Hybrid Method for Detecting Masqueraders Using Session Folding and Hidden Markov Models

Román Posadas[1] Carlos Mex-Perera[1] Raúl Monroy[2] Juan Nolazco-Flores[3]

[1] Center for Electronics and Telecommunications, ITESM, Campus Monterrey
Av. Eugenio garza Sada 2501 Sur, Col. Tecnologico
Monterrey, N. L., CP 64849 Mexico
[2] Computer Science Department, ITESM, Campus Estado de Mexico
Carretera al lago de Guadalupe, Km. 3.5, Estado de Mexico, CP 52926, Mexico
[3] Computer Science Department, ITESM, Campus Monterrey
Av. Eugenio garza Sada 2501 Sur, Col. Tecnologico
Monterrey, N. L., CP 64849 Mexico
{A00790428, carlosmex, raulm, jnolazco}@itesm.mx

Abstract. This paper focuses on the study of a new method for detecting masqueraders in computer systems. The main feature of such masqueraders is that they have knowledge about the behavior profile of legitimate users. The dataset provided by Schonlau *et al.* [1], called SEA, has been modified for including synthetic sessions created by masqueraders using the behavior profile of the users intended to impersonate. It is proposed an hybrid method for detection of masqueraders based on the compression of the users sessions and Hidden Markov Models. The performance of the proposed method is evaluated using ROC curves and compared against other known methods. As shown by our experimental results, the proposed detection mechanism is the best of the methods here considered.

1 Introduction

A masquerader is a person that uses somebody else's computer account to gain access and impersonate a legitimate user to complete a malicious activity. Masquerade detection is usually undertaken using an anomaly detection approach, which aims at distinguishing any deviation from ordinary user behavior. A number of different methods of masquerade detection that use the Schonlau *et al.* dataset SEA [7] have been published [1, 3]. The SEA dataset consists of clean and contaminated data of 50 users. This collection was obtained using the *acct* auditing mechanism, it consist of 15000 UNIX commands without arguments for each of 70 users. Then, 50 users were randomly selected as intrusion targets, so that the remaining 20 users were used as masqueraders and their data was interspersed into the data of the other 50 users. The normal behavior *profile* of each user is obtained from the first 5000 commands, which are legitimate. This

is the *training data*. The last 10000 commands is the *testing data*, since it may contain masquerade information.

The original SEA dataset presents some problems when more realistic conditions are considered. The users who were labelled as masqueraders did not have knowledge about the behavior profiles of the users to be impersonated. Besides, they typed UNIX commands as they usually do in a normal working session. Thus, the data used as masquerade sessions is not intended to impersonate the users' activity. Moreover, it is possible to find masquerade sessions in SEA with very simple and repetitive sequences that can be easily identified even by human inspection.

To overcome these difficulties, we modified SEA using synthetic sessions as masquerade sessions. The new sessions were created using the knowledge of the behavior profile of the user to be impersonated, the modified dataset is here referred as *SEA-I* and it resembles a situation where the masquerader avoids detection mechanisms in a smarter manner.

A simple strategy was used to synthesize the masquerade sessions, we considered the knowledge of the commands frequency seen at the training phase of each legitimate user. The resultant sessions were generated following the same probability distribution of the commands of the legitimate user.

In this paper we proposed a new hybrid detection method, it is composed of two main parts, the first one is a session folding mechanism and the second is a detection mechanism based on Hidden Markov Models (HMM). The session folding is a compression stage that substitutes with labels the more relevant sequences derived from the extraction of the user grammar at the training phase. Once the test folded sessions are obtained, these are passed to the HMM based detector.

To evaluate the proposed detection method we made experiments using SEA and SEA-I datasets. The performance obtained was compared against some known methods described in the next section.

2 Overview of Methods of Masquerader Detection

In this section we describe some known methods of masquerader detection from other authors: uniqueness[2] and two others proposed by Latendresse [3].These methods have been used in this paper for comparison purposes. We evaluated the performance of such methods and the proposed one against SEA and SEA-I datasets.

A masquerade detection method is said to be *local*, if the normal behavior of a user is only based on the user's legitimate data. If the normal behavior of a user is also based on data from other users, it is said to be *global*. Since they are more informed, global detection methods are usually more accurate than local ones. However, demand more computational effort.

A given method performs *update* if, during the detection phase, the data used to classify is confirmed to be legitimate and it is added to the user profile. The update may also be local or global.

2.1 Uniqueness

Uniqueness [2] is based on the fact that commands not previously seen in the training data may indicate a masquerade session. This method extracts global statistics, that is the user profiles are based on the other users data. In this method, the fewer users that use a particular command the more indicative that a user that entered that command is a masquerader.

This method uses what is called popularity, which is an indicative of how many users use a particular command. A command has popularity i if only i users use that command. Almost half of the commands appearing in the training part of the dataset are unique with popularity one and it represents 3.0% of the data.

Uniqueness's detection model is a session score, x_u, given by:

$$x_u = \frac{1}{n_u} \sum_{k=1}^{K} W_{uk} \left(1 - \frac{U_k}{U}\right) n_{uk}$$

where

n_u is the length of the testing data sequence of user u;
n_{uk} is the number of times user u typed command k in the testing data;
K is the total number of distinct commands;
U is the total number of users;
U_k is the number of users who have used command k in the training data;

where the weights W_{uk} are

$$W_{uk} = \begin{cases} -v_{uk}/v_k, & \text{if user u}'\text{s training data contains command } k, \\ 1, & \text{otherwise,} \end{cases}$$

where $v_{uk} = N_{uk}/N_u$ and, $v_k = \sum_u v_{uk}$.

N_u is the length of the training data sequence of user u;
N_{uk} is the number of times user u typed command k in the training data.

From the above formulas it is easy to see that temporal ordering of commands in a given testing sessions is ignored. The thresholds used for plotting the ROC curve can be obtained from [7]. We did not apply update for this method.

2.2 Customized Grammars

Latendresse [3] developed an intrusion detection system based on grammar extraction by the Sequitur algorithm created by Nevill-Manning and Witten [4]. By extracting hierarchical structures from a sequence of commands and generating a context-free grammar capable of building that sequence, he constructed user profiles based on the training data of SEA. Once he obtained the grammar

(production rules) that represents the training data of each user, he computed the total frequency of its expansion for that user and also the frequency of that same expansion for the other users, that is the *across* frequency.

For the purpose of this paper we used the version of the experiment where no global scripts are used and the evaluation is based on the frequency of the commands of the user and across all users. As an additional variation we did not apply update.

The production evaluation function used was

$$e(p) = l_p \frac{f_p}{f_p + \frac{F_p}{k}}$$

where

p is a production of the session.
l_p is the length of the expansion of production;
f_p is the frequency of the expansion of the production;
F_p is the across frequency of the expansion of the production;
k is a tuning constant.

Finally, let F be the set of productions found in a session s, then the evaluation of s consists of the sum over all productions of s, in symbols: $\sum_{p \in F} e(p)$. This method just like Uniqueness makes use of global profiles where the normal behavior of a user is also based on data from the other users.

2.3 Command Frequencies

Another method shown by [3] where only repetitive single commands were used was also evaluated. This is a simpler version based on the frequencies of the commands without global scripts and without taking into account their temporal ordering. Because the length of the productions is equal to 1, that is, single commands, then the evaluation function shown below is obtained.

$$v(c) = \frac{f_c}{f_c + \frac{F_c}{k}}$$

where

f_c is the frequency of the command c;
F_c is the across frequency of the command c among all other 49 users;
k is a tuning constant.

The sum over all commands of a session gives the score of the session. Also it is a global profile method.

3 The proposed method.

We propose a local hybrid method that uses compression and hidden markov models for masquerade detection. With this method we take the training sessions of a given user and use the Sequitur algorithm [4] to extract his grammar and make a compressed version of all the sessions. This is accomplished by substituting the sequences found by the algorithm for production rules. This process is referred to as *session folding*. These compressed sessions are what we use as training data for the HMM model. Figure 1 shows an architecture of the detection mechanism.

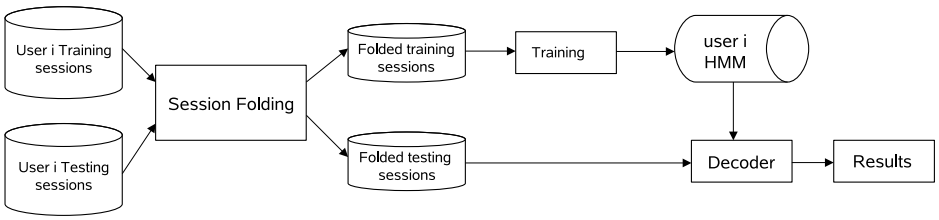

Fig. 1. Proposed detection mechanism architecture

The session folding block works as shown in Figure 3. The Sequitur algorithm constructs a context-free grammar from the training sessions. The grammar extracts hierarchical structures that generate the sequence of training commands, see Figure 2. For testing purposes we fold the test session with the grammar obtained from the training. Once the testing session has been folded it is then evaluated by the trained HMM model to get the probability that the session was generated by the model.

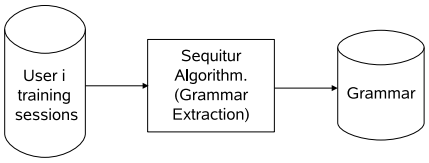

Fig. 2. Shows the grammar extraction steps

The way the sessions are folded is based on the priority of the grammar symbol [6], that is the production. This priority is computed taking into account the length of the production rule, the frequency of that production in the training

data and also on the total amount of data in the training sessions. The priority of the grammar symbol (rule) is obtained as follows:

$$P = \frac{l * f}{N}$$

where:

P is the priority of the grammar symbol;
l the size (length) of the grammar symbol;
N the total number of commands in the training data; and
f is the frequency of the grammar symbol.

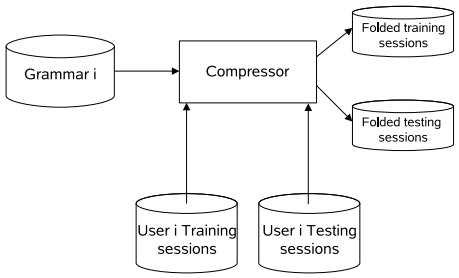

Fig. 3. Session folding architecture

Sequences of commands are substituted for production rules extracted from the Sequitur algorithm. Sequences with higher priority are preferred over those with a lower priority. The folded session may also contain single commands, since those may not be substituted for any production rule.

As depicted in Figure 1, the last stage of the detection mechanism is based on a HMM. As defined by [5] an HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols.

In an HMM one or more starting and finishing states are specified. Possible transitions are made successively from a starting state to a finishing state, and the relevant transition probability and symbol output probability can be multiplied at each transition to calculate the overall likelihood of all the output symbols produced in the transition path up to that point. When all transitions are finished, the HMM generates a symbol sequence according to the likelihood of a sequence being formed along each path. In other words, when a sequence is given, there is one or more transition paths that could have formed the sequence, each path have a specific likelihood that the sequence was formed by it. The sum

of all the likelihoods obtained for all such transition paths is regarded as the likelihood that the sequence was generated by the HMM. It must be mentioned that this is a local profile method.

4 Synthesis of Masquerade Sessions

In this section we present how the motivation of avoiding intrusion detection mechanisms takes the masquerader to create synthetic sessions that follow the same behavior profile of the legitimate user.

Original SEA dataset contains 15000 UNIX commands for each of 50 users, the first 5000 commands are legitimate, however in the last 10000 we may have some commands entered by a masquerader. In order to make data more tractable the user data is divided in blocks of 100 commands, each block is treated as a session. The first 50 sessions are not contaminated and thus constitute the *training dataset*. The last 100 blocks of each user may or may not be masquerade blocks. A block is either totally contaminated or legitimate from the user. Masquerading data was inserted using this rule: if no masquerader is present, then a new masquerader appears in the next session with a probability of 1%. Otherwise, the same masquerader continues to be present in the next block with a probability of 80%. SEA dataset comes with a matrix that shows where the masquerade sessions are located for each user.

SEA has been very popular to test masquerader detectors, however masquerade sessions in SEA come from working sessions of other users who had no intention to act as intruders. This is, they did not masquerade their activities.

In order to avoid detection, the intruder must act like the legitimate user, that is, he must know the victim's work profile. Here we propose a scenario where the intruder has knowledge of the legitimate user behavior and uses it to build a session with the victim's profile, this way the intrusion will not be detected, since it has the same legitimate user's customs.

Following this philosophy an intruder in order to avoid detection mechanisms may build a session with the same statistical properties than those found in the training data. Then once the command frequency properties of one user were extracted, a masquerade session can be built having the same probability distribution than the probability distribution of the training data. Masquerade sessions for all users were created with this procedure. Sequences of commands (scripts) were not taken into account when building the masquerade sessions.

Once we created the masquerade sessions for all users, they were located in exactly the same positions as the original dataset. This builds **SEA-I dataset**. In fact only the masquerade sessions were modified from SEA. All the masquerade sessions of any particular user were created taking into account only the command frequency properties of that specific user.

Now we want to know the performance of our proposed method and the other three previously mentioned methods against SEA and SEA-I datasets.

5 Experiments and Results

We need a way for comparing the performance of the methods. This can be accomplished by the usual Receiver Operating Characteristic (ROC) curves, which are parametric curves generated by varying a threshold from 0% to 100%, and computing the false alarm (false positive) and missing alarm (false negative) rate at each operating point. The false alarm rate is the rate at which the system falsely regards a legitimate user as a masquerader, while the missing alarm rate is the rate at which the system falsely regards a masquerader as a legitimate user. In general, there is a trade-off between the false alarm rate and missing alarm rate. The lower and further left a curve is, the better it is.

As we can see in Figure 4 different curves have different performance in distinct regions of the graph. We are interested in having a low false alarm rate and a low missing alarm rate. In Figure 4 we show the results of our proposed method against the other three for SEA dataset. Command Frequencies seems to be the best of all four methods. This method is global since it uses information of all other 49 users to detect masquerade sessions of a single user. The constant k has a value of 7.

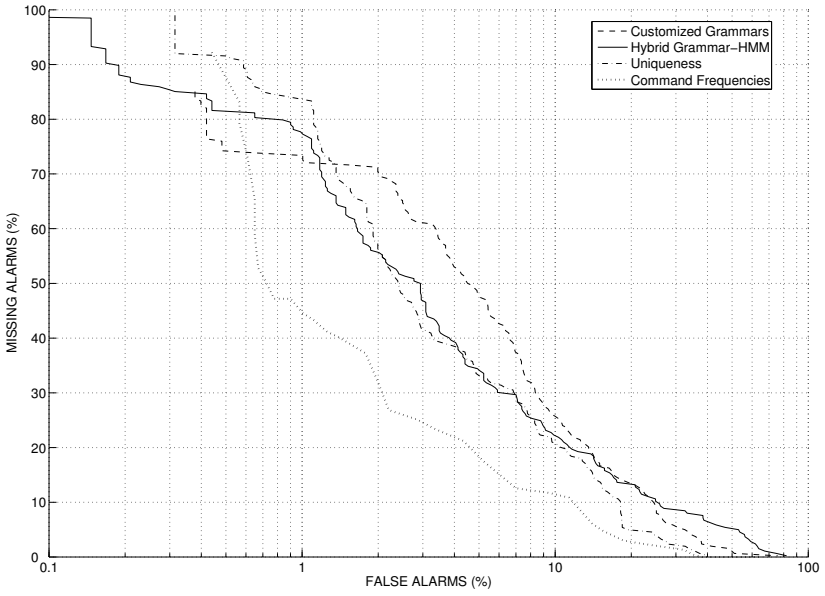

Fig. 4. Performance of detection methods with SEA dataset

In Figure 5, we show the performance of the four methods when SEA-I is used. The upper and further right the curve it is, the worse the performance the method has. Methods like Uniqueness and Command Frequencies seemed to be very good with original SEA but they are severely affected with the modified dataset no matter they have the added value of being global methods. The reason of the poor performance of Uniqueness and Command Frequency is caused by the fact that they are customized methods based on the commands frequency property of the users sessions, then they are incapable of detecting masquerade sessions created using that same frequency property. Our Hybrid Grammar-HMM method outperforms the others since it is the most further left located and always has the least false alarms variation for almost all the missing alarms range. The second better method is Customized Grammars, like ours also uses grammar extraction but its false alarm rate is bigger because the only mean to test command sequence properties is by the grammar extraction itself.

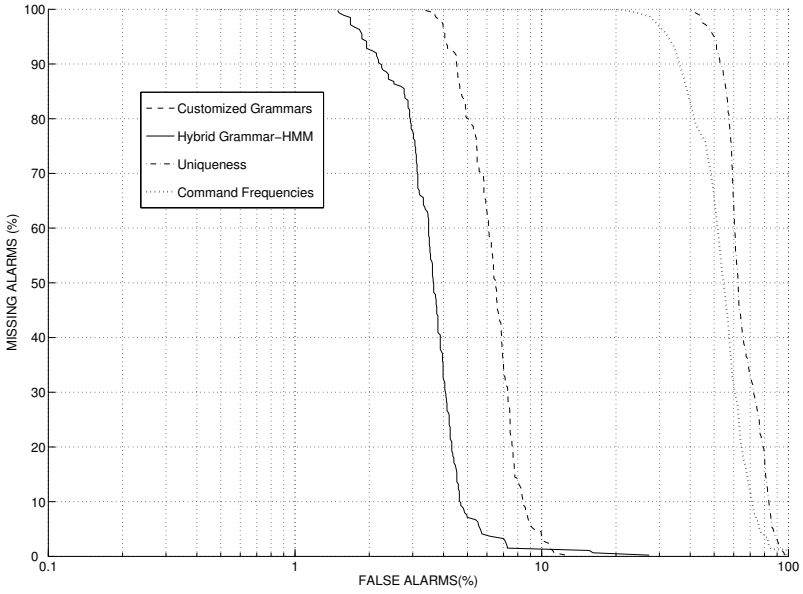

Fig. 5. Performance of detection methods with SEA-I dataset

The reason for this good behavior of the proposed method is based on the fact that the initial session folding phase compresses the sessions based on grammar (scripts) properties. The less grammar symbols appear in the folded session the

more indicative that the session is from a masquerader. The probability that the sequence found in the test folded session comes from the legitimate user, is found by the previously trained HMM model.

6 Conclusions

In this paper we show the performance of an hybrid method that detects command-frequency based masquerade sessions in a UNIX-like system that outperforms best global profiles methods. This method has a good performance with original SEA dataset, even though it uses local profiles.

The incorporation of session folding based on grammar extraction and hidden markov models makes this detection method more robust and with a wider range of use. As a pending job, near future masqueraders detectors should be able to detect intruder sessions when these are based on both, frequency and sequence properties of a legitimate user profile.

7 Acknowledgments

The authors would like to acknowledge the Cátedra de Biométricas y Protocolos Seguros para Internet, ITESM, Campus Monterrey and Regional Fund for Digital Innovation in Latin America and the Caribbean (Grant VI), who supported this work.

M. Latendresse provided the data for the ROC curve of the Command Frequencies method.

References

1. Schonlau, M., DuMouchel, W., Ju, W., Karr, A., Theus, M., Vardi, Y.: Computer Intrusion: Detecting Masquerades. Statistical Science **16** (2001) 1-17.
2. Schonlau, M., Theus, M.: Detecting masquerades in intrusion detection based on unpopular commands. Information Processing Letters **76** (2000) 33-38
3. Latendresse, M.: Masquerade detection via customized grammars. In Julisch, K., Krügel, C., eds.: Second International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Volume 3548 of Lecture Notes in Computer Science., Springer(2005) 141-159
4. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: a linear-time algorithm. Journal of Artificial Intelligence Research, JAIR 7 (1997) 67-82
5. Rabiner, L.R., Juang, B.H.: An introduction to Hidden Markov Models. IEEE ASSP Magazine **3** (1986) 4-16
6. Godínez, F. Hutter, D., Monroy, R.: Audit File Reduction Using N-Gram Models (Work in Progress). In patrick, A., Young, M., eds.: Proceedings of the Ninth International Conference on Finantial Cryptography and Data Security, FC'05. Volume 3570 of Lecture Notes in Computer Science., Roseau, The Commonwealth Of Dominicana, Springer-Verlag (2005) 336-340

7. Schonlau, M.: Masquerading used data. (Matthias Schonlau's home page) http://www.schonlau.net.