IJICT International Journal of Information & Communication Technology

ITRC

Volume 3- Number 2- March 2011 (67-71)

Technical Note

Masquerade Detection Using GUI Events in Windows Systems

Parisa Kaghazgaran
Dept. of Computer Engineering & IT
Amirkabir University of Tehren (AUT)
Tehran, Iran
Parisa.kaghazgaran@gmail.com

Babak Sadeghyan
Dept. of Computer Engineering & IT
Amirkabir University of Tehren (AUT)
Tehran, Iran
basadegh@aut.ac.ir

Received: December 28, 2010 - Accepted: February 14, 2010

Abstract— **Masquerade attack in computer systems refers to the illegitimate user activities while pretending to be legitimate user. Detection of such attacks is done by discovering significant changes in user's behavior based on his profile. Profile is built by data produced from mouse, keyboard and other devices. In this paper we propose a practical approach for collecting GUI data and deriving useful parameters included both mouse and keyboard events from Windows OS. We model user identification and masquerade detection as a binary classification problem. Profiling and user classification is accomplished by use of Support Vector Machine (SVM) algorithm. Feature vectors are fed to SVM. The output is behavioral pattern which builds the profile. System is trained by normal behavior and detects deviations from profile. According to the results of implementation the proposed approach ensure detection rate up to 94% with few false alarm.**

Keywords- Masquerade detection; Intrusion detection; Anomaly; Profile; Detection rate; GUI; Behavior.

I. INTRODUCTION

Masquerading is the process where a person spoofs someone else's identity and utilizes privileges he is not entitled to [1]. Masquerading or impersonation is one of most dangerous attacks; not only because of hard detection but also attacker undermines sensitive information [2]. Masquerader gets access to legitimate user account either due to that the victim left his terminal open or his password was disclosed somehow. Masquerade attacks can be carried out by either organization's insider or outsider. Outsider attacks can be detected by signature-based IDSs but masquerading is lethal when an insider penetrates systems. In next section different kind of masquerade attacks are introduced. We would propose a new approach to detect second type of insider attacks.

Insider attacks can be detected only when current system behavior deviates from normal profile considerably. So at first we should build user's profile. The IDS should learn normal behavior. Our IDS collects data from user sessions and extracts features to construct profile.

There have been several attempts to take the problem of masquerade detection. A progressive research area is discovering new methods to decrease false alarms generated by IDS. This aim will be accessible with providing comprehensive data sets.

2005 PITAC report shows that apart from virus and worm attacks, the insider threats are rapidly rising, roughly at the rate of 20% [4].

In GUI[1] based systems most of user activities are derivative from mouse and keyboard events. As command line data are not able to capture GUI data; future attempts need to combine various types of data to detect effectively.

[1] Graphical User interface

[Downloaded from journal.itrc.ac.ir on 2024-04-20]

Our goal is to increase masquerade detection rate in GUI based systems. Similar work for Linux has already been done in [2]. Although academic environments aim to encourage using Linux but people at industrial and commercial milieu, like organizations, are willing to work with Windows operating system; so we have chosen our target to be Windows OS, and have implemented and evaluated our proposed approach.

We collect data from both mouse and keyboard events. To our knowledge, no profiling technique is available in literature for capturing both mouse and keyboard events in windows systems.

The presented IDS in this paper include following characteristics:

1. It is Host-based IDS and source of data is operating system audit data.
2. It uses Anomaly technique for detection. Masquerader activities are mostly seemed authorized to the system, and might not have a known signature; so the misuse detection is not suggested for it.
3. It utilizes machine learning algorithm (SVM) for learning and classification.
4. Its Architecture obeys centralized structure because IDS components are stood in one system.
5. Analysis Time is online, means Audit data immediately after being collected are analyzed. In other words analysis is done during connection.
6. The implementation is software based.

A. *Paper organization*

This paper is organized as follows: In section III, we review masquerade detection concept and the related works. In section IV, our approach will be explained. Section V describes results. Finally in section VI we outline conclusions, future works; and open problems.

B. *Our Contribution*

We can summarize paper contribution in proposing a new approach to detect masquerading.

II. MASQUERADE DETECTION CONCEPT AND RELATED WORKS

Generally there are two types of masquerade attacks against an organization. Masquerading can be performed by insiders or outsiders. Outsiders are outside the organization and can access to victim's machine remotely through network and steal information or cause damage to the system [1]. On the other hand there are two types of insider attacks. First; a legitimate user misuses his privileges for malicious or unauthorized purposes [3] and gets access to resources which is not authorized to access them. Second; an insider impersonates another user inside the organization. The former one can be controlled by access control mechanisms. In latter one; most of masquerader actions are technically legal to the system; so it is more difficult to identify such violations; Also attacker has enough knowledge about system and victim behavior that he/she can avoid detection for a long time.

Figure 1 shows masquerading attacks. Our masquerade detection appraoch attempts to detect second type of insider attacks which is highlighted in the figure.

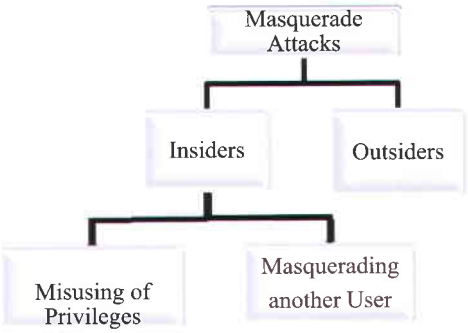

Figure 1. Masueurade Attacks Taxonomy

Behavioral path concept is expressed according to [1]. Behavioral paths are based on users' activities on the system and show the amount of effort to do a task. The probabilistic paths for different users are drawn in figure 2. Activities may contain mouse events, keyboard typing speed, number of commands executed, and time spent to do a function but are not limited to these.

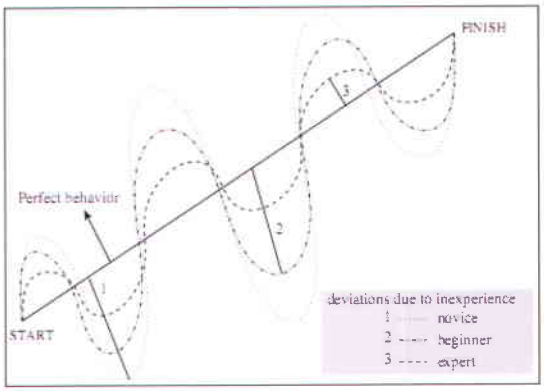

Figure 2. Behavioral Path for Various Types of Users (Garg & others 2006 [1])

As it can be seen from the figure, user's behavior depends on their skill levels. An expert one will be more towards the perfect behavior[2], because of his experience and knowledge of using the system. A novice user will have great violations from perfect behavior due to lack of experience and knowledge about system and applications. Numbers 1, 2 and 3 show deviations from perfect behavior due to users' skill levels. Although expert path shows shortest time with minimum step to complete the task, it will not touch perfect behavior ideally. To do the same process, various users will utilize system differently, thus different behavioral path will be generated.

This fact is similar to finger print which is unique for everyone; so behavioral profile based on GUI events is unique for each user. There are some related works on masquerade detection.

[2] The perfect behavior is defined as minimum effort level or system activity involved in achieving a task [1].

[Downloaded from journal.itrc.ac.ir on 2024-04-20]

A. Masquerade Detection based on command line data

There have been many techniques to handle masquerade detection problem. In [3], [5], [6] UNIX commands collected at command line are used for masquerade detection. The assumption is: type, number and sequence of commands to complete a task are different for each user. The drawback of these methods is that they are not able to capture user interactions, so they fail in GUI based systems.

B. Masquerade Detection based on GUI data

In [7] manipulation of windows, icons and menus are profiled rather than capturing all system events. This method has several disadvantages: first, the profiling seems to be manual instead of being an automated process; second this method is applicable for specific application because windows, menus and icons are different among various applications; third, the time, as an analysis factor, is not considered while it is important factor for an intrusion analysis.

In [1] [8] [9] the process table and GUI events details for windows OS are used to build profile. In [1] user profile is built from only mouse movements and events. Its drawback is that only few GUI events were captured. In [9] the mouse data is used to profile users. [1] uses SVM algorithm while [9] uses decision tree algorithm.

In [2] data is collected from GUI activity in Linux. Since most of Linux users do the tasks by entering commands [11], only the GUI data is not sufficient; so a mixed approach is suggested.

III. OUR APPROACH

We explain our approach into two phases: Profiling Users and Masquerade Detection.

A. Profiling Users phase

This phase has two steps: data collection and feature extraction. To collect raw data from user interactions with system we need a tool to log the events. So we have implemented an active system logger using Microsoft .NET framework 3.5 and C# language on windows XP systems. The logger gathers data from mouse movements and events and keyboard events.

1) Data Collection

We have used Hook mechanism to capture Data. A hook is a mechanism by which a function can intercept events (messages, mouse actions, keystrokes) before they are received by an application. The function can act on events and, in some cases, modify or discard them. The system supports many different types of hooks; each type provides access to a different aspect of its message-handling mechanism. We utilized Windows API Hook.

Since users should not notice the logging of their activities, the logger runs as Windows service. Windows service program runs automatically while system is booting without interface and does processing at system backgrounds.

To evaluate our framework, we collected data from three different users through several sessions. [1] and [2] collect data from 3 and 4 users respectively. We eliminated some peripheral factors; for example users work with same mouse and keyboard.

2) Feature Extraction

After collecting raw data, it is turn to extract useful parameters to construct feature vectors. Each vector involved several values. These values are parameters described the user behavior. The Features can be classified into two classes: mouse events and keyboard events.

- The features we considered for mouse events are:
  - Mouse clicks (lc, rc, mc): the average number of left, right and middle clicks per user session.
  - Distance (d): the average distance traveled by mouse between two clicks.
  - Time (t): the average time between two clicks.
  - Mouse event angle (θ): the angle of mouse event relative to X-axis
  - Mouse speed (s): the average speed of movement for entire session.
  - Mouse event location (x, y): coordinate of mouse.
- Keyboard events features are:
  - Key pressed (kp): the average number of keystroke per session.
  - Key shortcuts (ks): the average number of key shortcuts per session.

Since mouse can move a few pixels in millisecond, data gained from mouse movements are huge. In [3] to overcome this problem data is logged each 100 milliseconds. But this solution has some other limitations, for example during 100 milliseconds mouse moves 100 pixels then return 60 pixels near to its initial location, so 100-60=40 pixels is considered as distance traveled by mouse and speed is calculated as 40/100= 0.4 pixel per millisecond, but this assumption is not precise. To solve this problem our solution is to store data according to location instead of according to time. So coordinates of mouse locations are logged which distances between points are at least 20 pixels so mouse speed is toward actual speed. Hence, we have avoided storing enormous mouse movement data.

The profiling phase architecture and relation between components are shown in figure 3.

[Downloaded from journal.itrc.ac.ir on 2024-04-20]

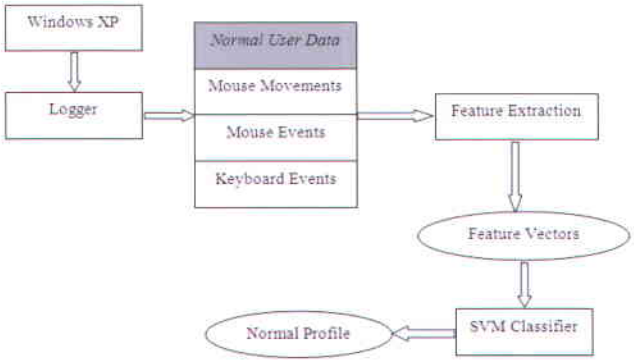

Figure 3. Profiling Pahse

B. *Masquerade Detection phase*

Detection phase has two steps: feature calculation and using SVM for learning and classification.

1) *Feature calculation*

As session's durations are not equal, it is not reasonable to do our calculations based on features acquired from entire session, so we have used sliding window concept.

If the session's starting time is t=0, and we calculate the features for 50 seconds period, then the first period would be [0, 50] and the second will be [50,100] and so on. Hence the session is divided into 50 second periods. After finishing first step, we advance the window for 5 seconds and assume the second step starting time to be t=5, and calculate the features for the following periods: [5,55] , [55,105] and so on. After each step we advance window for 5 seconds. We calculated the values for 10 steps. After 10 steps the pattern repeats itself. According to number of periods in each step we calculate average and standard deviation of features. The advantage of this technique is clear when we use SVM. In other words, a SVM vector is built in each step. We obtain 10 vectors for one session, so the useful parameters independent of the session duration construct 10 vectors. The total number of features are 10 (lc, rc, mc, d, t, θ, s, (x,y), kp, ks) and we calculate mean(m) and standard deviation (sd) for all of them per step. This gives a total of 20 unique features as: 10 * (m ,sd)=20.

It is necessary to mention that our session's durations are between 20 to 25 minutes. (users were not willing to cooperate more than 25 minutes.)

2) *Applying SVM*

Classification is one of application of machine learning. There are several methods for classification in machine learning. We chose to use SVM due to advantages listed in [1].

We used two class SVM [10] because data vectors are labeled as positive (normal) or negative (masquerader). We collected GUI based data sets for 3 users which are called A, B, C, through several sessions. Each user's sessions are divided into two parts: training and testing sessions.(referred to table I) . As mentioned earlier we obtain 20 features from each session. To build the profile and test our approach, first we assume user A is an authorized and users B & C are masqueraders for A. The data vectors are fed to SVM to train normal behavior to system. The output of algorithm is the normal behavioral pattern (profile for A). Then the test data vectors are fed to SVM and are classified by SVM function according to the profile. Detection Rate, False Positive (FP) and False Negative (FN) are indicated in the next section.

In next step we assume user B is an authorized and users A & C are masqueraders for B. In other experiment we assumed user C is an authorized and users A & B are masquerader for C.

The detection phase architecture and relation between components are shown in figure 4.

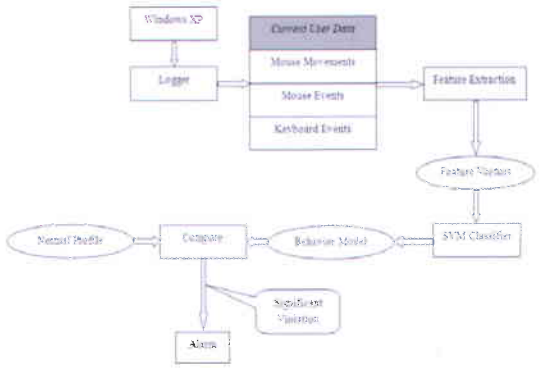

Figure 4. Detection Phase

IV. RESULTS

The captured data set was split for training and testing as shown in table 1:

TABLE I. USERS' SESSIONS

User	#Training Session	#Testing Session
A	5	4
B	5	4
C	4	3

Table 2 shows results of paper approach when we use only mouse events.

TABLE II. RESULTS ONLY WITH MOUSE EVENTS

Model	Detection Rate	FP
A-(B,C)	85%(94/110)	12
B-(A,C)	80%(88/110)	14
C-(A,B)	94%(104/110)	5

Table 3 shows results when we use both mouse and keyboard events.

TABLE III. RESULTS WITH BOTH OF MOUSE AND KEYBOARD EVENTS

Model	Detection Rate	FP
A-(B,C)	88%(97/110)	10
B-(A,C)	85%(94/110)	11
C-(A,B)	99%(108/110)	2

As the results show detection rate increased in table 3 in comparison with table 2. In conclusion, user

[Downloaded from journal.itrc.ac.ir on 2024-04-20]

behavioral pattern is not limited to one type of GUI data, so if we have more features we can achieve more significant results. It is noteworthy that FP rate is more than FN because we are using anomaly detection technique; as we know FP in anomaly detection is high.

Due to lack of a comprehensive data set for GUI events; it is not reasonable to compare achieved results with other related works, but the results of [1] are given below:

TABLE IV. [1] RESULTS

Model	Detection Rate	FP
A-(B,C)	92.31%(240/260)	20
B-(A,C)	85.77%(223/260)	37
C-(A,B)	96.15%(250/260)	10

As it can be seen the number of data vectors in [1] is more than ours. (260 vectors is available in [1] while we have 110 vectors), so naturally their data set is more detailed. But detection rate in table 3 and table 4 are rather similar and in model (C-A, B) our approach work better.

V. CONCLUSIONS

We have designed a new framework for capturing GUI events in Windows. Then we evaluated our framework. We have extracted useful parameters to construct features vectors then used SVM to train the system and test the users by both mouse data only and combination of mouse and keyboard data.

Our results demonstrate that using more features is better and detection rate will be higher than using only features of mouse events.

Since we were able only to collect data from 3 users with limited sessions, as part of future works we plan to test our system with more users and more sessions so we hope that results become more significant.

Also we tested our system with limited set of data. This is due to the fact that there are no public GUI based data sets available.

In conclusion one of the important open problems in GUI based authentication is to collect a comprehensive GUI data set and publish it.

VI. REFERENCES

[1] Garg, A., Rahalkar, R., Upadhyaya, S.: Kevin Kwait "Profiling Users in GUI Based Systems for Masquerade Detection." In: Proceedings of 7th Annual IEEE Information Assurance Workshop, United States Military Academy, West Point, New York , pp.48-53, 2006.

[2] Bhukya W, Kommuru S, Negi A. Masquerade detection based upon GUI user profiling in linux systems, Advances in Computer Science–ASIAN Computer and Network Security, springer, pp.228-239, 2008.

[3] Coull SE, Szymanski BK. Sequence alignment for masquerade detection, Computational Statistics & Data Analysis, Elsevier, Vol. 52, No. 8, pp. 4116-4131, 2008.

[4] Benioff MR, Lazowska ED, Cyber Security: A Crisis of Prioritization, President'S Information Technology Advisory Committee Arlington VA.. 2005.

[5] Kim HS, Cha SD. Empirical evaluation of SVM-based masquerade detection using UNIX commands, Computers & Security, Elsevier, Vol.24, No.2, pp. 160-168, 2005.

[6] Bhukya, W.N., Kumar, S., Negi, A.:"A study of effectiveness in masquerade detection" IEEE TEN CON 2006 14-17, pp. 1–4 Digital Object Identifier , 2006.

[7] Imsand, E.S., Hamilton Jr., J.A.:"GUI Usage Analysis for Masquerade Detection." In: Proceedings of IEEE, Information Assurance Workshop (IAW), United States Military Academy, West Point, New York,pp.270-276, 2007.

[8] Li, L., Manikopoulos.:"Windows NT One-class Masquerade Detection." In: Proceedings of IEEE,Information Assurance Workshop (IAW), United States Military Academy, West Point, New York,pp. 82-87, 2004.

[9] Pusara, M., Brodley, C.:"User Re-authentication via mouse movements". In: Proceedings of the ACM Workshop on visualization and data mining for computer security, Washington D.C., USA, pp.1-8, 2004.

[10] CHRISTOPHER J.C. BURGES Bell Laboratories, Lucent Technologies, " A Tutorial on Support Vector Machines for Pattern Recognition", Data mining and knowledge discovery, Springer, Vol. 2, No. 2, pp. 121-167, 1998.

[11] Leibovitch E. The business case for Linux. Software, IEEE, 2002.

Parisa Kaghazgaran is a M.Sc. student in information security at the Amirkabir University Technology. She received her B.Sc. degree in 2009 from the same university. Since 2008, she has been working in the area of intrusion detection systems and privacy preserving IDS's.

Babak Sadeghiyan is an associate professor in the department of computer engineering and information technology at the Amirkabir University of Technology since 1993. He received his B.Sc. degree in Electronics Engineering from Isfahan University of Technology in 1985. He received his M.Sc. degree in Electronics Engineering from Amirkabir University of Technology in 1989. He received his Ph.D. degree from University of New South Wales, Australia in 1993. His research interests include cryptology, cryptographic protocols and intrusion detection systems.

[Downloaded from journal.itrc.ac.ir on 2024-04-20]