Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2012, Article ID 262193, 12 pages
doi:10.1155/2012/262193

Research Article

A Novel Behavior-Based Virus Detection Method for Smart Mobile Terminals

Yanbing Liu,[1] Shousheng Jia,[1] and Congcong Xing[2]

[1] *School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China*
[2] *Department of Mathematics and Computer Science, Nicholls State University, Thibodaux, LA 70310, USA*

Correspondence should be addressed to Yanbing Liu, liuyb@cqupt.edu.cn

Received 1 July 2012; Revised 2 August 2012; Accepted 4 August 2012

Academic Editor: Xiaofan Yang

The security of smart mobile terminals has been an increasingly important issue in recent years. While there are extensive researches on virus detections for smart mobile terminals, most of them share the same framework of virus detection as that for personal computers, and few of them tackle the problem from the standpoint of detection methodology. In this paper, we propose a behavior-based virus detection method for smart mobile terminals which signals the existence of malicious code through identifying the anomaly of user behaviors. We first propose a model to collect and analyze user behaviors and then present a polynomial time algorithm for the virus detection. Next, we evaluate this algorithm by testing it with two commercial malwares and one malware written by ourselves and show that our algorithm enjoys a high virus detection rate. Finally, we notice that the rate of change of the virus detection rate of the algorithm with respect to thresholds matches the real-world situation of user behaviors, which indicates that the proposed algorithm is feasible.

1. Introduction

Smart mobile terminals are a new type of communication devices that combine the functionality of a traditional mobile terminal (i.e., voice-mail and text-messaging) with that of a handheld computing device such as PDA. Unlike traditional mobile terminals, smart mobile terminals can run third-party software applications and are data-centric [1]. These advantages make the smart mobile terminal very attractive to a large number of users. According to a study by the International Data Corporation, the number of smart mobile terminals worldwide was 303.4 million in 2010 and more than 450 million in 2011 [2].

Unfortunately, the appealing features of smart mobile terminals with respect to their data transmission ability, their growing popularity, and their capability of running

third-party software applications have been taken advantage of by many virus writers. This is partially due to the fact that there is a lack of regulations and standards on the third-party application software developments and usages. As a result, those application programs are typically "certified" by the developers themselves and cannot be fully trusted. A paradigmatic example is the mobile phone worm Cabir, which was developed by the virus writing group 29A in 2004 [3], and spreads across wireless networks through Bluetooth. In a study conducted by some German researchers [4], smart mobile terminals are identified to be extremely vulnerable allowing easy intrusions from a malicious third party. In particular, they found that usernames and passwords are at a high risk when they are transferred via smart mobile terminals. Evidently, virus detection for smart mobile terminals is a vitally important and pressing issue.

The rest of this paper is organized as follows. Section 2 overviews the existing major techniques for virus detections on smart mobile terminals. Section 3 proposes our behavior-based model for virus detection. Section 4 presents and analyzes the algorithm for the model. Section 5 evaluates the algorithm by testing it with three malwares and Section 6 concludes the paper.

2. Related Work

Virus detections mainly depend on the antivirus softwares tools. The detection technique used by most of the antivirus softwares detects viruses by scanning files against a (known) virus signature database. While this detection technique is instantaneous, accurate, and mature, it unfortunately cannot work out well in the context of smart mobile terminals for the following reasons. (1) The variety of operating systems and hardware implementations of smart mobile terminals, as well as their relatively limited openness to the public, make it rather difficult to collect the type of smart mobile terminal viruses [5]. (2) The mobile and compact natures of any smart mobile terminal, such as its memory size, processing power, and battery life, affect the execution of an antivirus program installed on it [6]. (3) The situation that an antivirus program installed on a smart mobile terminal needs to be updated frequently causes a dilemma: if the user does not stay online long enough, then the antivirus program will not have a chance to be updated properly which will directly affect the virus detection on the terminal; if the user does stay connected long enough, however, then that means he has to pay extra fees which could have been saved otherwise [1]. In [7], four currently available antivirus solutions for handheld devices are evaluated to pinpoint the weaknesses in the present antivirus software tools.

In addition to detecting viruses by scanning files, there is another virus detection technique which determines whether a terminal is infected with viruses by comparing the real-time code or terminal behaviors with the user normal behavior profile formed before to see if there are any behavior anomalies. Two types of virus detection technique in this line exist: static analysis and dynamic analysis.

Among the researches of static analysis, Schmidt et al. [8, 9] proposed two static analysis models that can be used in the Android system. Schmidt et al. [10] also used a classification algorithm to monitor the Symbian system function calls in programs in order to determine if some behavior anomaly has occurred.

In the study of dynamic analysis, Egele et al. [11] provided a complete survey on malicious code detections and program behavior anomaly discoveries and commented on the development, advantages, and disadvantages of various dynamic virus detection techniques.

Cheng et al. [1] proposed the first Windows Mobile dynamic unknown-virus detection model. Schmidt et al. [12] presented a terminal virus detection model which works by checking Linux kernel calls from application programs. In [13], Enck et al. demonstrated an Android-based behavior anomaly detection system which basically keeps track of an application program's access to sensitive system information and thereby determines whether or not this program's behavior is normal with the invocation of a relevant algorithm.

It seems that most of the existing researches on mobile terminal malicious code detections focus on selecting different attributes of the system to monitor or to study and rarely seek to find a new approach in terms of methodology. Note that mobile terminals are a part of people's daily life; therefore, the observed behaviors of mobile terminals naturally reflect some aspects of human behaviors. Note also that human behaviors are multidimensional and have cycles, which should be considered when we deal with the virus detections on smart mobile terminals. As such, we, in this paper, propose a novel user-behavior-based virus detection scheme for smart mobile terminals which takes the factor of human behaviors into account and detects virus by identifying anomalies in users' behaviors.

3. Behavior-Based Model for Virus Detection

We now describe the user behavior-based model for virus detections. The model creates a user behavior profile through collecting the user's regular behaviors first and then compares the user's real-time behavior with the established behavior profile through an invocation of the corresponding algorithm to determine if the terminal is infected with malicious code.

3.1. Selection of the User Behavior Indicator

The current researches for virus detections mostly concentrate on some properties of the application programs installed on mobile terminals, rather than on the properties of smart mobile terminals themselves. Although a user might make frequent changes in terms of using a specific application program, the usage of the terminal as a whole tends to be invariant which can be seen and measured by such terminal properties as power consumptions and traffic volumes. For example, if a user experiences a high power consumption (on using some programs), then he would try to keep the total power consumption balanced by reducing the uses of other programs. Thus, determining whether a terminal is infected with virus by inspecting a certain property of the terminal itself seems to be a more reasonable approach. Actually, Buennemeyer et al. [14] proposed a remarkable virus detection model which determines user behavior anomaly in real time by inspecting the power consumptions of terminals; Shabtai et al. [15] demonstrated a host-based behavior anomaly detection system which, via the technique of machine learning, also works by inspecting the usage situation associated with some terminal properties.

With the rapid development of the mobile Internet, we have been experiencing an increasing number of application programs that depend on the access to the mobile Internet, for instance, the weather forecasting software and the Instant Messenger software. Also, some application program developers embed certain adware (e.g., AdMob) into their products (especially free products) which needs a frequent access to the mobile Internet to update itself [16]. All of these not only can result in a huge undesired increase of traffic on users' mobile terminals, but also may provide hackers an opportunity to invade users' privacy. Unfortunately, the traffic management component of most security software tools can only

keep track of the traffic volume for each individual program installed on a terminal and is unable to determine whether the terminal experiences an abnormal behavior in a timely manner. As a result, it may just be simply too late when users themselves notice some unusual traffic situations on their terminals. Thus, it is significant to be able to detect users' traffic anomalies in real time. In this paper, we select the users' traffic volume as their behavior indicator to devise the virus detection technique.

3.2. Description of the Model

The model first collects a user's traffic data over a certain period of time and designates this set of data as the initial value regarding the user's behavior properties. This initial set of data is then classified into different classes through the clustering technique; thereby, a user's normal behavior profile is established. Subsequent traffic data collected from the user is then compared to the user's profile via a similarity-checking scheme to determine whether a user behavior anomaly has occurred. Accordingly, the model consists of the following modules: a user behavior collecting module, a user behavior profile creating module, a virus detection module, and a virus alerting and bookkeeping module.

It is worth noting that users' behaviors are heterogeneous. For example, a user may exhibit distinct (but normal) behaviors at different occasions (such as on vacation, at work, on business trips, etc.). As such, simply classifying a user's real time behaviors into one class may not be able to correctly determine if this behavior is normal or not, and in the worst case scenario, may treat a normal behavior as abnormal. We, thus, in addition to the clustering, formalize the notion of similarity between two instances of data and use the ensuing calculation of similarity to ensure the correctness of the data classification and virus detection. A user normal behavior profile is first created by the model using the initial data, and subsequent data instances collected on each day are checked against this profile by the corresponding virus detection algorithm. If the computing result shows no sign of virus, then the user profile will be updated by including the new instances of data into appropriate classes; otherwise, the user will be alerted that the mobile terminal is probably infected with some viruses. The flow chart of the model is depicted in Figure 1.

3.3. Module for Collecting User Behaviors

This module acquires the factual data regarding the user terminals' traffic, preprocesses this set of data for the purpose of preparation, and then passes it to the user behavior profile creating module for the establishment of user normal behavior profiles.

We divide a day (24 hours) evenly into p time intervals. For example, if $p = 4$, then the four time intervals could be (02–08, 08–14, 14–20, and 20–02). The traffic volume for a specific time interval t $(1 \leq t \leq p)$ can be obtained by subtracting the traffic volume at the beginning time of the interval t from that at the end time of the interval t. In the example above, if $t = 3$, then the beginning time and the end time for this internal would be 14 and 20, respectively. The aggregation of the traffic volume of each interval forms a piece of data for the user behavior collecting module. The format of data is defined as follows.

Definition 3.1. A data instance X is a tuple

$$(a_1, a_2, \ldots, a_p), \tag{3.1}$$

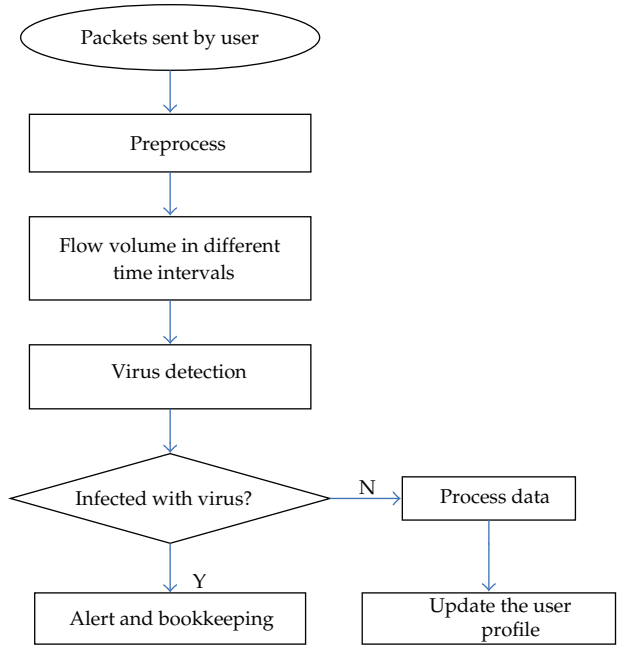

Figure 1: Model flow chart.

where p is the number of time intervals in a day, and a_i $(1 \leq i \leq p)$ is the traffic volume in the ith time interval.

Much of our experiences and heuristics indicate that human behaviors bear a certain kind of periodicity and regularity, which is further evidenced by [17] in which it is shown that the interevent time associated with human behaviors follows the fat-tailed distribution. In our work, we stipulate that the users' behavioral period for smart mobile terminals is one week; subsequently, the user behavior collecting module gathers the user traffic information for a week as the initial data for the model.

3.4. Module for Creating User Profiles

The task of this module is to establish user normal behavior profiles which give a defining description for the normal user behaviors. In other words, this module records the traffic flow volumes at different time intervals when the user is not under attack.

As mentioned before, human behaviors are heterogeneous and multifarious. Even within the same behavioral period, the behaviors exhibited by a person when he is at work, on vacation, or on business trips tend to be all different. As an indispensable part of people's life, mobile terminals should accordingly possess multifarious behaviors or performances. Clearly, it would be unreasonable to treat all user behaviors in one behavioral period as the same and classify them into only one group. We, instead, split the user behaviors in one behavioral period into several classes using the clustering technique proposed in [18] and thereby create multiple user behavior subprofiles (one subprofile for each class). By doing so, the heterogeneous and multifarious nature of user/terminal behaviors will be respected. The details of creating user behavior profiles will be discussed in Section 4.

3.5. Module for Virus Detections

This module consists primarily of a virus detection algorithm which takes the data collected from a user terminal in a day as input, performs the relevant computation, and produces an answer to the question: "Is this terminal infected with virus?" If the result is negative, then the user's normal behavior profile contents will be updated with the current data to make it more accurate; otherwise, the virus alerting module will be invoked with a warning message. Both the virus detection algorithm and the user behavior profile updating will be described in detail in Section 4.

3.6. Module for Virus Alerting and Bookkeeping

Once having received the warning message sent from the virus detection module, this module alerts the user that the terminal may have been infected with some viruses or worms and at the same time records such information as date, time, similarity degree calculation, traffic volume, and so forth into a log. Users can then likely locate the viruses by examining the recently installed programs on and/or downloaded files to the terminal. Moreover, users can also educate themselves about terminal behavior anomalies by studying the contents and properties shown in the log, which would be valuable in helping them to be more vigilant to terminal behavior anomalies in the future.

4. Algorithms Used in the Model

In this section, we present the algorithms that deal with the user behavior profile creation and updating and virus detections.

4.1. Creation and Updating of User Profiles

Our model uses the clustering technique to classify user behaviors into different classes. An entire user behavior profile consists of several classes (or subprofiles), whereas each class consists of several data instances which are collected by the user behavior collecting module. The clustering technique used in our work is the classic MacQueen's k-means algorithm [19] which is notably simple and fast at converging.

Let m be the number of classes resulted in from the k-means clustering algorithm, and $X_{\text{class}(1)}, X_{\text{class}(2)}, \ldots, X_{\text{class}(m)}$ be the initial centers of these classes. If a data instance X (i.e., traffic flow volumes collected on some day) can be classified into class k and does not reveal any abnormal behaviors, then class k needs to be updated. Assuming there are n data instances $X_1, X_2, \ldots, X_n$ with each $X_i = (a_{i1}, a_{i2}, \ldots, a_{ip})$ in class k, the updating of class k is a two-step process.

Step 1. Include X into class k as a new data instance, that is, let $X_{n+1} = X$.

Step 2. Recalculate the center of class k, $X_{\text{class}(k)} = (\sum_{i=1}^{n+1} a_{i1}/(n+1), \sum_{i=1}^{n+1} a_{i2}/(n+1), \ldots, \sum_{i=1}^{n+1} a_{ip}/(n+1))$.

4.2. Definition of the Similarity Degree

We now address the issues concerning the algorithm for virus detections. We introduce a new notion *similarity degree* which formalizes the situation of how close (or similar) two data objects can be. Given any two data objects, the basic idea is to use the reciprocal of their distance to capture the degree of their similarity, so that the smaller (or larger) the distance the greater (or littler) the similarity.

Note that there are mainly two different formulations regarding the measurement of distance: Euclidean distance [20] and Mahalanobis distance [21]. Euclidean distance is the one that is commonly used in calculating the distance between two points in an n-dimensional space. In our work, Euclidean distance is adopted in calculating similarity degrees between two data instances or between a data instance and a class.

Definition 4.1. If p is the number of time intervals in a day, and $X_i = (a_{i1}, a_{i2}, \ldots, a_{ip})$ and $X_j = (a_{j1}, a_{j2}, \ldots, a_{jp})$ are two data instances, then the similarity degree between X_i and X_j is defined as flows:

$$\mathrm{sim}(X_i, X_j) = \frac{1}{\sqrt{\sum_{k=1}^{p} (a_{ik} - a_{jk})^2}}. \tag{4.1}$$

Definition 4.2. Let $X = (a_1, a_2, \ldots, a_p)$, $\mathrm{class}(i) = (X_{i1}, X_{i2}, \ldots, X_{in})$, and $X_{\mathrm{class}(i)}$ be, respectively, a data instance, a class with n data instances, and the center of class i. The similarity degree $\mathrm{sim}(X, \mathrm{class}(i))$ between X and class i is defined to be the similarity degree between X and the center of class i, that is,

$$\mathrm{sim}(X, \mathrm{class}(i)) = \mathrm{sim}(X, X_{\mathrm{class}(i)}). \tag{4.2}$$

4.3. Virus Detection Algorithm

We are now ready to describe the virus detection algorithm which is shown in Algorithm 1.

This algorithm consists of four steps. The first step (lines 1-2) uses Definition 4.2 to calculate the similarity degree between X and each class $\mathrm{class}(i)$ of the entire user profile; $\mathrm{sim}(i) = \mathrm{sim}(X, \mathrm{class}(i))$ ($i = 1, 2, ..., m$, m is the number of user subprofiles). The second step (lines 3-4) finds the maximum similarity degree $MaxClassSim$ among all $\mathrm{sim}(i)$ and the class index x for which $\mathrm{sim}(x) = MaxClassSim$. The result of this step means that X belongs to the user subprofile (or class) x. The third step (lines 5–8) computes the similarity degrees between X and each data instance in class x and produces the minimum $MinSim$ and the maximum $MaxSim$ of all these similarity degrees. The last step (lines 9–14) determines if the terminal is infected with virus. Namely, if either $MinSim$ or $MaxSim$ is below the threshold value, then X exhibits an abnormal behavior and the terminal is infected with some virus; otherwise, X is a normal behavior and will be used to update the user subprofile x using the algorithm in Section 4.1.

4.4. Algorithm Time Complexity Analysis

We now show that the proposed algorithm has polynomial time complexity.

Input:
The traffic data instance X collected on one day (m is the number of user subprofiles; n is the number of instances in the user subprofile into which X can be classified.);
Output:
The answer to the question: "Is this terminal infected with virus?";
(3.1) **for** $i = 1$ to m **do**
(4.1) Calculate the Similarity degree sim(X, class(i)) between X and user subprofile i;
(4.2) **for** $i = 1$ to m **do**
(4) Find the maximum value *MaxClassSim* among all sim(X, class(i)) and the corresponding class class(x) that yields *MaxClassSim*;
(5) **for** $j = 1$ to n **do**
(6) Calculate the similarity degree sim(X,X_j) between X and the jth instance in class x;
(7) **for** $j = 1$ to n **do**
(8) Find the minimum value *MinSim* and the maximum value *MinSim* among all sim(X,X_j);
(9) **if** *MinSim* > a&&*MaxSim* > b **then** (a, b are predetermined thresholds.)
(10) Update subprofile x by including X into it;
(11) **return** there's no virus;
(12) **else**
(13) **return** the terminal is infected with virus;
(14) **end if**

Algorithm 1: Virus detection algorithm.

Proposition 4.3. *Algorithm 1 has time complexity* $O(m + n)$.

Proof. From Algorithm 1 and the explanation above, we can see that the first step takes $O(m)$ time to compute the similarity degrees between X and all user subprofiles; that the second step finds $MaxClassSim$ in $O(m - 1)$ time; that the third step takes $O(n + 2(n - 1))$ time in computing the similarity degrees between X and all instances of the user subprofile which it belongs to and finding their maximum and minimum values; that the last step of the algorithm is of $O(1)$ time. It follows immediately that the entire algorithm has time complexity $O(m + n)$. □

5. Experimental Results

In order to evaluate the virus detection algorithm presented in the previous section, we conducted some experiments by implementing the algorithm on the Android open platform. In the experiment, we split a day (24 hours) into four time intervals 02–08, 08–14, 14–20, and 20–02 and collected the regular traffic volume information of 10 volunteers on these intervals of a day, for a period of one week. This set of data was used to create the user normal behavior profiles by the relevant module described in Section 3.4. Each user profile was classified into 3 classes corresponding, respectively, to the different user behaviors exhibited when they are at work, on vacation, and on business trips, so that each user has 3 normal behavior subprofiles.

As an example, Figure 2 depicts the traffic volumes of a particular user on the four intervals of a day for consecutive 7 days (a week), and Figure 3 shows his total traffic volume of each day for that week. From these figures, it can be seen clearly that this user exhibits similar behaviors on Monday, Friday, Saturday, and Sunday and on Tuesday and Wednesday as well. His behavior on Thursday, however, is not close to that of any of the other days.

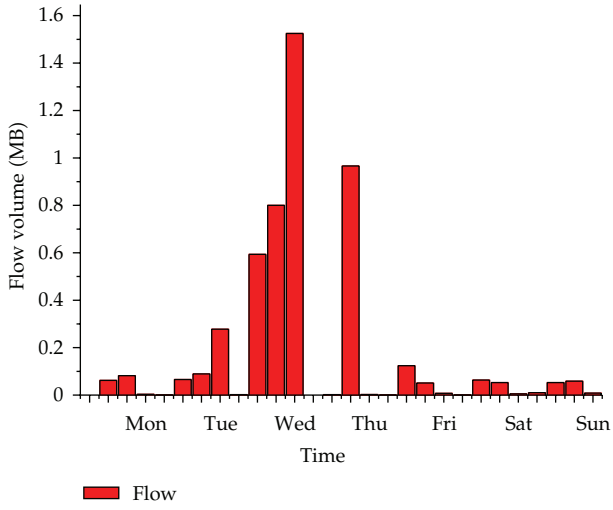

Figure 2: Traffic volumes indexed by time intervals.

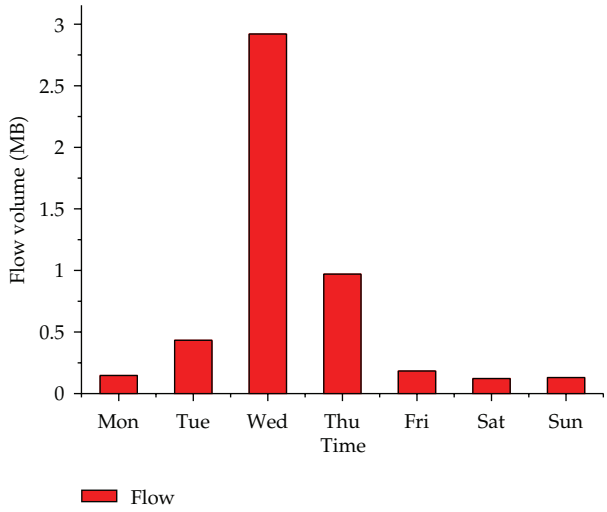

Figure 3: Traffic volumes indexed by a day.

We tested our algorithm by installing some malwares onto the volunteers' mobile terminals and running the algorithm to detect those malwares. The malwares used were the malicious version of Steamy Window and Monkey Jump 2, as well as a worm Furtively Load, which was written by ourselves. The detection rates are shown in Table 1 .

Evidently, Table 1 demonstrates that the algorithm has an excellent detection rate against the malware written by ourselves and a very good detection rate (above 85%) against the other two commercial malwares, which indicates that our algorithm is fully capable of detecting malicious code in an effective manner.

Figures 4 and 5 show the changes of the detection rate of the algorithm with respect to one of the two thresholds (a and b in Algorithm 1) when the other one is fixed. As we can see from these two figures, the detection rate increases with a decreasing rate of change as

Table 1: Comparison of virus detection rates.

Malware name	Number of total behavior anomaly	Number of successful detections	Detection rate
Steamy Window	100	91	91%
Monkey Jump 2	100	86	86%
Furtively Load	100	96	96%

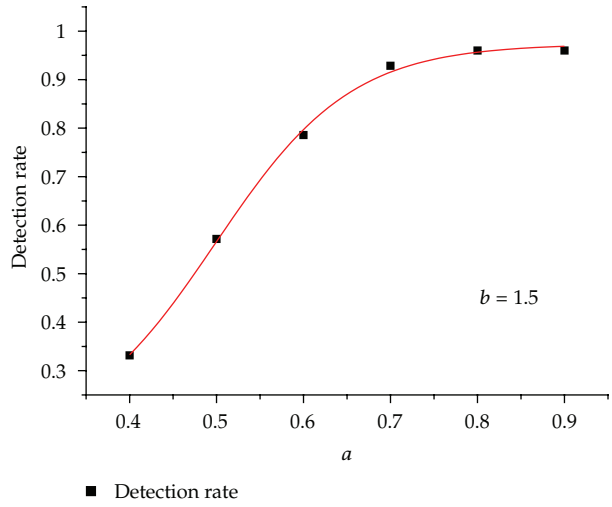

Figure 4: Detection rate with respect to a when $b = 1.5$.

the threshold increases, converging to a stable value when the threshold is beyond a certain point ($a = 0.8$ in Figure 4, $b = 1.5$ in Figure 5). This shows that the virus detection rate cannot be gained any more after the threshold reaches a certain value, which is consistent with the realistic situation that user behaviors are not identical in spite of being periodically similar since a higher threshold requires a closer matchup between user behaviors. Thus, we claim that our model faithfully reflects the human behaviors in the real world. Also, we found that when $a > 0.8$ with b being fixed, some normal user behaviors will be treated as abnormal creating a false alarm thereby. So a is set to be 0.8 in our algorithm, and b to 1.5 for the same reason.

6. Conclusion and Future Work

Human behaviors are heterogeneous, multifarious, and periodic; user behaviors on smart mobile terminals naturally resemble these characteristics of human behaviors. As such, these behavioral characteristics should be factored in when devising a behavior anomaly detection technique for smart mobile terminals, which is unfortunately not the case in the current virus detection research for mobile terminals.

In this paper, as an initial research effort to deal with this situation, we have proposed a novel virus detection strategy/algorithm for smart mobile terminals which takes into account the behavioral characteristics inherited by mobile terminals from humans. The analysis and implementation of the algorithm show that it is of polynomial time complexity, has a very high virus detection rate, and reflects the true user behaviors in the real world.

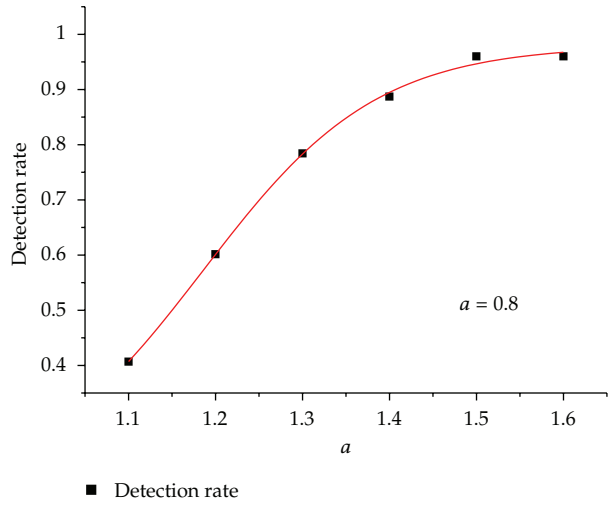

Figure 5: Detection rate with respect to b when $a = 0.8$.

Considering that the evaluation of a virus detection algorithm can be enhanced by checking how many normal user behaviors are caught as abnormal (in addition to the abnormal behavior detection rate), and that everyone lives in a society and interacts with a group of people, we plan, as the future work, to further the study of user behavior anomaly detections with a false alarm rate report and on the levels of individual, group, and the combination of both.

Acknowledgments

This work was partially supported by the program for NCET and the following grants: National Science Foundation of China (Grant no. 60973160), National 973 Foundation of China (no. 2010CB334710), R&D Foundation of Chongqing (Grant no. Kjzh10206), and Open Project of Key Lab of Information Network Security of Administration of Public Security (Grant no. C11609).

References

[1] J. Cheng, S. H. Y. Wong, H. Yang, and S. Lu, "SmartSiren: Virus detection and alert for smartphones," in *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys '07)*, pp. 258–271, ACM, June 2007.

[2] R. T. Llamas, W. Stofega, S. D. Drake, and S. K. Crook, "Worldwide smartphone 2011–2015 forecast and analysis," Tech. Rep., International Data Corporation, 2011.

[3] D. H. Shih, B. Lin, H. S. Chiang, and M. H. Shih, "Security aspects of mobile phone virus: a critical survey," *Industrial Management and Data Systems*, vol. 108, no. 4, pp. 478–494, 2008.

[4] B. Konings, J. Nickels, and F. Schaub, "Catching AuthTokens in the Wild The Security of Google's Client Login Protocol," http://www.uni-ulm.de/en/in/mi/staff/koenings/catching-authtokens.html.

[5] Z. Chang-Xing, "Prevention and analysis on development of mobile phone virus," *Information Tecnology*, vol. 2, pp. 123–124, 2010.

[6] "Celeste Biever, Boston, Phone viruses: how bad is it?" http://www.newscientist.com/article.ns?id=dn7080.

[7] J. A. Morales, P. J. Clarke, Y. Deng, and B. M. Golam Kibria, "Testing and evaluating virus detectors for handheld devices," *Journal in Computer Virology*, vol. 2, no. 2, pp. 135–147, 2006.
[8] A. D. Schmidt, R. Bye, H. G. Schmidt et al., "Static analysis of executables for collaborative malware detection on android," in *Proceedings of the IEEE International Conference on Communications (ICC '09)*, pp. 1–5, June 2009.
[9] A.-D. Schmidt, A. Camtepe, and S. Albayrak, "Static smartphone malware detection," in *Proceedings of the 5th Security Research Conference*, p. 146, 2010.
[10] A. D. Schmidt, J. H. Clausen, A. Camtepe, and S. Albayrak, "Detecting symbian OS malware through static function call analysis," in *Proceedings of the 4th International Conference on Malicious and Unwanted Software (MALWARE '09)*, pp. 15–22, October 2009.
[11] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware analysis techniques and tools," *ACM Computing Surveys*, vol. 44, 2012.
[12] A.-D. Schmidt, H.-G. Schmidt, J. Clausen et al., "Enhancing security of linux-based android devices," in *Proceedings of the 15th International Linux Kongress*, October 2008.
[13] W. Enck, P. Gilbert, B.-G. Chun, and N. A. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," in *Proceedings of the 9th USENIX Cconference on Operating Systems Design and Implementation*, ACM, October 2010.
[14] T. K. Buennemeyer, T. M. Nelson, L. M. Clagett, J. P. Dunning, R. C. Marchany, and J. G. Tront, "Mobile device profiling and intrusion detection using smart batteries," in *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS '08)*, pp. 296–305, January 2008.
[15] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, ""Andromaly": a behavioral malware detection framework for android devices," *Journal of Intelligent Information Systems*, vol. 38, pp. 161–190, 2012.
[16] "PConline, The first quarter of 2011 global Android mobile phone safety reports," http://pcedu.pconline.com.cn/android/1104/2388612_2.html.
[17] C. A. Hidalgo, "Conditions for the emergence of scaling in the inter-event time of uncorrelated and seasonal systems," *Physica A*, vol. 369, no. 2, pp. 877–883, 2006.
[18] W. Li-Na, X. We, and L. Zhu, "Research and implementation of an anomaly intrusion detection system model based on similarity cluster analysis," *Minimicro Systems*, vol. 25, pp. 1333–1336, 2004.
[19] S. Hong-Guang and W. Yu-wei, "An improved k-means algorithm for document clustering," *Journal of Shandong University*, vol. 43, pp. 1–5, 2008.
[20] M. N. Do and M. Vetterli, "Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance," *IEEE Transactions on Image Processing*, vol. 11, no. 2, pp. 146–158, 2002.
[21] I. Lopez and N. Sarigul-Klijn, "Distance similarity matrix using ensemble of dimensional data reduction techniques: vibration and aerocoustic case studies," *Mechanical Systems and Signal Processing*, vol. 23, no. 7, pp. 2287–2300, 2009.