Masquerader Detection Based on Normal User Activity Variation

Peng Jia
Center for Automated Learning and Discovery
School of Computer Science
5000 Forbes Ave.
Pittsburgh, PA 15213
Email: pengj@cs.cmu.edu
Tel: 412-268-3422

Abstract

Masqueraders are people impersonate actual users on a computer system. Finding masqueraders using anomaly detection technique includes building user profile using only user's normal activity data, comparing current user behavior with the normal user profile and finally triggering alarm when current user behavior deviates significantly from his/her profile. When being provided only command sequence accumulated over time of each user, previous research has focused on detecting masquerader using the frequency property and order property of the data. However, their detectors act pretty much the same: they either all succeed in triggering the alarm or all fail to trigger the alarm. Focusing on different properties of the sequence data may provide means to solve this problem. This paper proposed to use normal user activity variation to build up the user profile for masquerader detection. The relevant measures as well as the ways to implement and evaluate these measures are discussed.

1. Introduction

Timely detection of computer system intrusion is a problem that is receiving increasing attention. While firewalls and access control are being used to keep intruder from breaking into a system from outside an organization's network, 80% of intrusions or attacks are from within an organization [10]. People have the access of a system, whether they are legitimate or illegitimate, usually knows the layout of the system and where the valuable data is, thus potentially are bigger threat to the system security.

However, no techniques can prevent legitimate user from abusing their rights in a computer system. Thus automatic and real-time detection based on user auditing information is viewed as the ultimate defense of insider intrusion.

Anomaly detection refers to methods that are able to detect intrusions without knowing intrusion patterns. The only information available to an anomaly detection system is the normal user activity data. Usually, a historical profile of the user is built based merely on normal data. Based on the premise that a disguised intruder's behavior must to some degree be different from a normal user's behavior, a classical anomaly detection system compares current user activities with the user profile. Any significant deviation from the profile may indicate an intrusion of the system and will cause the detection system trigger the alarm.

While a variety of information such as CPU usage, I/O usage, command names and its options, timestamp etc. can be obtained from a system auditing mechanism, lots research has focused on anomaly detection using only UNIX command sequences, where the only information a detector can have is the user id and the command names the user has used. Given the difficulty to collect real intrusion data, synthetic data are often used in anomaly detection research. Several researches have been conducted used masquerade data ----- normal user data contaminated partially at random using other normal user's data.

Detectors have been built for these synthetic masquerader data. The experimental performance of these detector indicates that more work wait to be done to get better profiling of user behavior. Moreover, different perspectives to look at the data should be taken in designing the detector so that diversified opinion among detectors can be achieved. This diversification is very important for building a detecting system where detectors specialized on different aspects of the data can vote for whether to trigger alarm and thus increase the overall accuracy of intrusion detection. This voting mechanism is not realizable because available detectors give us similar predictions distributions.

This project will work on design a new perspective to look at the data. The success of the design indicates the possibility to increase the prediction accuracy of intrusion detect system by mining hard for hiding features of system auditing data.

2. Prior work

Researchers from both statistical area and computer science area have implemented various methods to conquer the problem of masquerader detection. The research particularly focuses on intrusion detection using UNIX command sequences are reviewed or briefly mentioned below.

[1][6] have tried method called uniqueness. This method based its judgment on the intuition that commands not previously seen in training data is a good indicator of intrusion happened in the testing data, where the rarer the command has been seen in a community of users, the higher the indicating power of the command. They designed a testing score that increases with the number of unpopular commands a user used in the testing data but rarely or not used in the training data. A higher score trigger the alarm of possible intrusion. Although this method is based on a simple intuition, it obtained good results compare with other more complex detectors experimented on the same set of data. This implies that frequency property of the command sequence is an important indicator of intrusion.

Not as uniqueness method that only considered command frequencies, Bayes First-order Markov by [2] and [4] utilized the information of 1-step command transition probabilities. Transition matrices were built for each user's training data and testing data and the detector trigger the alarm when there is considerable difference between the training data transition matrix and testing data matrix. This method has comparable alarming accuracy as that of uniqueness method. Thus it discloses the importance of the order property of command sequence while only 2-order sequences are considered.

A higher order Markov, although has the potential to be the basis of a more accurate detector, is being avoided in anomaly detection because of its intimidating computational cost. A more efficient method proposed by [1] is a hybrid model that combines higher-order model with an independence model. The higher-order model is used when user's testing data does not have too much unobserved data. It focuses on a subset of the most used commands and thus only need to deal with significantly reduced dimensionality. The independence model is utilized when the test data contain many commands unobserved in the training data. Although this model, by its higher complexity, brings more characteristics of the data into detection consideration, it does not achieve higher detection accuracy. In fact, it is even inferior to uniqueness model and first-order Markov model when it is being tested on the same set of data.

Sequence Match is a method proposed by [4]. A similarity measure is used in comparison of a command sequence with sequences in a user profile. A similarity score lower than a threshold will trigger the alarm. This method, when tested using the same data used in [1] and [2], generated accuracy result no better than the above methods.

In Incremental Probabilistic Action Modeling (IPAM) of [3], a 1-step command transition probabilities matrix is used to predict the new command given the new command's previous command. Too many bad predictions indicate a possible masquerader intrusion. This method has similar accuracy as sequence matching.

Other major methods mentioned in anomaly detection literature include: decision tree, neural network, expert system, Hotelling's T^2 test, Chi-square multivariate test and compression method. All the detection methods, except for compression, have the common characteristic that they do not utilize very complex mathematical measures. Instead, simple statistics such as frequencies of the commands and 2-gram probabilities are the main focus.

However, UNIX command sequences nevertheless have complex structure and hard to decode because they also account for intrinsic human behavior variation (i.e. users learn and use new commands) or environmental regularities[8]. [1] mentioned that any single method above is not sufficient to do the work given the fact that each of them only considers frequency property or order property or at best both properties of the data. The not very satisfying performance of those detectors (false alarms rate ranges between 1.4% to 6.7% and missing alarm rate range between 30.7 to 65.8.)as well as the fact that these detectors have similar alarming patterns (their distribution of detection failure or success are almost the same) indicate that it is necessary to look the data from different perspective to achieve better performance.

The approach proposed for this project attempts to quantify normal human activity variation and utilize this information for anomaly detection. The following section describes the experimental data as well as the approach to be used in this project.

3. Approach

The data

The data will be used in this research is semi-synthetic data collected on a UNIX system by the system's auditing mechanism. It includes 15,000 UNIX command sequences for each of 50 users accumulated overtime. The reason to call it semi-synthetic data is: the data of normal activities of each user is real data collected by the system auditing mechanism, whereas the data of masquerader intrusion is synthesized by inserting different (normal) user's data into the normal data randomly. Initially, about 70 users' activity data were collected. 50 of these users were selected randomly as the intrusion targets while the rest users were left as the potential masqueraders. The first 5,000 commands of each potential target are purely the corresponding actual user's normal activity data, whereas the left 10,000 has been divided into blocks and some of the blocks has been replaced with potential masquerader's command block with certain probability.

The task here is: Use the first 5,000 commands of each potential intrusion target as the training data to build a profile for the user. Test each blocks in the rest 10,000 commands and trigger the alarm when contaminated blocks are found, where a significant deviation of a user's current behavior from it built profile will trigger the alarm for possible masquerader intrusion.

The procedure

It is necessary to first describe the overall procedure we attempt to use for utilizing human activity variation for masquerader detection on a UNIX system.

The overall procedure is:

1. Divide the training data into blocks of size 100 (commands). Build a signature vector for each of these blocks. Where signature vector refers to a vector that summarized the key characteristic contained in a sequence block.
2. For each block (except block one), a distance from it to the previous block is calculated and the value is averaged across all the blocks in the training data. This average will be used as the threshold for anomaly detection in the testing data.
3. For each testing block, we compute its distance from the previous block. If the distance is above the threshold, the intrusion alarm will be triggered.
4. Update the user profile----re-compute the average distance by incorporate uncontaminated data (judge by the system) into the training data.
5. Repeat step 3 to 4 for each testing block.

The intuition behind this detection approach is: users of a UNIX system adapt their behavior continuously at a relative stable rate. Their behavior adaptation rate can be measured by the commands they use as well as the command frequencies within a certain period. When a masquerader intrude a system, he/she has high probability in failing to imitate an actual user' variation rate because he/she has high probability to use set of commands that familiar to him/herself. Therefore we can learn each user's variation rate and use the learned rate as the threshold to detect masqueraders.

First we need to compute signature vector for each block. Prior research [7] [1] has discovered that detecting based on frequency alone provides rather good performance with relatively low computational cost. Thus, this project plans to use frequency or probability distribution of each command within a block as the component of the signature vector. The signature vector will be built to take the following form

$$VF_b = (f_1, f_2, \ldots\ldots f_i \ldots f_n) \text{ or } VP_b = (p_1, p_2, \ldots\ldots p_i \ldots\ldots p_n)$$

Where b ----the serious number of the block,
n ----the total number of commands of the corresponding UNIX system,
f_i ----the number of times command i presents in block b
p_i ----the probability that command I presents in block b.
VF_b ----the frequency vector for block b
VF_b ----the probability vector for block b
$\Sigma f_i = 100$
$\Sigma p_i = 1$

The signature vector is expected to be a vector that contains a lot of zeros because the total number of commands of the corresponding UNIX system is at least twice of the block size 100. Whether to use VF_b *or* VP_b depend on the particular distance measures.

The next step is to compute the distance between blocks. In this project, several distance measures will be exploited. They are: Euclidean distance, cosine of the angle between V_i and V_j and a measure borrowed from information retrieval research documented in [9], where

information entropy is used in the calculation of vector distances. Different detection performances may result from these different distance measures. It will be interesting for us to look at these possible differences to infer the data's implicit characteristic. It may also be interesting to see the whether the combination of the results from different distance measures can achieve better detection performance.

a. Euclidean distance: for signature vector VF_a and VF_b, the distance $d(VF_a , VF_b)$ is

$$d(VF_a, VF_b) = [\sum_{i=1}^{n} (f_{ai} - f_{bi})^2]^{1/2}$$

b. Cosine of vectors of frequencies

$$d(VF_a, VF_b) = \frac{\sum_{i=1}^{n} (f_{ai} f_{bi})}{\sum_{i=1}^{n} (f_{ai})^2 \sum_{i=1}^{n} (f_{bi})^2}$$

This distance is identical to the correlation coefficient of the two sequences of frequencies.

c. Distance between probability vectors

$$d(VP_a, VP_b) = H(\frac{1}{2}VP_a + \frac{1}{2}VP_b) - \frac{1}{2}[H(VP_a) + H(VP_b)]$$
$$= \sum_{i=1}^{n} \frac{p_{ai} + p_{bi}}{2} \log \frac{p_{ai} + p_{bi}}{2} + \frac{1}{2}\sum_{i=1}^{n} p_{ai} \log p_{ai} + \frac{1}{2}\sum_{i=1}^{n} p_{bi} \log p_{bi}$$

where H(.) is the information entropy function.

The distance expressed in the above function decreases not only when the number of shared commands increase, but also when the term distribution of these vectors becomes more alike.

For each pair of adjacent command blocks in the training data, these three distance measures will be calculated. An averaging score for each of these three distance measures over the training blocks will be used as the thresholds for masquerader detection. Exponentially weighted moving average method will be used in calculation of the average. Thus later blocks will have higher weights in the average rate computation.

When detecting masquerader command blocks for the test data. Each test blocks' distances from its previous blocks are computed and compared with the threshold----the current average distance measure. A distance value higher than the averaged value will cause the alarm to be triggered.

The Evaluation

ROC (Receiver Operator Characteristic) curve will be use to evaluate the performance of the detector. Each point in an ROC carve indicates a pair of the hit (alarm when there is masquerader) rate and the false alarm (alarm when there is no masquerader) rate. Where:

hit rate = the total number of hits / the total number of intrusive events in the testing data.
false alarm rate = the total number of false alarms / total number of normal events in the testing data.

An ROC closer to the top-left corner of the chart indicates a better detection performance.

The detector design will be considered as successful if the experiment results in higher ROC compared to previous research on the same data, or if it results in a distribution of hits and false alarm significantly different from that of previous methods. While the former indicates a better profiling method, the later shows different perspective can result in different detector. A detector has different opinion makes a voting mechanism among various detectors meaningful.

4. Work plan

Feb. 8th---Feb. 16th: Process the data to prepare for implementation of the detection algorithm: Divide training data into 50 blocks for each user; For each block in the training and testing set (50*150 blocks), build the signature vectors.

Feb. 17th --- March. 25st. Implement the distance computation algorithms and perform training to get the threshold of average learning rates. Test the detector on test set.
March. 26th. --- April 10th Generate ROC curve and evaluate the proposed method. Make comparison of the three distance measure.

April. 10th --- May. 10. summarize the experimental results and wrap up the work for final paper

References

[1] M.Schonlau, W.Dumouchel, W-H. Ju, A.F. Karr, M.Theus and Y. Vardi. Computer Intrusion: Detecting Masquerades

[2] W. DuMouchel and M. Schonlau. A fast computer intrusion detection algorithm based on hypothesis testing of command transition probabilities.

[3] B.D. Davison and H. Hirsh. Predicting sequences of user actions. Presented at the AAAI-98/ICML'98 Workshop on Predicting the Future: AI Approaches to Time Series Analysis, Madison, WI, July 27, 1998 and published in Predicting the Future: AI Approaches to Time Series Problems, Technical Report WS-98-07, pp. 5-12, AAAI Press

[4] W. DuMouchel. Computer intrusion detection based on Bayes factors for comparing command transition probabilities.

[5] T. Lane and C.E. Brodley. Sequence matching and learning in anomaly detection for computer security. In Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management (1997), pp. 43--49.

[6] M. Schonlau and M. Theus. Detecting Masquerades in intrusion detection based on unpopular commands. Preprint submitted to information processing letters. 21 June. 2000

[7] N. Ye, X.Y. Li, Q. Chen, S.M. Emran and M.M. Xu. Probabilistic techniques for intrusion detection based on computer audit data. In press, IEEE Transactions on system, man and cybernetics.

[8] R.A. Maxion and K.M.C. Tan. Benchmarking anomaly-based detection systems. 1st international conference on dependable systems & networks: New York, NY 25-28 June 2000.

[9] S.D.M Wong and Y.Y. Yao. A statistical similarity measure. Proceedings of the tenth annual international ACM SIGIR conference on Research and development in information retrieval June 3 - 5, 1987, New Orleans, LA USA

[10] http://www.cerias.purdue.edu/coast/intrusion-detection/introduction.html

[12] Dorothy E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222-232, February 1987.

.