Malware Detection on Mobile Devices using Distributed Machine Learning

Ashkan Sharifi Shamili, Christian Bauckhage
Bonn-Aachen International Center
for Information Technology

Tansu Alpcan
Deutsche Telekom Laboratories
Technical University of Berlin
alpcan@sec.t-labs.tu-berlin.de

Abstract

This paper presents a distributed Support Vector Machine (SVM) algorithm in order to detect malicious software (malware) on a network of mobile devices. The light-weight system monitors mobile user activity in a distributed and privacy-preserving way using a statistical classification model which is evolved by training with examples of both normal usage patterns and unusual behavior. The system is evaluated using the MIT reality mining data set. The results indicate that the distributed learning system trains quickly and performs reliably. Moreover, it is robust against failures of individual components.

1. Introduction

The widespread use and general purpose computing capabilities of smartphones make them the next big targets of malicious software (malware) and security attacks. Given the battery, computing power, and bandwidth limitations inherent to mobile devices, embedded detection of malware is a nontrivial research challenge that requires a different approach than the ones used for desktop or laptop computing.

Mobile device usage patterns such as the number of text messages sent or the duration of calls can serve collaboratively to derive flexible, personalized, and behavioral signatures of malware. For example, a security laboratory can provide the malware behavior data while the participating users provide the system with their normal usage data. Once a classifier has been collectively trained, it is used to detect malware and other attacks.

The problem of malware detection has recently gained interest in the field of pattern recognition [9, 3]. Anomaly detection can be considered as a binary classification problem which is solvable using statistical learning [7, 13]. Unsurprisingly, most of previously proposed schemes in anomaly detection are based on machine learning and data mining techniques [8, 10, 11, 12]. Decentralized malware detection approaches have been investigated in [5]. Support Vector Machines (SVMs) have been applied to malware detection on mobile phones in [4]. Given the limitations of mobile computing platforms, how to decrease the overhead of a detection algorithm by distributing the computation to multiple units and exchanging only a minimum number of support vectors has been discussed in [1].

Distinguishing between actual malware and mere unusual behavior is a challenge and heed has to be paid to avoid high false alarm rates. The inherent non-convex nature of malware detection complicates the problem, as does the limited availability of attack information data. Real time monitoring is another issue that has to be considered and unnecessary overhead must be avoided. These limitations and constraints motivate further research in the area of mobile malware detection.

This paper studies the distributed support vector machine scheme [1, 2] by experimenting on the well known data set of the MIT reality mining project [6] with different settings. The distributed learning approach adopted provides multiple advantages:

- It is a lightweight scheme in terms of bandwidth usage, since it does not require the mobiles to send all of their behavior data to a security center.
- It preserves the privacy of the participating users, since the communicated data are highly abstract and since a central repository for all of user data is not required.
- It takes into account usage patterns of ordinary users in order to automatically generate a general behavioral signature of malware.

Given its favorable properties, this scheme provides a promising and low-overhead defensive layer for mobile devices, possibly alongside with existing approaches.

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.1057

2. Background

The key idea of our distributed machine learning framework, is to divide the quadratic SVM binary classification problem into multiple separate sub-problems by relaxing it using a penalty function. Then, distributed continuous- and discrete-time gradient algorithms are applied to solve the relaxed problem iteratively. It can be shown that the synchronous parallel update scheme converges to the approximate solution geometrically. Furthermore, an asynchronous algorithm, where only a random subset of processing units are updated in each round, also converges geometrically which increases practical applicability of the scheme. The optimal margin nonlinear binary SVM classification problem is formalized in the quadratic problem

$$\begin{aligned} \max_{\alpha_d} \quad & \sum_{d=1}^{N} \alpha_d - \frac{1}{2} \sum_{d=1}^{N} \sum_{e=1}^{N} \alpha_d \alpha_e q_{de} \\ \text{such that} \quad & \alpha_d \geq 0, \quad d = 1, \ldots, N \\ \text{and} \quad & \sum_{d=1}^{N} \alpha_d y_d = 0, \end{aligned}$$

where the α_d are the Lagrange multipliers of the corresponding *support vectors* (SVs) and

$$q_{de} = y_d y_e \, k(x_d, x_e).$$

Here, k denotes the positive definite kernel function, x the data vectors, and y the positive and negative labels.

After relaxation and Lagrangian decomposition, each of the mobile devices solves the following unit problem on a subset S_i of the available usage data

$$\begin{aligned} \max_{\alpha_d^{(i)} \in [0, \alpha_{max}]} \quad & F_i(\alpha) = \sum_{d \in S_i} \alpha_d - \frac{1}{2} \sum_{d \in S_i} \sum_{e=1}^{N} \alpha_d \alpha_e q_{de} \\ & - \frac{\beta}{2} \Big(\sum_{l=1}^{N} \alpha_l y_l \Big)^2 \\ \text{such that} \quad & \alpha_d \geq 0, \quad d = 1, \ldots, N \end{aligned}$$

using the following discrete-time gradient algorithm for each of their training samples

$$\alpha_d(n+1) = \alpha_d(n) + \kappa_d G_d\big(\alpha(n)\big) \; \forall d$$

where

$$\begin{aligned} G_d\big(\alpha(n)\big) = 1 - \frac{1}{2} \Big(& \alpha_d(n) q_{dd} - \sum_{e=1}^{N} \alpha_e(n) q_{de} \Big) \\ & - \beta y_d \sum_{l=1}^{N} \alpha_l(n) y_l \, . \end{aligned}$$

We refer to [1] for the details of the algorithm.

Table 1. Histogram Features.

Feature (in numbers per 6 hour intervals)
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

3. Data Set and Analysis

In our quantitative experiments with the proposed scheme we consider the multi-user data set of the MIT Reality Mining project [6]. It consists of data of phone calls, short messages (SMSs), and data communication logs collected via a special application during normal daily usage of volunteers. In total, the Reality Mining data consists of 897922 communication logs collected from 97 users. We experiment with the data of 75 users whose recordings exceed 25 days of activity.

This usage data is pre-processed to generate histograms over a set of 20 features (see Table 1) that cover time intervals of 6 hours. Here, short periods refer to less than 1 hour, medium ones to between 1 and 3 hours, and long ones to more than 3 hours, respectively. A short call duration is considered to be less than 2 minutes, a medium one to be between 2 and 6 minutes, and a long one to be more than 6 minutes. Due to the statistical nature of the histogram features, the privacy of users who participate in distributed training is preserved.

We experiment with malware that behaves similar to well-known Viver 1 or Beselo 2 Trojans. It sends out an SMS every other minute, up to 20 in less than an hour, but at least once per day. In each experiment, we infect

half of data set with malware symptoms. For training and test data, a random number $R < 20$ is added to the counts of *Outgoing SMSs* and the count of *Short periods between outgoing SMSs* are varied by $R - 1$.

4. Experiments

We implement the presented framework on smart phones with *Symbian S60* and emulate a network of such mobile devices. A light weight Python (PyS60) runtime environment allows implementation client code on these phones. The clients can, for example, communicate via a router and a PC which acts as the server. In this case, the clients are emulated on the PC. Each client processes the data of a single user of the Reality Mining data set where the data of two users are contaminated with the malware signature.

Each mobile sends its own set of SVs to the server and the server replies by sending back the aggregated set of SVs of all clients to each participating client. After it has received the set of all the SVs that characterize the current learning progress of the whole the distributed system, each unit updates its own SVs and once again sends them to the server.

We initially invoke 25 clients to collaboratively derive the maleware detection model. After the training phase, when the model has been learned, testing is done with the data of the other 50 users in our data set. Since every client has access to the aggregated SVs, the testing can be done in either single unit. Since the number of SMSs sent by the maleware influences the detection rates (true- and false positives alike), we evaluate the effect of different levels of activity, i.e. the effect of various amounts of SMSs sent by the malware.

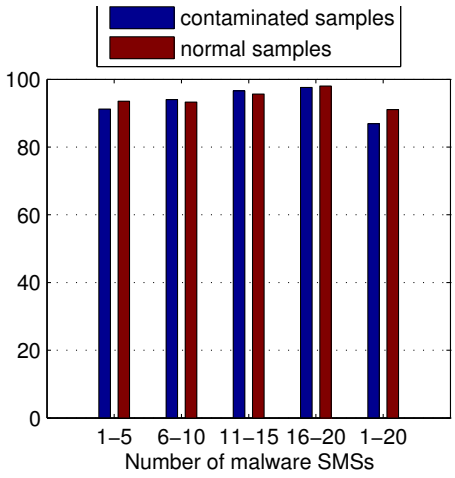

Figure 1. Recognition rates for different levels of maleware activity.

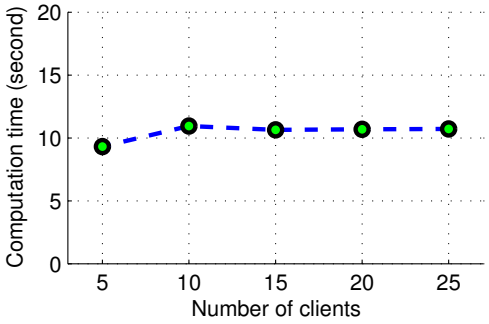

Figure 2. Average computation time per client during the training phase of the distributed system plotted w.r.t. different numbers of participating clients.

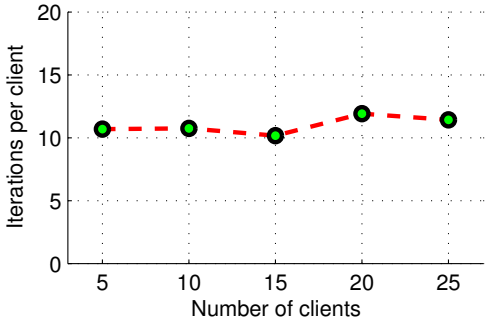

Figure 3. Average number of updates per client during the training phase of the distributed system plotted w.r.t. different numbers of participating clients.

Figure 1 illustrates the average recognition rates obtained from experimenting with different amounts of SMSs sent by malware. For a higher activity on the part of the malware, we achieved better accuracy rates. Highly invariant behavior (i.e. randomly sending between 1 and 20 SMSs in an hour) proved harder to be accurately detectable.

Requirements with respect to computational efforts are an important issue in real time systems. Figure 2 illustrates the average computation time per client that is measurable when different numbers of clients are learning the model in the training phase. This plot underlines another advantage of the distributed scheme: while the number of clients, and correspondingly the amount of data available to the system, are increasing the time required for training the system does not significantly increase, since, the participating clients perform their computation in parallel and update their SVs simultaneously. Therefore, the average number of updates per

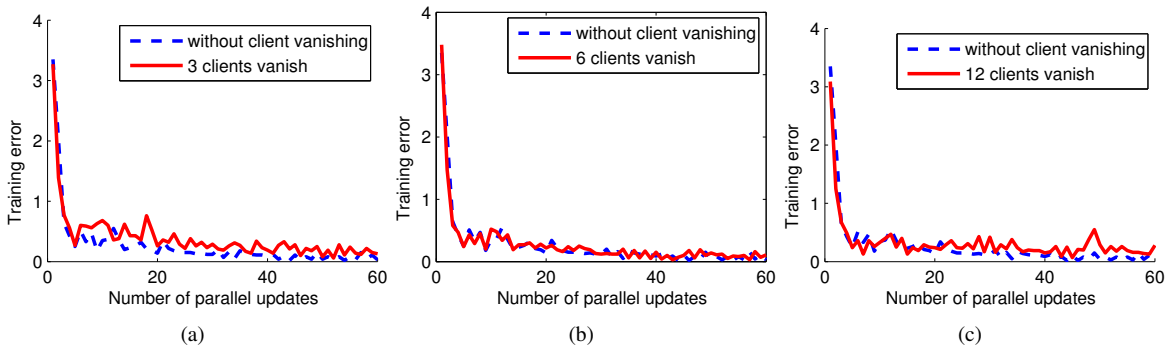

Figure 4. Effect on the convergence rate of the training process when different numbers of clients vanish from the system during the training phase.

client is a good indicator for the required computation time and should behave similarly. Figure 3 indicates that this is indeed the case,

Finally, we evaluate the robustness of the proposed system and examine its behavior when a client drops out of the system (say, because a user switches off the phone). Figure 4 shows the effect on the convergence behavior of the training process. We tested with up to half of the clients dropping from the system and found it to converge to useful solutions in all cases. The number of iterations to reach a solution as good as that of an undisturbed system did increase but never significantly.

5. Conclusion

This paper has investigated the practical behavior of a novel, distributed system for anomaly detection in mobile networks. Examples of normal and abnormal usage patterns are used to train the statistical classifier. Experiments were conducted with a network of cell phones processing data from the MIT reality mining project. Our results underline that the system performs reliably, trains quickly, and is robust against component failures.

Acknowledgements

This work has been supported in part by Deutsche Telekom Laboratories.

References

[1] T. Alpcan and C. Bauckhage. A discrete-time parallel update algorithm for distributed learning. In *Proc. Int. Conf. on Pattern Recognition*, 2008.

[2] T. Alpcan and C. Bauckhage. A distributed machine learning framework. In *Proc. IEEE Conf. on Decision and Control*, 2009.

[3] T. Alpcan, C. Bauckhage, and A.-D. Schmidt. A probabilistic diffusion scheme for anomaly detection on smartphones. In *Proc. Workshop on Information Security Theory and Practices*, 2010.

[4] A. Bose, X. Hu, K. G. Shin, and T. Park. Behavioral detection of malware on mobile handsets. In *Proc. Int. Conf. on Mobile Systems, Applications And Services*, 2008.

[5] R. Bye, K. Luther, S. A. Camtepe, T. Alpcan, S. Albayrak, and B. Yener. Decentralized detector generation in cooperative intrusion detection systems. In *Proc. Int. Symp. on Stabilization, Safety, and Security of Distributed Systems*, 2007.

[6] N. Eagle and A. S. Pentland. Reality mining: Sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[7] I. D. Gesu, G. L. Bosed, and J. H. Friedman. Intruders pattern identification. In *Proc. Int. Conf. on Pattern Recognition*, 2008.

[8] M. A. E. Maloof. *Machine Learning and Data Mining for Computer Security*. Springer, 2006.

[9] M. Reif, M. Goldstein, A. Stahl, and T. M. Breuel. Anomaly detection by combining decision trees and parametric densities. In *Proc. Int. Conf. on Pattern Recognition*, 2008.

[10] K. Rieck, T. Holz, C. Willems, P. Duessel, and P. Laskov. Learning and classification of malware behavior. In *Proc. Int. Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2008.

[11] K. Wang and S. J. Stolfo. Anomalous payload-based network intrusion detection. In *Proc. Int. Symp. on Recent Advances in Intrusion Detection*, 2004.

[12] N. Wu and J. Zhang. Factor-analysis based anomaly detection and clustering. *Decision Support Systems*, 42:375–389, 2006.

[13] D. Yang and H. Qi. A network intrusion detection method using independent component analysis. In *Proc. Int. Conf. on Pattern Recognition*, 2008.