Implicit User Re-authentication for Mobile Devices*

Sausan Yazji[1], Xi Chen[2], Robert P. Dick[2], and Peter Scheuermann[1]

[1] EECS Dept., Northwestern University, Evanston, IL. 60208
[2] EECS Dept., University of Michigan, Ann Arbor, MI. 48109
s-yazji@northwestern.edu, peters@ece.northwestern.edu,
chexi@umich.edu, dickrp@eecs.umich.edu

Abstract. Portable computers are used to store and access sensitive information. They are frequently used in insecure locations with little or no physical protection, and are therefore susceptible to theft and unauthorized access. We propose an implicit user re-authentication system for portable computers that requires no application changes or hardware modifications. The proposed technique observes user-specific patterns in filesystem activity and network access to build models of normal behavior. These are used to distinguish between normal use and anomalous use. We describe these automated model generation and user detection techniques, and explain how to efficiently implement them in a wireless distributed system composed of servers and battery-powered portable devices. The proposed system is able to distinguish between normal use and attack with an accuracy of approximately 90% every 5 minutes and consumes less than 12% of a typical laptop battery in 24 hours.

1 Introduction

Portable computing devices such as personal digital assistants (PDAs), cellphones, and laptop computers are now used to store and access sensitive information, e.g., bank account, commercial transaction, and private emails. However, they are frequently used in insecure locations with little or no physical protection, and are therefore susceptible to theft and potential unauthorized access.

User authentication is an essential part of any security policy that grants permission to access a specified account. Currently, the most widely used authentication method is explicit authentication, i.e., authentication is performed once at start-up by asking for a password [1], a fingerprint [2], a face profile [3], or a combination. However, all these approaches are intermittent and therefore susceptible to attack, e.g., an unauthorized user can access a portable computer either by stealing a password or exploiting an open account of an authorized user.

Theft of confidential data by unauthorized users accounts for much of the financial losses due to computer crime [4]. Much of this loss could be prevented by requiring a frequent login, which would impose an unwarranted burden on

* This work was supported in part by the National Science Foundation under awards CNS-0347941, CNS-0720691, and CNS-0613967.

D. Zhang et al. (Eds.): UIC 2009, LNCS 5585, pp. 325–339, 2009.
© Springer-Verlag Berlin Heidelberg 2009

the user. We propose an implicit re-authentication technique, MIA, to compliment to explicit authentication. It provides on-line protection for private data without burdening the user. Our system uses file access patterns and network activities to detect anomalous behavior. In a behavioral authentication system, the authentication module samples behavioral data and compares them with user-specific models. The comparison results are then used to identify users. Researchers have proposed using keystrokes [5] and mouse movements [6] for user re-authentication. However, they have not designed or evaluated these techniques for portable battery-powered computers.

In this paper, we describe the first implicit re-authentication technique for battery-powered wireless portable systems. This is also the first implicit re-authentication technique for portable systems based on filesystem and network activity. We describe how the framework may accept data from different authentication mechanisms to improve security. Our evaluation indicates that the proposed technique has the potential to substantially increase security and/or decrease frequency of explicit user authentication compared to explicit authentication techniques, alone. The proposed system is able to distinguish between normal use and attack with an accuracy of approximately 90% with a latency of 5 minutes.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 gives an overview of the proposed system architecture. Section 4 describes each architectural component. Section 5 analyzes the energy consumption of MIA. Section 6 presents the experimental setup and examines the results. Section 7 concludes the paper. Section 8 describes future work.

2 Related Work

Numerous user authentication techniques exist for portable devices. While password based mechanisms are the most used [1], they are insufficient. Therefore some researchers have proposed other techniques that rely on hardware or software enhancements for efficient authentication based on fingerprint [2], face recognition [3], and iris data [8]. Moon et al. [7] implemented a Gaussian mixture model based speaker verification module on a mobile system. However, all these approaches are used for one-time explicit authentication. In addition, most of them require application changes and hardware support. In contrast, our technique requires no application changes, additional hardware, or explicit actions by the user.

A number of previous approaches re-authenticate users by monitoring their behavior without explicitly requiring special inputs for re-authentication. Denning and Neumann [14] were the first to introduce a behavioral re-authentication system. Denning [10] proposed using audit logs for anomaly detection. Monrose and Rubin [5] proposed using keystroke dynamics for re-authentication. Lane and Brodley [9] studied a system that models the normal behavior at the command line prompt. Pusara and Brodley [15] used mouse movements for external user re-authentication. Pusara [6] built a signature-based intrusion detection

system based on a combination of keystroke dynamics, mouse movements, and graphical user interface events, in which the attacker's behavior was known a priori. However, energy consumption and other metrics relevant to use in portable, battery-powered systems were not taken into account. In contrast, our technique requires no automated or manual attacker modeling: it detects attacks entirely based on models of normal user behavior. In addition, we evaluate the energy consumption of MIA when used in battery-powered mobile wireless systems.

Other techniques that use implicit re-authentication by monitoring system call traces and program execution traces have also been proposed [12,13,11].

3 Problem Definition and System Architecture

The proposed MIA architecture was motivated by the following observations: (1) different individuals have differing computer use patterns, e.g., filesystem and network access, which may be used for identification; (2) operating systems have access to a great deal of information, e.g., file access and network activities; and (3) there is a great imbalance in the cost of expending energy on portable battery-powered devices and stationary computers with easy access to energy.

It is our goal to detect whether a portable device is under attack, i.e., whether an unauthorized user is attempting to access data, or modifying the system to allow such data to be later gathered. We also have a number of other requirements: (1) to simplify deployment, application or hardware changes should not be required; (2) to avoid burdening the user, the system should not require explicit re-authentication input; (3) the latency of attack detection should be minimized; (4) detection accuracy should be maximized; (5) the power consumption and computational overhead of the technique must be low enough to permit operation on portable, battery-powered devices; and (6) the system should still be able to operate temporarily in the absence of wireless network access.

Architecture Overview. Figure 1 shows the system architecture of the proposed MIA system. The *information capturing system* consists of operating system modules and other analysis software installed on the portable battery-powered device to register user activity continuously, and to create new log file every T minutes [1]. It also consists of feature extraction and data compression processes to reduce the energy consumption of data transmission. The data captured are then sent to *information management system*, which resides on a computer with higher performance and much looser power consumption constraints than the portable system. It is responsible for periodically rebuilding models for filesystem and network access, performing anomaly detection, and transferring this model back to the portable computing system. Wireless communication between the portable device and the high-performance computer used for the information management system may be intermittent. In the absence of a wireless link, model evaluation will continue to occur on the portable computer system, allowing the detection of attack at some power consumption penalty.

[1] The detection latency T should be small enough to detect an attack before the attacker is capable of harming the system.

Algorithm 1. Information Management System

```
For each user U_n do the following:
while TRUE do
    Receive data from the information capturing system
    if NewUser = TRUE then
        if time < D then
            Add log data to the data source
        else
            Build the user's profile
            NewUser = FALSE
            Send user's profile to the client
        end if
    else
        Perform anomaly detection
        if Abnormal behavior then
            Take action, e.g., re-authenticate, lock client, or send warning
        else
            Continue
        end if
    end if
end while
```

Algorithm 1 denotes that the *information management system* consists of two major processes; the model building process, and the anomaly detection process. In order to build a behavioral model for authenticated user, data from this user needs to be captured for a training duration, D. To determine the appropriate value of D, we performed a small-scale, long-duration user study with the network-based technique and found that a D of two months results in a false rejection rate, FRR, of 2% which is the probability that the normal user will be mistakenly identified as an attacker. However, our primary full-scale multiple-metric user study is limited to two weeks to avoid undue burden on the volunteer participants, somewhat reducing the accuracy achieved in our full-scale study.

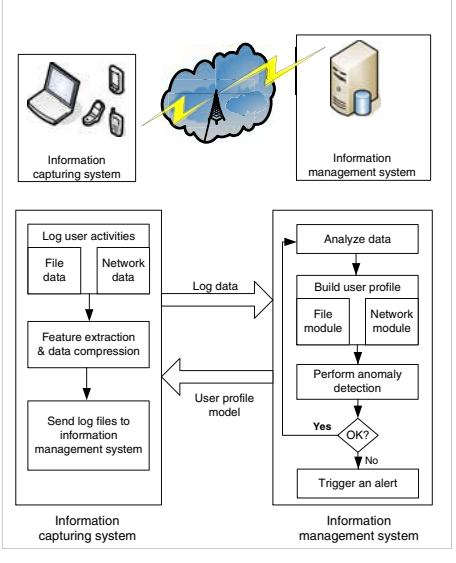

Fig. 1. MIA system architecture

Upon the completion of the data capturing phase, the "New User" flag is reset, and the user access model is built. To perform anomaly detection, the *information management system* compares the received user activity data with the same user's model. If the behavior and model are sufficiently dissimilar, the behavior is identified as anomalous and appropriate action is taken, e.g., the user is required to re-enter the password, and an alert is sent to the system administrator or the portable system is disabled.

4 User Filesystem and Network Access Modeling

The MIA system is composed of four main processes: data collection, feature extraction, model construction, and anomaly detection. In this section we will describe each process in detail.

Data Collection. To capture the files access pattern, we modified the source code of FileMon [17] to log file access events transparently in real time. We gathered timestamps, names of the process responsible for access, locations of accessed files, and operations on the file. A file access record is generated on each system call, with a time resolution of 1 ms. The file access records generated by system services, not the user, are filtered out to save transmission energy and reduce noise in classification. To capture the network activities we used the WireShark 0.995 network packet analyzer [18] to gather the following data for each real-time network access event: timestamp of each access, source IP address, destination IP address, protocol identifier, and detailed packet information. Broadcast and router-specific protocols were filtered out in order to focus on user-dependent network events for the same reasons described above.

Feature Extraction. In order to choose the most robust features to model user behavior, and to reduce the amount of data in each log file, and based on our observation that in general, a normal user tends to use a small set of processes within a small subset of directories, connects to a small subset of the available machines/sites, and performs similar activities every day, we choose the *process*, *time*, and *location* fields to build the user file access model and *destination IP*, *time*, and *protocol* fields to build the user network access model.

Model Construction. There are two major challenges to developing the MIA system: determining the best data representation and selecting the appropriate data mining algorithm.

Data Representation. Due to the limitation of the software that we use for model construction [19], we need to map some of the fields to integer values such as *location*, *source IP*, and *destination IP* fields.

The directory structure can be represented as a tree, where the leaves are files and intermediate nodes are directories. We number the directories in each level from left to right, starting from 1. We use W to represent the width of the directory tree, i.e., the largest value used to number the directories at any level of the tree. For convenience, we represent the location of a file as follows:

$$\overrightarrow{x_n} = (x_1, x_2, \cdots, x_n) \stackrel{\text{def}}{=} x_1/x_2/\cdots/x_n. \tag{1}$$

where $x_1, x_2, \cdots, x_n$ are the integers corresponding to the sub-directories encountered sequentially on the path from the top-level directory to the file location. We use the following one-to-one mapping function

$$f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n-1} x_i \times W + x_n. \tag{2}$$

Note that distortion is unavoidable when we use linear functions to express the nonlinear relationship between file locations. To map the IP address (denoted as $A.B.C.D$) into an integer value, the following equation is used:

$$IP\ \ Value = A \times (255)^3 + B \times (255)^2 + C \times (255) + D. \tag{3}$$

K-Means Clustering. Although previous work used neural networks for intrusion detection [20,21], we choose K-means clustering [22] to build the models of filesystem and network behavior. This decision is based on a small-scale (four-user) three-month user study on the network-based technique. For this study, we compared the detection results based on two different data mining algorithms, a K-means clustering algorithm with the total of 8 clusters and a maximum of 20 iterations, and a three-layer sigmoid activation function neural network. With three months of training data, the accuracy of the neural network based approach did not exceed 84%, while the accuracy of the K-means clustering based approach increased to 98% within 90 days. We conclude that the K-means clustering algorithm allows more accurate prediction given a reasonable amount of training data (more than 14 days in our experiments). In addition, the K-means method is often the fastest and most energy-efficient for large data sets.

We used SPSS Clementine [19], a commercial data mining and statistical analysis software, as our model construction environment because it has built-in support for K-Means clustering. The input fields are automatically rescaled to have values between 0 and 1 before the clustering process such that each field is weighted equally.

Anomaly Detection. For an accurate anomaly detection, the main idea is to discover K clusters such that records within the same cluster are similar to each other but distinct from records in other clusters. The algorithm first selects K initial cluster centers. For each record, it computes the Euclidean distance between the record and the K cluster centers and assigns the record to the closest cluster. After all records have been processed, the cluster centers are updated to reflect the new set of records assigned to each cluster, and the records are checked again to see whether they should be reassigned to a different cluster. This record assignment and cluster update process is repeated until the change between two consecutive iterations is below a certain threshold.

One interesting problem is how to define the difference between the model and the current behavior without previous knowledge of anomalous behavior.

We define the *distribution vector* (DV) of a data set as a K-element vector, the ith element of which is equal to the percentage of records in the ith cluster. Our solution is based on the following observation: given a set of recent file access records generated by a normal user, the DV obtained from these records, i.e., $DV(evaluation)$, should be close to that obtained during the training phase, i.e., $DV(training)$, if the normal user behaves consistently during both the evaluation and the training phases. We compute the Euclidean distance[2] between $DV(evaluation)$ and $DV(training)$ and compare the result with a predefined

[2] We also tried using Manhattan distance but this yielded poorer accuracy.

threshold ϵ. This threshold is user-dependent and is set to a distance that results in a low false reject rate (FRR) in the training phase. If the distance exceeds ϵ, the model reports anomalous behavior. Otherwise, the current behavior is identified as normal behavior. Intuitively, a low threshold value would result in low false acceptance rate (FAR) at the cost of high FRR, while a high threshold value corresponds to a low FRR and a high FAR. In our approach, ϵ is chosen such that the FRR value does not exceed 10% during the training phase. ϵ is not determined by FAR value because we do not have access to the hacking data during the training phase in a real system with MIA deployed.

5 Energy Consumption Design Decisions

Portable systems such as laptops are usually battery powered. Since the batteries may be charged infrequently, MIA system should be energy-efficient. In this section, we analyze the energy consumption of our proposed system in a client-server architecture containing client laptop computers equipped with Intel 3945ABG WLAN cards. The parameters used for energy consumption estimation are listed in Table 1. They are measured using Power Manager, a built-in tool for power laptop consumption measurement [23], or derived from datasheets [16].

Table 1. Energy Consumption Parameters

Parameter	Explanation	Nominal Value
$P_{collect}$	average power consumption of data collection	0.15 W
P_{build}	average power consumption during model construction	17.53 W
$P_{anomaly}$	average power consumption in anomaly detection	22.45 W
P_{tr}	transmitting power for WLAN card	1.8 W [16]
E_{bat}	total battery energy	303.3 kJ
$E_{prepare}$	average energy consumption of feature extraction	4.6 J
$E_{compress}$	average energy consumption of data compression	2.8 J
E_{Total}	total energy consumption by portable device	34.675 kJ
$B_{wireless}$	uploading bandwidth of a wireless channel	1.5 Mb/s
T_{build}	time to build the user's model	540 s
$T_{anomaly}$	time to cluster new records in anomaly detection	20 s
T_{day}	total number of seconds in a day	86,400 s

Energy Consumption On the Client. When the network is available, data are sent to the server for anomaly detection. Therefore, the total daily energy consumption on the client is

$$E_{Total} = E_{collect} + E_{prepare} + E_{compress} + E_{transmission} \tag{4}$$

When the wireless network is not available, the anomaly detection is performed on the portable device, resulting in a daily client energy consumption of

$$E_{Total} = E_{collect} + E_{prepare} + E_{compress} + E_{anomaly} \tag{5}$$

Data Collection. The daily energy consumption for data collection is

$$E_{collect} = P_{collect} \times T_{day} = 0.15\,\mathrm{W} \times 86,400\,\mathrm{s} = 12.96\,\mathrm{kJ} = 4.27\% E_{bat}. \quad (6)$$

Feature Extraction. We perform a customized feature extraction process on the client that reduces the log file size by 50% and consumes 4.6 J on average. Since the log file has to be sent to the server every 5 minutes (see Section 6), or 288 times per day, the total daily energy consumption in the feature extraction process is 1.324 kJ. This takes only 0.436% of E_{bat}.

Data Compression. We used WINRAR, a software program that uses prediction by partial matching (PPM) algorithm [24], to compress the log file. On the average, data compression reduced the file size to 11% of its original size and consumed 10.15 J every 5 minutes. Therefore, the energy consumption of the data compression process in a day is 2.923 kJ which equals 0.963% of E_{bat}.

Data Transmission. In our experiment, we used an Intel 3945ABG 802.11 WLAN card for data transmission. Our experimental data indicate that the size of raw log files associated with both filesystem and network events increases at a rate of 0.064 MB/s on average. After feature extraction and compression, the file size is reduced to 5.5% of its original value. The energy consumption per transmission is

$$E_{transmission} = P_{tr} \times \frac{File\_Size}{B_{wireless}} = 1.8\,\mathrm{W} \times 0.055 \times \frac{0.064\,\mathrm{MB/s} \times 8 \times 300\,\mathrm{s}}{1.5\,\mathrm{Mb/s}} \quad (7)$$

$$= 10.137\,\mathrm{J}. \quad (8)$$

Therefore, the data transmission consumes 10.137 J every 5 minutes, and the total daily energy consumption of data transmission is 2.919 kJ which is 0.96% of E_{bat} per day.

In conclusion, when the network is available, the proposed technique consumes a total daily energy of

$$E_{Total} = 12.96\,\mathrm{kJ} + 1.324\,\mathrm{kJ} + 2.923\,\mathrm{kJ} + 2.919\,\mathrm{kJ} = 20.126\,\mathrm{kJ} = 6.635\% E_{bat}. \quad (9)$$

Design Considerations. We also analyzed the energy consumption for model construction and anomaly detection to determine whether they should be performed on the server or battery-powered clients. There are trade-offs between the energy consumed during feature extraction, compression, data transmission, model building, and anomaly detection that must be considered in order to arrive at an energy-efficient design.

Model Building. If the model is built on the client, the energy consumption during model building using one week of training data is

$$E_{build} = P_{build} \times T_{build} = 17.53\,\mathrm{W} \times 540\,\mathrm{s} = 9.466\,\mathrm{kJ} = 3.12\% E_{bat}. \quad (10)$$

Note that this model is built based on only one week of data. In a real system with the proposed MIA system deployed, we need two months of data to construct

the model to achieve an FRR of less than 10%, which will result in a significant increase in total energy consumption. Although it might appear that building the model on the client would reduce energy consumption by eliminating the transmission of training data to the server, the same log data would still need to be sent to server every 5 minutes for anomaly detection (see Section 6). Therefore, we decided to build models on the server.

Anomaly Detection. The energy consumption by each run of the anomaly detection process is

$$E_{anomaly} = P_{anomaly} \times T_{anomaly} = 22.45\,\mathrm{W} \times 20\,\mathrm{s} = 449\,\mathrm{J}. \tag{11}$$

Given 5-minute detection latency, the total daily energy consumption of anomaly detection is 129.3 kJ, i.e., 42.6% of E_{bat}. Therefore, performing anomaly detection on the server greatly increases battery life, thus motivating our design decision. However, intermittent anomaly detection on the client may still be required when the wireless network is unavailable, although this may increase energy overhead.

6 Experimental Evaluation

In this section, we evaluate the MIA system. Our experiments are designed to determine whether filesystem and network behaviors can provide sufficient information for fast anomaly detection. We first describe the experimental setup for gathering user data. Then, we explain how to generate data sets for training, testing, and evaluation. We then indicate the FARs and FRRs when we use the filesystem and network based re-authentication techniques separately. Finally, we evaluate the proposed combined technique. Experimental results indicate that we can achieve an FAR of 13.7% and an FRR of 11% with a detection latency of 5 minutes by combining filesystem and network based re-authentication techniques.

Experimental Setup for Data Collection. Since the data collection process is related to the participants' privacy, recruiting volunteers for this experiment was a challenge. We were able to recruit 16 volunteers, of which only 8 successfully completed the experiment. The filesystem and network monitoring programs were deployed on each participant's machine, which runs in the background generating log files without user intervention.

Data collection is divided into two phases: training phase and attacking phase. In the training phase, participants were asked to use their machines normally so that we could gather the training data to build a model of normal behavior for each user. We monitored the users' normal filesystem and network activity continuously for two weeks. The size of the log files generated in the training phase ranges from 80 MB to 25 GB.

The goal of the attacking phase is to obtain anomalous behavior data for each machine. Note that attacking must be done on the same set of machines used in the training phase, otherwise users and attackers can easily be distinguished

because the files they access will have different file names. Before the attacking phase starts, we embedded several files that contain fake bank account information in each participant's machine, without indicating the locations of those files. Each user was assigned to attack another user's machine at random. In the attacking phase, the attacker was asked to discover as much private information on the victim machine as possible, e.g., account information, frequently visited web pages, and information about user's friends and usual contacts.

Evaluation Process. After data collection, we first partition each user's log files, generated during the training phase, into two disjoint parts: training logs and testing logs. For each user, we sampled the records in the training logs randomly, which are then used to build the model of normal behavior. We then randomly chose 20 data blocks from the training logs, each of which contains consecutive file access records generated during a T minute time window, where T was set to 2,5,10, and 20 minutes. Each data block is then used as a training data set. The testing data sets and attacking data sets (represented as *evaluation data sets*) are generated in the same way. To avoid biasing the results, we took out the records associated with the files that contain fake bank account information.

As we described in Section 4, we first build a model of normal behavior for each user along with the distribution vector $DV(model)$. After that, for each training data set associated with a given T value, we assign the records in the set to the existing clusters, which generates $DV(training)$. We then compute the Euclidean distance between $DV(model)$ and $DV(training)$ to measure how much the training data set deviates from the model of normal behavior. The comparison threshold ϵ is chosen such that the FRR value does not exceed 10% during the training phase. Finally, we calculate $DV(evaluation)$ for each evaluation data set. If the Euclidean distance between $DV(model)$ and $DV(evaluation)$ is above ϵ, the model indicates an anomaly. Otherwise, it indicates normal behavior. We compute FARs and FRRs based on the results from the 20 testing data sets and 20 attacking data sets. Based on our experiment, we concluded that T =5 minutes provides the most appropriate FRR of 11%.

Using Filesystem/Network Based Technique Alone. Figure 2 shows the FARs and FRRs for eight users with an anomaly detection period of 5 minutes using filesystem and network based approaches separately. As indicated in Figure 2, the FAR ranges from 0% to 65% for the filesystem-based technique and 26.7% to 94.1% for the network-based technique. The FRR ranges from 8% to 28% for the filesystem-based technique and 0% to 20% for the network-based technique. The average FAR and FRR are 28.8% and 15.6% for the filesystem-based technique and 46.4% and 7% for the network-based technique. The results indicate that single-metric techniques have high FARs because most of users have some time periods within which there are either insufficient network or filesystem data to accurately identify an attack. For example, the filesystem model for user #7 has an FAR of 65%. A closer examination of user #7's data reveals that the corresponding attacker rarely accessed files. More specifically, 12 of the

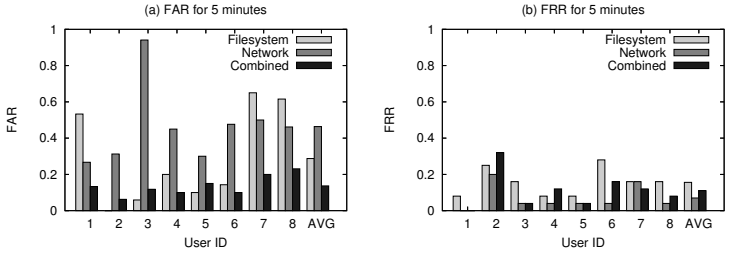

Fig. 2. (a) FARs and (b) FRRs for eight users with an anomaly detection time of 5 minutes using filesystem and network based techniques separately and jointly

20 attacking data set periods yielded no filesystem access data. This made it impossible for the filesystem activity based technique to accurately detect attacks during these periods. However, there is substantial network data for those same periods. Note that the FRRs are not significantly affected by the way we chose the evaluation data sets. In conclusion, using one technique alone results in high FARs due to intermittent use of, e.g., network or filesystem. However, using multiple metrics has the potential to overcome this problem.

Combining Filesystem and Network Based Techniques. If both the filesystem and network activity based re-authentication techniques are available, the online re-authentication manager can potentially combine them to improve detection accuracy. This improvement has two sources: (1) when the amount of data from one data source is insufficient for anomaly detection but data are available for the other source, the re-authentication manager can use the re-authentication technique with sufficient data for accurate classification and (2) when both filesystem and network are active, it is possible for the individual techniques to produce contradicting results. In this case, we may still reach the correct decision most of the time by carefully combining the decision variables derived from the two metrics.

We evaluate the combined approach for the 8 users to determine whether a multiple-metric re-authentication technique improves anomaly detection accuracy relative to single-metric techniques. Note that we ruled out the periods within which neither network nor filesystem is active because attacks generally use at least one of these. For each 5-minute period under evaluation, we first determine if only one type of data is available. If so, we use the single metric associated with that type of data, as described above. If both types of data are available, we first normalize the distance of each technique to that technique's threshold value. The aggregate distance is then computed using a weighted average of the normalized distances for the different metrics. An aggregate distance of greater than one during the testing phase (no attacks) results in a false reject and an aggregate distance less than one during the attacking phase implies a false accept. The multiple-metric re-authentication method is otherwise accurate.

Figure 2 shows the FARs and FRRs for the combined approach given 5-minute detection latency. The FAR ranges from 6.25% to 23.1%, while the FRR ranges

from 0 to 32%. The average FAR and FRR are 13.7% and 11%, respectively. The results indicate that the multiple-metric (combined filesystem and network) technique reduces the FAR by a significant amount compared to using either the filesystem-based or network-based technique alone.

The data for some users shown in Figure 2 merits further explanation. Attacker #3 did not produce enough network data for identification, resulting in a FAR of 94.1% when the single-metric network-based technique was used. However, since there are sufficient data for the filesystem model to accurately detect the attack during these periods, the multiple-metric approach achieves a FAR of 11.8%. On the other hand, attacker #1 actively uses the network but has some time periods with little filesystem access. As a result, the network model achieves a better FAR (26.7%) than the filesystem model (53.3%). In this case, the multiple-metric approach also outperforms either single-metric technique, achieving an FAR of 13.3%. When both filesystem and network are active, e.g., user #6, the combined approach also improves the FAR to 10%. The FRR of the combined approach is similar to that using either single-metric technique. We conclude that the multiple-metric approach outperforms single-metric re-authentication techniques. Note that additional metrics could potentially be included in the framework without fundamental changes. Although an FRR of 11% implies that a user may need to re-enter the password every 90 minutes, this is far better than the alternative necessary to achieve the same level of security, i.e., forcing the user to manually re-authenticate every 5 minutes.

Sensitivity Analysis. When both filesystem and network are active, instead of computing the average value of normalized distances from both techniques, we can favor one technique by assigning them different weights when averaging their normalized distribution vector distances. We tried a number of different weight combinations. Figure 3 illustrates the relationship between the average FAR and FRR and the weight assigned to network-based technique (the sum of the two weights is fixed at 1). As the network weight increases from 0 to 1, the FAR fluctuates within the range from 12.3% to 18.3%, while the FRR fluctuates within

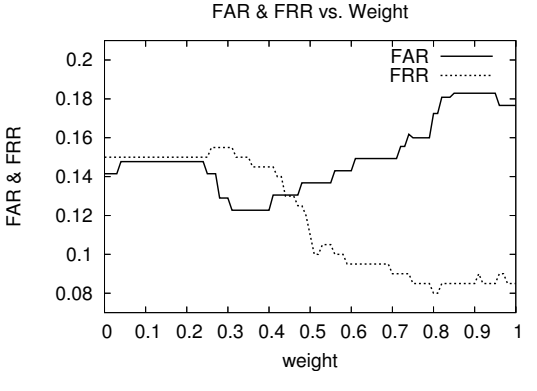

Fig. 3. Sensitivity of FAR and FRR relative to weight combinations

the range from 8% to 15%. The FAR reaches the minimum value of 12.3% with weights ranging from 0.31 to 0.4, while the FRR reaches the minimum value of 8% with weights ranging from 0.8 to 0.81. We conclude that FAR and FRR are not highly sensitive to weight combinations for moderate weights; using equal weights yields good results.

System Comparison. Designing a real-time implicit user re-authentication system is challenging due to the trade-off between accuracy and detection latency. A number of implicit user re-authentication techniques based on behavioral data have been proposed. However, all of them have their own advantages and limitations. Table 2 lists the FAR, FRR, detection latency, and limitations of numerous implicit user re-authentication schemes. Keystroke dynamics based techniques usually require long training and testing times. In addition, the user is generally required to enter a particular phrase, making the technique inappropriate for implicit re-authentication in many circumstances. Re-authentication based on mouse movement is application-dependent because a user may have different mouse behavior for different applications [6]. Pusara also found that combining multiple data sources can increase authentication accuracy and decrease detection latency. However, her conclusion can benefit from additional support: it was based on a short (four hour) training period. In contrast, we believe filesystem and network events are appropriate data sources for implicit re-authentication. It is difficult to conduct an attack while generating neither filesystem nor network activity. The proposed multiple-metric network and filesystem based approach achieves a reasonable FAR and FRR with a detection latency of 5 minutes.

Table 2. Characteristics of Different Implicit User Re-authentication Schemes

Scheme	Data Source	(FAR, FRR)	Latency	Limitation
[5]	keystroke dynamics	(N/A, 10%)	N/A	1. structured text instead of arbitrary text is favored 2. fail when the attackers do not generate keystroke inputs
[15]	mouse movements	(1.75%, 0.43%)	17.6 minutes in the worst case, 4.5 minutes on average	1. only deal with data from a single application. 2. fail for users who do not use the mouse
[6]	keystrokes, mouse movements, and GUI events	(1.78%,14.47%)	2.2 minutes	1. high FRR 2. small training data set and testing data set
Ours	filesystem and network events	(13.7%, 11%)	5 minutes	relatively high FAR for short (2-week) training period

7 Conclusions

Portable battery-powered computer systems are highly-susceptible to unauthorized access. The proposed implicit re-authentication system provides a way to protect sensitive information on these computers from attack without inconveniencing the user. However, designing a system that can accurately detect attacks on portable computers by implicit real-time monitoring of user behavior, e.g., filesystem and network accesses, is challenging. Portable computers have limited

battery energy, limited performance, and potentially-intermittent wireless network access. In this paper, we have proposed and evaluated a software architecture and user identification algorithms for implicit re-authentication on portable computers. The proposed system is able to distinguish between normal use and attack with an accuracy of approximately 90% with a detection latency of 5 minutes for network activity and 10 minutes for file access.

8 Future Work

Filesystem and network activities are the main data sources used for model construction and anomaly detection in this paper. Therefore, the MIA system is not valid for systems in which users don't have frequent access to the network, or are not based on the filesystem for daily activity, such as cellphones and PDAs. to overcome this limitation, we are considering the use of spatial and temporal information in addition to the filesystem and network activities to improve the accuracy and decrease detection latency.

References

1. HP iPAQ Pocket PC h5500 User Guide. Hewlett-Packard Company, `http://bizsupport.austin.hp.com/bc/docs/support/SupportManual/lpia8006/lpia8006.pdf`
2. Gupta, P., Ravi, S., Raghunathan, A., Jha, N.K.: Efficient Fingerprint-Based User Authentication for Embedded Systems. In: Proc. Design Automation Conf. (June 2005)
3. Aaraj, N., Ravi, S., Raghunathan, A., Jha, N.K.: Architectures for Efficient Face Authentication in Embedded Systems. In: Proc. Design, Automation & Test in Europe Conf. (March 2006)
4. Richardson, R.: 2007 Computer Crime and Security Survey. Computer Security Institute, Tech. Rep (2007), `http://i.cmpnet.com/v2.gocsi.com/pdf/CSISurvey2007.pdf`
5. Monrose, F., Rubin, A.: Authentication Via Keystroke Dynamics. In: Proc. Conf. on Computer and Communications Security (April 1997)
6. Pusara, M.: An Examination of User Behavior for Re-authentication. Ph.D. dissertation, Center for Education and Research in Information Assurance and Security, Purdue Univeristy (August 2007)
7. Moon, Y.S., Leung, C.C., Pun, K.H.: Fixed-Point GMM-based Speaker Verification Over Mobile Embedded System. In: Proc. Multimedia Workshop in Biometrics Methods and Applications (November 2003)
8. Gu, H., Zhuang, Y., Pan, Y., Chen, B.: A New Iris Recognition Approach for Embedded System. In: Wu, Z., Chen, C., Guo, M., Bu, J. (eds.) ICESS 2004. LNCS, vol. 3605, pp. 103–109. Springer, Heidelberg (2005)
9. Lane, T., Brodley, C.E.: Temporal Sequence Learning and Data Reduction for Anomaly Detection. ACM Trans. on Information and System Security 2(3), 295–331 (1999)
10. Denning, D.E.: An Intrusion-Detection Model. IEEE Trans. Software Engineering 13(2) (1987)

11. Wolf, T., Mao, S., Kumar, D., Datta, B., Burleson, W., Gogniat, G.: Collaborative Monitors for embedded System security. In: Proc. Wkshp. of Embedded System Security (October 2006)
12. Li, Y., Wu, N., Jajodia, S., Wang, X.S.: Enhancing Profiles for Anomaly Detection Using Time Qranularities. J. of Computer Security 10(2), 137–157 (2002)
13. Wagner, D., Dean, D.: Intrusion Detection Via Static Analysis. In: Proc. Symp. of Security and Privacy, pp. 156–169 (2001)
14. Denning, D.E., Neumann, P.G.: Requirements and Model for Ides–A Real-Time Intrusion Detection System. In: SRI International, Tech. Rep (1985)
15. Pusara, M., Brodley, C.E.: User Re-Authentication Via Mouse Movements. In: Proc. Wkshp. of Visualization and Data Minining for Computer Security (October 2004)
16. Intel PRO/Wireless 3945ABG Card Specification, Hewlett-Packard Company, `http://h18019.www1.hp.com/products/quickspecs/12510na/12510na.PDF`
17. Russinovich, M., Cogswell, B.: FileMon for Windows 7.04 (2006), `http://technet.microsoft.com/en-us/sysinternals/bb896642.aspx`
18. WireShark 0.99, `http://www.wireshark.org`
19. Clementine 11.1. SPSS Corporation, `http://www.spss.com/clementine/index.htm`
20. Ghosh, A., Schawrtzbard, A.: A Study in Using Neural Networks for Anomaly and Disuse detection. In: Proc. USENIX Security Symp. (1997)
21. Mukkamala, S., Janoski, G., Sung, A.: Intrusion Detection Using Neural Networks and Support Vector machines. In: Int. Joint Conference on Neural Networks (May 2002)
22. Hartigan, J., Wong, M.A.: A K-means Clustering Algorithm. J. of Applied Statistics (1979)
23. Thinkpad power manager, Lenovo Group Ltd., `http://www-307.ibm.com/pc/support/site.wss/MIGR-61583.html`
24. Cleary, J., Witten, I.: Data Compression Using Adaptive Coding and Partial String Matching. IEEE Trans. on Communications 32 (April 1984)