced sandboxing models, it is only a matter of time for the malware authors exploit the underlying weaknesses of the respective platforms or resort to social engineering to circumvent these measures. Furthermore, some of the recently emerging platforms forsake such sandboxing schemes in favor of open computing, which unfortunately brings well-known security problems with it. As a consequence of the underlying architecture of current computing systems

⋆ Research supported in part by Deutsche Telekom Laboratories.

P. Samarati et al. (Eds.): WISTP 2010, LNCS 6033, pp. 31–46, 2010.

and due to its social underpinnings, security will continue to pose significant challenges in the mobile domain [9].

Mobile computing platforms such as smartphones have unique features that make the task of detecting security-related problems even more challenging than desktop systems. Severe limitations on battery life, computing power, and available bandwith (and its cost) reduce the effectiveness of signature-based detection schemes as a result of their high bandwidth usage due to frequent signature updates and battery cost due to computational requirements. Furthermore, signature-based detection is mainly centralized, non-automatic, and not very scalable. Many security vendors acknowledge the problems with this decades-old methodology and search for new techniques to address the problem of malware detection in a more efficient, flexible, and scalable manner [20].

Anomaly-based detection has the potential of automated and scalable detection of never before seen malware, also known as *zero-day attacks*. It can be seen as a form of binary classification problem in machine learning. Consequently, anomaly-based detection algorithms usually require a training (learning) phase before the actual detection (monitoring). Over the years, a large number of anomaly detection schemes have been proposed, most of them relying on a variety of machine learning and data mining techniques [19, 25, 26]. Decentralized malware detection schemes as well as deployment of filtering mechanisms have been studied in [6, 18, 24, 4, 5]. A game theoretic model, which takes into account the attacker behavior in decision making, has been presented in [2]. Other approaches in the existing literature include semantic models, grammars, or rule bases [8]. A detailed survey of the field can be found in [14]. Recently, malicious software detection in smartphones and mobile devices has been a topic of increasing interest [27, 7, 23, 21].

Despite the extensive research efforts, an inherent limitation of anomaly-based detection has been its high false alarm rate. It is mainly due to the difficulty in distinguishing between actual malware and mere "unusual behavior" of the system. The detection algorithm is expected to be sensitive enough to detect tiny deviations yet precise enough to separate abnormal behavior, which may indicate existence of malware, from previously observed daily usage patterns. The *base-rate fallacy* captures the essence of this problem. Even if the detection scheme achieves low false-negative and false-positive rates, the legitimate network traffic is much higher in volume than the malware traffic that the number of false alarms is substantial [3]. In other words, most of the time detection process is similar to searching for a needle in a haystack, and current methods are often not good enough to find "the needle" without processing a lot of "hay".

Many classical detection and estimation techniques rely on probabilistic models based on well-known random processes with known characteristics. However, such statistical descriptions are most of the time not applicable to malware detection. The problem of malware detection is further complicated by its inherent non-convex nature and the limited availability of normal usage data for training the algorithms. The availability of attack information is often even more limited. In addition, the monitoring has to be done in near real-time and should not bring excessive overhead in terms of resource usage. All these requirements clearly pose a significant challenge and motivate our research on advanced detection techniques.

We present, in this paper, a behavior-based approach to anomaly-based malware detection. Our method is based on the study of higher order relationships between data samples that stem from smartphones daily usage patterns. The basic idea is to model dependencies of samples and features by means of a bipartite graph which then serves as the domain of a Markov process. On the one hand, as the algorithm involves mappings between the feature and the sample space, it can be seen as another instance of the *kernel trick* well-known in machine learning. On the other hand, it shares some of the characteristics of the famous Google pagerank algorithm. The algorithm is applied to two separate data sets obtained from smart mobile phones during normal daily usage. In one case, a –rather harmless– malware (Trojan) is activated on the mobile phone after the training period that sends an SMS message whenever the user presses button 2 on the keypad. In the other case, we simulate a malware that exhibits symptoms similar to *Viver* and *Beselo* Trojans. Both data sets, which contain real phone useage data, have allowed us to conduct experiments and test the performance of our algorithm in a realistic setting and investigate multiple scenarios involving different malware.

The rest of the paper is organized as follows: in the next section we present the model and our algorithm. Section 3 reports and discusses experimental results. The paper concludes with remarks of Section 4.

2 Detection Algorithm

In this section, we present our computational model and algorithm for anomaly detection. For the rest of the paper, we adopt the following notational conventions. Vectors are column vectors and the ith component of a vector $\boldsymbol{v}$ is denoted by v_i. Correspondingly, the (i, j) entry of a matrix $\mathbf{M}$ is denoted by M_{ij}. Finally, $\mathbf{I}$ denotes the identity matrix.

2.1 Model

The basic idea behind our approach to anomaly detection is to consider diffusion processes over a graph. Diffusion processes as a means for computing similarities on manifolds or among the vertices of graphs have recently been studied by several authors [16, 1, 15, 28]. Given an unstructured set of *observations* or *feature vectors*, these approaches compute a matrix that represents *local* structures in the data by means of the distances between each data point and its k nearest neighbors. Once the adjacency matrix is available, *global* similarities among the data points can be determined from graph traversal.

The approach we present in this paper differs from the work in [16, 1, 15, 28] in that it does not rely on geometry-based adjacency matrices. Instead, we assume a bipartite graph $G = (N, E)$ as shown in Figure 1. Its set of nodes N is partitioned such that $N = F \cup O$ and $F \cap O = \emptyset$ and for the edges we have $E \subseteq O \times F \cup F \times O$. We assume that $F = \{f_1, \ldots, f_m\}$ represents a set of observable features that allow for characterizing a given set $O = \{o_1, \ldots, o_n\}$ of acceptable patterns of cell phone usage. In other words, we assume that n feature vectors $\boldsymbol{f} \in \mathbb{R}^m$ representing "normal" behavior are available to train our malware detection system.

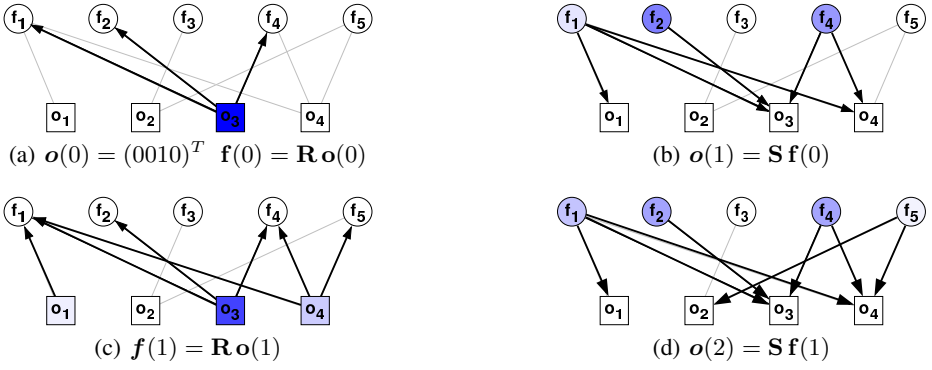


Fig. 1. Bipartite graph model of relations between observations o_j and their features f_i. 1(a) Index vectors, e.g. $\boldsymbol{o}(0) = [0010]^T$, indicate individual observations, e.g. o_3. Individual observations or mixtures of observations correspond to a feature distribution $\boldsymbol{f}(0)$. 1(b) A feature distribution $\boldsymbol{f}(0)$ diffuses to an updated distribution over the observations which is represented by a vector $\boldsymbol{o}(1)$. 1(c) and 1(d) The diffusion process continues and provides a ranking of the observations with respect to the initial one. If the stochastic matrices $\mathbf{R}$ and $\mathbf{S}$ are defined as in the text, diffusion processes like this are guaranteed to converge to a unique limiting distribution $\boldsymbol{o}^*$.

In addition, we require the nodes in F to be labeled. In a slight abuse of notation, we represent a given labeling by means of a vector $\boldsymbol{f} = [f_1, \ldots, f_m]^T$. Furthermore, we normalize the feature vectors $\boldsymbol{f}$ such that they become stochastic vectors whose components sum 1. This normalization turns vectors into distributions of features and is a rather common procedure in fields such as pattern recognition, data mining,or information retrieval, especially if the features are given in form of histograms [10, 13, 22]. Similarly, we assume the nodes in O to be labeled and again identify a given labeling with a stochastic vector $\boldsymbol{o} \in \mathbb{R}^n$. In this way, the ith prototypic pattern or observation o_i will be indexed by the vector $\boldsymbol{o} = \mathbf{e}_i$, i.e. the ith standard basis vector in $\mathbb{R}^n$. Note that this modeling approach also allows for representing *mixtures of prototypes*.

The intuition behind the formulation above is that, even though there are no immediate relations (i.e. no edges) among the individual observations in O, the model nevertheless allows for determining partial orders. The key idea is to understand their relations as the outcome of a stochastic diffusion process over the bipartite graph.

If the normalized feature vectors $\boldsymbol{f}_i$ that characterize the observations o_i are stored in a column stochastic $m \times n$ matrix $\mathbf{R} = \left[\mathbf{f}_1 \ldots \mathbf{f}_n\right]$, its elements R_{ij} can be interpreted as the probability or extent of an occurrence of a feature f_i given a prototype o_j

$$R_{ij} = p(f_i|o_j). \tag{1}$$

Applied to a mixture of observations $\boldsymbol{o}$, the matrix $\mathbf{R}$ realizes a probabilistic mapping $\boldsymbol{f} = \mathbf{R}\mathbf{o}$ from the set of observations to the set of features.

Given the transition matrix $\mathbf{R}$, we can deduce a probabilistic mapping $\mathbf{S} = \mathbf{R}^T\mathbf{D}^{-1}$ that maps a distribution of features to a distribution of observations (see Fig. 1). Here $D_{jj} = \sum_i A_{ij}^T$ and $D_{ij} = 0$ is used to normalize the columns of $\mathbf{R}^T$. Note that

$\mathbf{S}$ maps an arbitrary distribution of features o the distribution of prototypes that most likely explains the observation, which follows from simple Bayesian reasoning.

2.2 Stochastic Diffusion Algorithm

For diffusion-based classification, we consider dynamic processes over the prototypes. Given a prototype distribution $\boldsymbol{o}(t)$ at time t, an updated distribution results from

$$\mathbf{o}(t+1) = \mathbf{Sf}(t) = \mathbf{SRo}(t) \equiv \mathbf{Ho}(t). \tag{2}$$

Some straightforward algebra shows that the $n \times n$ matrix $\mathbf{H}$ introduced above is a doubly stochastic matrix whose rows columns and rows both sum to 1. It is square, non-negative, and in accordance with the Perron-Frobenius theorem its eigenvalues λ satisfy $|\lambda| \leq 1$. The process defined by $\mathbf{H}$ is therefore a Markov process over O. Consequently, even though our model does not assume any direct relations among the $o_i \in O$, it can rank them: if we assume an initial distribution $\boldsymbol{o}(0)$ with only a few non zero entries, after t iterations, the probabilities in $\boldsymbol{o}(t) = \mathbf{H}^t\mathbf{o}(0)$ will be higher for observations more closely related to the initially indexed elements of O and less high for less closely related ones (see Fig. 1).

In order to produce similarity rankings where all $o_i \in O$ are properly taken into account, we would have to compute $\boldsymbol{o}^* = \lim_{t\to\infty} \mathbf{o}(t) = \mathbf{H}^t\mathbf{o}(0)$. However, the bi-stochastic nature of $\mathbf{H}$ forces $\boldsymbol{o}(t)$ to converge to the uniform distribution where all elements equal $\frac{1}{n}$. We therefore consider a modification where we assume the initial distribution $\boldsymbol{o}(0)$ to be a steady source of probability mass which is constantly fed into the process. The update rule for prototype distributions then becomes

$$\mathbf{o}(t+1) = \alpha\mathbf{Ho}(t) + (1-\alpha)\mathbf{o}(0). \tag{3}$$

The Perron-Frobenius theorem guarantees the convergence of this process, too. Its stationary distribution $\boldsymbol{o}^*$ depends on $\boldsymbol{o}(0)$ and can be shown to amount to

$$\mathbf{o}^* = (1-\alpha)\big[\mathbf{I} - \alpha\mathbf{H}\big]^{-1}\mathbf{o}(0). \tag{4}$$

Therefore, given an arbitrary initial distribution $\boldsymbol{o}(0)$ that represents a single observation, we can immediately determine the corresponding stationary distribution and the ranking it implies.

It is interesting to note that the matrix in equation (4) constitutes a diffusion kernel [16]. In fact, from the derivation, we recognize another instance of the *kernel trick*. The similarities among vectors $\boldsymbol{o} \in \mathbb{R}^n$ that are contained in $\mathbf{H} = \mathbf{SR}$ result from mapping the vectors back and forth to a –possibly larger– space $\mathbb{R}^m$.

Diffusion kernels for computing similarities on manifolds or graphs have recently found much attention [16,1,15,28]. Usually, they are derived similarity matrices which, in turn, are derived from geometry-based adjacency matrices. They thus require to introduce distance measures in order to determine adjacencies and similarities and it is left to the user to decide which metric best fits his or her practical needs. Our approach avoids such overhead. By basing our derivation on a bipartite graph model and due to the Bayesian nature of the mapping between prototypes and features, the resulting

matrix $\mathbf{H}$ is a stochastic matrix that allows for a concise interpretation of the ranking procedure in terms of a Markov process. Moreover, our model also provides a latent semantic interpretation of the diffusion process. The transition probabilities H_{ji} can be explained as the effect of hidden latent variables:

$$H_{ji} = p(o_j|o_i) = \sum_k p(o_j|f_k)\, p(f_k|o_i). \tag{5}$$

Ranking on manifolds has already been applied in systems for document and image retrieval [12]. In fact, equation (3) corresponds to the iterative version of the adjusted page rank procedure used by the Google search engine [17]. However, to the best of our knowledge, all such systems consider diffusion processes over adjacency graphs that represent local neighborhoods similar to the way discussed above. As these approaches do not distinguish between features and prototypes, their applicability for classification and prediction is limited. Either, the class label of a new feature vector has to be decided according to its k nearest neighbors on the manifold, or a new transition matrix would have to be computed that also regards the new vector. Our bipartite graph model, on the other hand, distinguishes between features and prototypic observation and immediately allows us to devise classifiers that take into account the whole set of given prototypes. This can be done in just a single step and does not require to update the given transition matrix.

2.3 Diffusion-Based Anomaly Detection

Let $\boldsymbol{f}$ denote a novel measurement of features which is normalized such that its components sum to 1. A stationary distribution that characterizes its relation to all of the prototypic observations o_i simply results from

$$\mathbf{o}^* = (1-\alpha)\big[\mathbf{I} - \alpha\mathbf{H}\big]^{-1}\mathbf{S}\mathbf{f}. \tag{6}$$

If we were dealing with a classification problem for which a labeled training set was available, a class label ω for the unknown observation could then, for instance, be obtained from $\omega(\mathbf{f}) = \mathrm{argmax}_k\{\omega(u_k^*)\}$. However, in anomaly-based malware detection the nature of a novel observation $\boldsymbol{f}$ must be decided from a given set of normal observations only. Since there is thus only one type of class label, we apply the stationary prototype distributions resulting from our method in a divergence-based classifier.

The algorithm (summarized in Fig. 2) first computes the stationary distribution that belongs to the novel observation $\boldsymbol{f}$ and then compares it to the stationary distributions that result from the prototypes available for training. In our current implementation, the differences between distributions are determined using the Kullback-Leibler divergence. Finally, the minimal divergence thus estimated is taken as a measure of how the current observation deviates from the manifold of prototypes. Classification is obtained through thresholding with an adjustable parameter. In contrast to the usual approaches in anomaly-based malware detection, our method does not compute distances between feature vectors and classification is not carried out in the feature space at all. In a series of experiments presented in the next section, the dual treatment of features and samples has been observed to outperform classical feature-based approaches.

Input: an observation $\boldsymbol{f}$ and a set $\{\mathbf{o}^{1*}, \ldots, \mathbf{o}^{n*}\}$ of stationary prototype distributions derived from a set $O = \{o_1, \ldots, o_n\}$ of acceptable prototypes
Output: a divergence measure d characterizing how far off $\boldsymbol{f}$ is from the manifold of prototypes

compute the stationary prototype distribution w.r.t. $\boldsymbol{f}$
$\boldsymbol{o}^* = (1-\alpha)\left[\mathbf{I} - \alpha\mathbf{H}\right]^{-1}\mathbf{S}\mathbf{f}$
for $i = 1, \ldots, n$
compute the Kullback-Leibler divergence
$D(i) = D_{KL}(\mathbf{o}^{i*} \| \mathbf{o}^*) = \sum_k o_k^{i*} \log \frac{o_k^{i*}}{o_k^*}$
endfor
determine the minimal divergence $d = \mathrm{argmin}_i\{D(i)\}$

Fig. 2. Algorithm for anomaly detection w.r.t a set of normal prototypes that have been submitted to stochastic diffusion

3 Experiments

In this section, we present experiments conducted on two different data sets. Altough both sets contain real phone usage data obtained from a real world setting, they have different characteristics and allow us to explore different aspects of the malware detection problem. In each case, the performance of the algorithm introduced in the previous section is compared to several baseline approaches and schemes.

3.1 Smartphone System Data Set and Experiments

The smartphone system data set is obtained from a monitoring client installed on a Nokia E61 smartphone. This client sends numerical feature vectors to a remote server that stores them in a database. The client code was developed in Symbian C++ version S60 3rd with Nokia Carbide.vs and consists of the three main components: User Interface, Communication Module, and Feature Extractor. The *User Interface* can be used to change server port and address. The *Communication Module* uses SOAP webservices in order to communicate with the server. As expected, collecting and sending system level data regularly is rather expensive in terms of battery power. To prevent the rapid depletion of the power source, the monitored data is sent in bulks. The *Feature Extractor* is triggered to fetch new data every thirty seconds which is stored locally and later, upon reaching a threshold, sent to the server using the appropriate webservice [23]. This data consists of system characteristics that describe all areas of the monitored device. Some of these values are shown in Figure 3.

The recording period of the used data spanned more than a 100 hours of activity where the monitoring client was activated for about 4 hours a day which were varied between different daytimes. While the monitoring client was active, the device was used in a typical manner to make phone calls, read and write messages, to install and

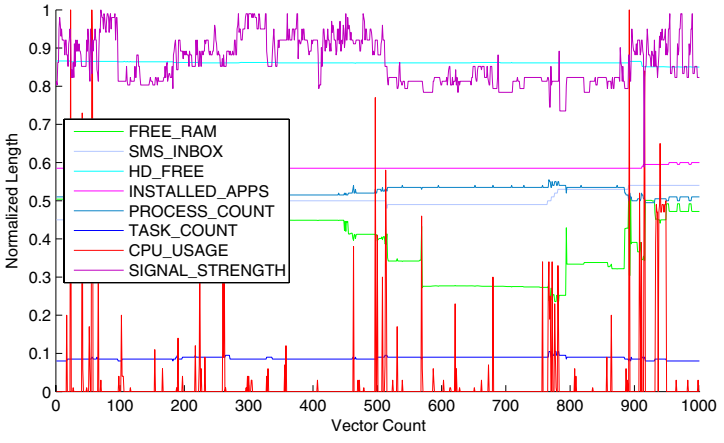


Fig. 3. Normal device usage on a Nokia E61 over a period of 1000 vectors

use applications when not idle. Figure 3 exemplifies how behaviors like these affect some of the observed features over a period of 1000 recordings where the the values have been normalized.

The malware deliberately installed on the device is a rather harmless trojan, which was chosen due to security reasons. It is activated automatically without the user of mobile phone being aware of it and sends an SMS message whenever the user presses button 2.

Figure 4 charts the outcomes of several attempts of detecting the activity of the malware in the recorded data. After dividing the data into independent sets for training and testing, the methods used to produce these results were:

(a) computing the minimal l_2 norm between the currently observed features and the feature vectors in the training set; in this rudimentary setting, distances were computed in the original feature space high dimensional feature space;

(b) an analysis of the eigenvalue spectrum of the vectors in the training set revealed that almost all the energy was contained in a six dimensional subspace; training and test samples were therefore projected to that subspace and the minimal l_2 norm between the current observation vector and the set of training prototypes was computed in the lower dimensional space;

(c) instead of the l_2 norm, we computed the Mahalanobis distances (since it takes into account higher moments, it is frequently considered the method of choice in present day anomaly detection systems [25, 26]); again, distance computation was carried out in in the low dimensional subspace;

(d) self organizing maps (SOMs) were fitted to the six-dimensional approximations of the training data; the minimal distances of test vectors were computed w.r.t. the weights of the SOM neurons;

(e) our algorithm presented in the previous section was applied; training and classification were based on feature vectors in $\mathbb{R}^{40}$.

For each of these methods, a set of 1500 feature vectors in $\mathbb{R}^{40}$ was available for training. However, methods (a), (b), (c), and (e) were trained with considerably smaller numbers of randomly sampled prototypes. For method (d), we considered corresponding small numbers of SOM neurons but used all available training data in the fitting process. For method (e), i.e. the approach proposed in this paper, the parameter α was set to 0.9.

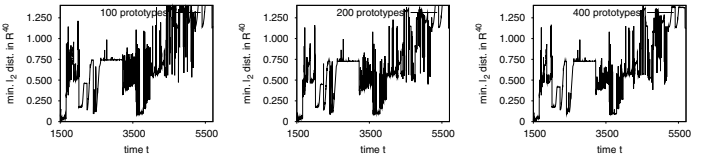
The observation period of $t \approx 1500, \ldots, 5500$ shown in Fig. 4 contains an instructive example that illustrates the benefits of the approach proposed in this paper. In this period, the trojan was active only in the interval $3400 \leq t \leq 3700$. Obviously, none of the common malware detection techniques (a) – (d) which we considered for baseline comparison was able to detect its activity. Regardless of the number of prototypes used for training, they fail to produce a coherent interpretation as to whether the mobile device's activity patterns are normal or not. Chaotic responses like these virtually rule out anomaly detection.

The algorithm, we proposed in the previous section, on the other hand, produces a clear peak at the point of activity of the malware. Figure 4(e) even shows that this effect becomes more pronounced the more training examples there are. While stochastic diffusion over a sample of 50 prototypes yields peaks of almost the same magnitude for normal and abnormal behavior alike, a growing number of training samples considerably widens the gap between peaks at points of normal and abnormal behavior (note the changing scale of the y-axis of the plots). Therefore, where anomalies are too small to be noticeable for norm-based techniques, divergence from a set of stochastically diffused training samples, still accomplishes reliable detection.

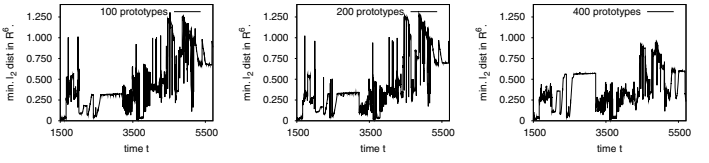
Apart from the activity of the malware, the example considered in this discussion is instructive for another reason. From regarding the plateau of peaks in Fig. 4(e) that cover the period $3700 \leq t \leq 4400$, one would suspect the device to have behaved suspiciously during that period, too. However, this was not the case. Immediately after the trojan was active, the user of the mobile phone received a multimedia message, an event that did not occur during the training period. In addition, deleting this message as well as a bunch of previously receives short text messages and a short time of lost connectivity immediately followed that event. None of these rare but unsuspicious patterns of activities was present in the training sample. This explains the considerably high divergence from the prototype distributions. However, although being rare, the stationary distributions resulting from these observations were still much closer to the prototype distributions than the distribution that resulted from the observations during the active period of the malware.

3.2 Smartphone Log Data Set and Experiments

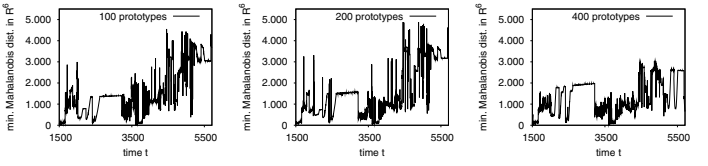
The experiment in the previous section focuses on collection and utilization of system-level features of a smartphone for malware and anomaly detection. In this second experiment, we rely on application level data for the same purposes. We use a subset of the data from the MIT reality mining project [11] which consists of smartphone call, SMS, and data communication logs collected via a special application during normal daily usage of volunteers. We use specifically the log file from a single user that covers 244 days of activity. Subsequently, we pre-process this data and generate per-day



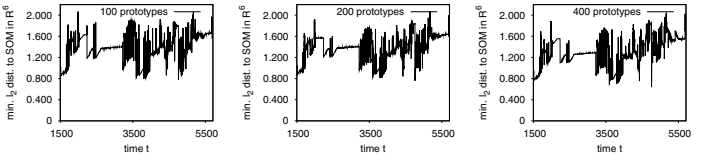
(a) min. l_2 distances of observations at time t to the set of samples of normal behavior; distances were computed among 40 dimensional feature vectors



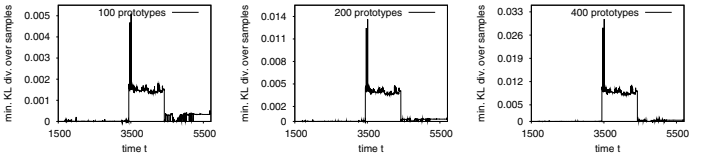
(b) min. l_2 distances of observations at time t to the set of samples of normal behavior; distances were computed among 6 dimensional feature vectors



(c) min. Mahalanobis distances of observations at time t to samples of normal behavior; distances were computed among 6 dimensional feature vectors



(d) min. l_2 distances of observations at time t to weight vectors of a SOM; distances were computed among 6 dimensional vectors



(e) min. Kullback-Leibler divergences resulting from the algorithm in Fig. 2; note the change in scale caused by the growing number of training samples

Fig. 4. Malware detection results obtained from various methods. For methods (a), (b), (c), and (e), the number of samples representing normal behavior are varied between 100 and 400; for method (d), the number of SOM neurons vary correspondingly.

Table 1. Histogram Features

Feature (in numbers per day)
Short duration calls (less than 2 min)
Medium duration calls (between 2 and 6 min)
Long duration calls (more than 6 min)
Short intervals between calls (less than 1 hour)
Medium length intervals between calls (between 1 and 3 hours)
Long length intervals between calls (more than 3 hours)
Outgoing SMS
Short periods between outgoing SMS
Medium periods between outgoing SMS
Long periods between outgoing SMS
Incoming SMS
Short periods between incoming SMS
Medium periods between incoming SMS
Long periods between incoming SMS
Short duration packet sending activities
Medium duration sending activities
Long duration sending activities
short periods between sending activities
Medium periods between sending activities
Long periods between sending activities

histograms with the following 20 entries: In Table 1, short periods or intervals refer to less than 1 hour, medium ones to between 1 and 3 hours, and long ones to more than 3 hours, respectively, whereas short duration refers to less than 2 minutes, medium one to between 2 and 6 minutes, and long one to more than 6 minutes, respectively.

The selected feature set is clearly statistical in nature, and hence, privacy-preserving to a large extend despite the algorithm running for each smartphone user individually. In other words, we are not interested in whom the user has called and when but the aggregate and high level usage characteristics. Thus, it is possible to run the algorithm on the server-side without intruding the privacy of the mobile user.

As training data we randomly choose 7, 14 and 21 days out of a training period of 44 days in order to randomize the data and avoid possible artificial regularities in the data set. After training, we inject malware symptoms into the test set which consists of 200 days of separate usage data. We specifically choose a malware that behaves similar to well-known *Viver*[1] or *Beselo*[2] Trojans. It sends out one SMS every other minute up to 20 in less than an hour but at most once per day. Then, we induce it on days 166, 169, and 171 of the test period.

In Figure 5, we show the results of various algorithms used for detecting anomalous activity in the test data spanning a period of 200 days. In the figure, the days with malware activity (166, 169, and 171) are colored red. The specific methods used are:

[1] `http://www.f-secure.com/v-descs/trojan_symbos_viver_a.shtml`
[2] `http://www.f-secure.com/v-descs/worm_symbos_beselo.shtml`

(a) computing the minimal l_1 norm between the currently observed features and the feature vectors in the training set;

(b) computing the minimal l_2 norm between the currently observed features and the feature vectors in the training set;

(c) instead of the l_2 norm, we compute the Mahalanobis distances;

(d) the algorithm presented in Section 2 is applied.

In each case, the higher the value, i.e. the distance between training prototypes representing normal behavior observed and the test sample of each day, the higher the anomaly. It is important to note that the usage data itself has a lot of natural variation due to its realism. Therefore, we observe algorithms including ours indicating these extreme deviations in addition to the ones caused by malware.

Based on the results depicted in Figure 5, we compute the precision and recall in order to quantify the performace of the algorithms. In order to do this, we resort to a simple method of choosing the top 3, 6, and 9 deviations from within the 200 day test data set and check if the malware induced days are among them[3] In this context, the *precision* measures the percentage of these deviations actually belonging to the malware set and *recall* the percentage of malware detected in the set of given deviations. Ideally, recall should be 1 and precision $\{1, 0.5, 0.33\}$ for $\{3, 6, 9\}$ preset deviations, respectively. We summarize the precision and recall values for various algorithms in Table 2. Here, we omit the l_2-based method as it gives very poor results in almost all cases. The results show that the diffusion algorithm proposed outperforms others in all cases and detects malware satisfactorily even when the training data is scarce.

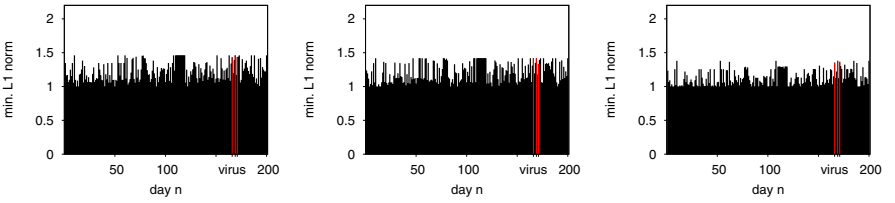
Table 2. Recall and Precision

Nbr. Dev.	Diffusion KL (R/P)	Mahalanobis (R/P)	L1 (R/P)
7 ptypes			
3	0.33 / 0.33	0.33 / 0.33	0 / 0
6	1.00 / 0.50	0.33 / 0.16	0 / 0
9	1.00 / 0.33	0.33 / 0.11	0 / 0
14 ptypes			
3	0.66 / 0.66	0.33 / 0.33	0 / 0
6	1.00 / 0.50	0.33 / 0.16	0.33 / 0.16
9	1.00 / 0.33	0.66 / 0.22	0.33 / 0.11
21 ptypes			
3	0.66 / 0.66	0.33 / 0.33	0 / 0
6	1.00 / 0.50	0.66 / 0.33	0.33 / 0.16
9	1.00 / 0.33	1.00 / 0.33	1.00 / 0.33

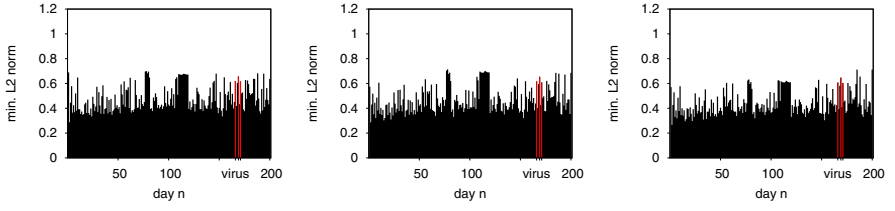
3.3 Discussion

The experiments in this section are conducted on two different data sets which reflect real world usage but differ from each other in terms of their characteristics. Both

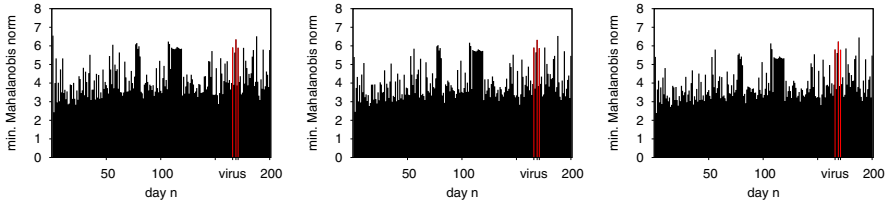
[3] Notice that precision and recall values can also be calculated by choosing thresholds. However, we do not want to limit our investigation to a specific threshold set or algorithm at this stage.



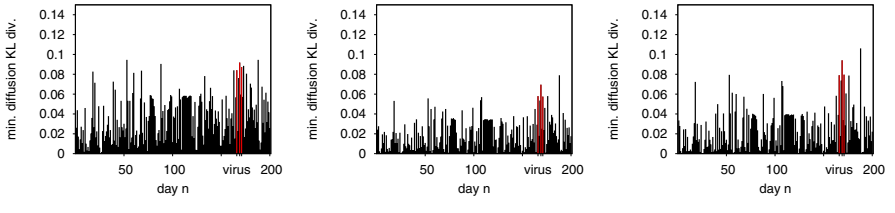
(a) min. l_1 distances between prototypes and test vectors on each day of the test period (left-to-right: 7, 14, and 21 prototypes)



(b) min. l_2 distances between prototypes and test vectors on each day of the test period (left-to-right: 7, 14, and 21 prototypes)



(c) min. Mahalanobis distances between prototypes and test vectors on each day of the test period (left-to-right: 7, 14, and 21 prototypes)



(d) min. Kullback-Leibler divergences between diffused prototypes and test vectors resulting from the algorithm in Fig. 2 on each day of the test period (left-to-right: 7, 14, and 21 prototypes)

Fig. 5. Malware detection results obtained from various methods. For all methods in (a), (b), (c), and (d), the number of prototypes representing normal behavior are 7, 14, and 21. The days with malware activity, 166, 169, and 171, are colored red.

data sets result from daily usage by ordinary volunteers over extended periods of time. Therefore, however limited, they can be seen as representative. Into both data sets we insert only one Trojan that exhibits itself only a few times within the test data. This is done on purpose in order to increase realism and simulate the situation of "searching a needle in a haystack". Similarly, we limit the available training data as much as possible for both cases to create realistic scenarios. In practice, the usage patterns are hardly stationary, i.e. even if usage data of a smartphone user from months ago was available, it would not be of much use due to changing behavioral patterns.

Each one of the experiments conducted and the respective data sets represent different approaches. Monitoring the system parameters of a smartphone continuously and using them for malware detection has advantages as it gives a broader picture of the situation by observing the complete profile of the device. However, resource usage –especially battery, CPU, and network access– are the clear disadvantages of this approach. Malware detection based on application (communication) logs, on the other hand, brings less overhead to the system. While the amount of information obtained on the system is much more limited, the focus is on behavioral aspects of the problem. This application-layer approach has its unique strengths such as generation of an individual profile of the user independent of the specific device brand/type and opening doors to the direction of semantic interpretations. As we have discussed, the second approach can also be less intrusive in terms of privacy after preprocessing depending on the type of features selected.

By outperforming alternative methods on both data sets in various realistic scenarios, the presented probabilistic diffusion algorithm establishes its wide-range applicability and robustness with respect to training data availability. Furthermore, its lightweight nature allows for personalization at the user level, yet it can also be applied in a privacy-preserving manner as demonstrated. Both of the malware detection schemes based on the proposed diffusion algorithm can be deployed in conjunction with more traditional methods such as signature-based detection or as a last line of defense in addition to various other security measures.

4 Conclusion

This paper presents a behavior-based approach to anomaly-based malware detection for application in mobile security systems. Despite extensive research, present day malware detectors still suffer from high false alarm rates. Towards the problem of distinguishing between anomalies (malware) and normal usage, we propose a probabilistic diffusion scheme that is based on a bipartite graph model that models probabilistic dependencies between a set of prototypes and a set of features that characterize these prototypes. Our model allows for mapping distributions of features onto distributions of prototypes and vice versa. In this manner, we can define Markov processes over the set of prototypes whose stationary distributions constitute similarity rankings of prototypes. This resembles the page rank procedure used by Google. However, in contrast to the page rank algorithm and other recent contributions to ranking on graphs and manifolds, the approach presented in this paper does not require the definition or computation of adjacency relations among prototypes. Rather, due to our dual treatment of features and

prototypes the diffusion kernel that allows for ranking with respect to the manifold of prototypes results from Bayesian reasoning.

Given the stochastic diffusion scheme, malicious activities are detected by first mapping feature vectors to stationary distributions over a set of prototypes of normal behavior and then computing their Kullback-Leibler divergence w.r.t. the stationary rankings of the prototypes themselves. Experiments with multiple real-world data resulting from monitoring usage of different mobile phones demonstrate the wide-range applicability of the approach. Compared to several baseline algorithms that are frequently applied in present day malware detection systems, our method outperforms them consistently and in a robust manner. Furthermore, it has favorable characteristics such as privacy preservation of individual users when applied to communication logs instead of system-level data. On the other hand, statistical methods, no matter how advanced, can not fully distinguish between rare user behavior and symptoms of malware. This is due to the fact that they lack semantic capabilities and context-awareness. Therefore, our approach should not be seen as an ultimate solution but as a step towards application of novel and advanced machine learning schemes in the security domain.

Future work will further exploit the lightweight nature of the algorithm. For training and application alike, it only requires multiplications of small to medium sized matrices. It is therefore suitable for application on modern mobile devices. Moreover, since the detector is based on data matrices only, update schemes seem worthwhile to explore. Currently, we are investigating mechanisms of dynamic feedback control for equation (3) as well ways to update the data matrix in order to adjust to evolving usage patterns. In another direction, combining statistical detection methods and semantic descriptions, i.e. closing the "semantic gap" remains as an open research challenge.

References

1. Agarwal, S.: Ranking on Graph Data. In: Proc. Int. Conf. on Machine Learning, ICML, pp. 25–32 (2006)
2. Alpcan, T., Başar, T.: A game theoretic analysis of intrusion detection in access control systems. In: Proc. IEEE Conf. Decision and Control, pp. 1568–1573 (2004)
3. Axelsson, S.: The base-rate fallacy and its implications for the difficulty of intrusion detection. In: Proc. ACM Conf. on Computer and Communications Security, pp. 1–7 (1999)
4. Bloem, M., Alpcan, T., Başar, T.: An optimal control approach to malware filtering. In: Proc. 46th IEEE Conference on Decision and Control, New Orleans, LA (December 2007)
5. Bloem, M., Alpcan, T., Schmidt, S., Başar, T.: Malware filtering for network security using weighted optimality measures. In: IEEE Conference on Control Applicatons (CCA), Singapore (October 2007)
6. Bye, R., Luther, K., Camtepe, S.A., Alpcan, T., Albayrak, S., Yener, B.: Decentralized detector generation in cooperative intrusion detection systems. In: Masuzawa, T., Tixeuil, S. (eds.) SSS 2007. LNCS, vol. 4838, pp. 37–51. Springer, Heidelberg (2007)
7. Cheng, J., Wong, S.H.Y., Yang, H., Lu, S.: Smartsiren: virus detection and alert for smartphones. In: Proc. of Int. Conf. on Mobile Systems, Applications, and Services (Mobisys 2007), pp. 258–271 (2007)
8. Christodorescu, M., Jha, S., Seshia, S., Song, D., Bryant, R.: Semantics-Aware Malware Detection. In: Proc. IEEE Symp. on Security and Privacy, pp. 32–46 (2005)
9. Coursen, S.: The future of mobile malware. Network Security (8), 7–11 (August 2007)

10. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: Proc. CVPR, vol. 2, pp. 886–893 (2005)
11. Eagle, N., Pentland, A.S.: Reality mining: sensing complex social systems. Personal Ubiquitous Computing 10(4), 255–268 (2006)
12. Fouss, F., Pirotte, A., Renders, J.M., Saerens, M.: Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation. IEEE Trans. on Knowledge and Data Engineering 19(3), 355–369 (2007)
13. Gao, B., Liu, T.Y., Ma, W.Y.: Star-Structured Higher-Order Heterogeneous Data Co-Clustering Based on Consistent Information Theory. In: Proc. ICDM, pp. 880–884 (2006)
14. Idika, N., Mathur, A.: A Survey of Malware Detection Techniques. Tech. Rep. SERC-TR-286, Software Engineering Research Center (March 2007)
15. Kashima, H., Tsuda, K., Inokuchi, A.: Kernels for graphs. In: Schölkopf, B., Tsuda, K., Vert, J.P. (eds.) Kernel Methods in Computational Biology, pp. 155–170. MIT Press, Cambridge (2004)
16. Kondor, R., Lafferty, J.: Diffusion Kernels on Graphs and Other Discrete Input Spaces. In: Proc. ICML, pp. 315–322 (2002)
17. Langville, A., Meyer, C.: A Survey of Eigenvector Methods for Web Information Retrieval. SIAM Review 47(1), 135–161 (2005)
18. Luther, K., Bye, R., Alpcan, T., Muller, A., Albayrak, S.: A cooperative AIS framework for intrusion detection. In: Proc. of the IEEE Conference on Communication (ICC), Glasgow, Scotland, June 2007, pp. 1409–1416 (2007)
19. Maloof, M. (ed.): Machine Learning and Data Mining for Computer Security. Springer, Heidelberg (2006)
20. Messmer, E.: New approaches to malware detection coming into view. Network World (April 2007),
`http://www.networkworld.com/news/2007/042507-malware-detection.html`
21. Miettinen, M., Halonen, P.: Host-based intrusion detection for advanced mobile devices. In: Proc. of 20th Intl. Conf. on Advanced Information Networking and Applications (AINA'06), vol. 2, pp. 72–76. IEEE Computer Society, Washington (2006)
22. Salton, G.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
23. Schmidt, A.D., Peters, F., Lamour, F., Albayrak, Ş.: Monitoring smartphones for anomaly detection. In: Proc. of First Int. Conf. on Mobile Wireless Middleware, Operating Systems, and Applications (MOBILWARE 2008) (February 2008)
24. Schmidt, S., Alpcan, T., Albayrak, S., Başar, T., Muller, A.: A malware detector placement game for intrusion detection. In: Lopez, J., Hämmerli, B.M. (eds.) CRITIS 2007. LNCS, vol. 5141. Springer, Heidelberg (2008)
25. Wang, K., Stolfo, S.: Anomalous Payload-Based Network Intrusion Detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 203–222. Springer, Heidelberg (2004)
26. Wu, N., Zhang, J.: Factor-analysis Based Anomaly Detection and Clustering. Decision Support Systems 42(1), 375–389 (2006)
27. Yap, T.S., Ewe, H.T.: A mobile phone malicious software detection model with behavior checker. In: Shimojo, S., Ichii, S., Ling, T.-W., Song, K.-H. (eds.) HSI 2005. LNCS, vol. 3597, pp. 57–65. Springer, Heidelberg (2005)
28. Zhou, D., Weston, J., Gretton, A., Bousquet, O., Schölkopf, B.: Ranking on Data Manifolds. In: Proc. NIPS, pp. 169–176 (2004)