Intrusion Detection of Masqueraders Based On Data Mining and Soft Computing Techniques

Yingbing Yu*
Department of Computer Science & Information Technology, Austin Peay State University, 601 College Street, Clarksville, Tennessee 37044, USA.

* Corresponding author. Tel.: 1-931-2217826; email: yuy@apsu.edu
Manuscript submitted March 31, 2015; accepted April 25, 2015.
doi: 10.17706/jsw.10.9.1111-1118

Abstract: Many organizations face the critical threat of inside attacks from masqueraders who can be either disgruntled employees or external hackers by exploit legitimate user identity to manipulate the system. Intrusion detection systems (IDSs) are deployed to build the normal user profiles and then detect the possible deviation from the past behavior patterns indicating a possible illegal access. In this paper, we apply a profiling method based on user command sequences and apply the data mining technique Naïve Bayes classification to measure the degree of deviation. A fuzzy system is applied to integrate multiple commands execution to evaluate the overall threat of the possible masquerader existence.

Key words: Anomaly intrusion detection, data mining, soft computing, computer security.

1. Introduction

This paper investigates new methods to more effectively detect computer system intrusions. Intrusions into a computing system can be defined as any set of actions that attempt to compromise the integrity, confidentiality, or availability of a computer system resource. Integrity requires that data will not been changed inappropriately, whether by accident or by deliberate malign activities. Confidentiality refers to limiting information access and disclosure to the set of authorized users, and preventing access by or disclosure to unauthorized ones. Availability refers to the ability to readily access of information resources.

Intrusion detection has a significant role in the overall computer security architecture. R. Bace defines intrusion detection as the process of monitoring computer networks and systems for violations of security policy [1]. Computer security policy is the set of laws, rules, and practices that define the system boundaries (what is permitted and what is denied) and details exactly what operations are allowed [2]. In 1980, Anderson published a paper in which computer security threat problem was examined for the first time [3]. Denning's paper "An Intrusion Detection Model" [4] in 1987 provided a methodological framework that later inspired many research projects and commercial products.

Intrusion detection systems (IDSs) attempt to perform the process of monitoring computer networks and systems for violations of security policy. This definition does not include the prevention of intrusions from occurring, only detecting and reporting of intrusions. An intrusion detection system attempts to perform intrusion detection as defined above, which may involve a combination of software and hardware [5].

Misuse (knowledge or signature-based) IDSs look for specific patterns that define a known attack. The information about known attacks and vulnerabilities of a system is encoded into "signatures". Any actions that trigger the matches will be reported as "attempts" of intrusions. Anomaly (behavior-based) IDSs

assume the deviation of normal activities under attacks and perform abnormal detection compared with predefined system or user behavior reference model.

Anomaly IDSs have been widely used to build system or user behavior profiles to detect inside attacks from masqueraders, which is one of the major threats facing many organizations. Masquerader can be internal users or external intruders who exploit legitimate users identification and password that one may obtain illegally and then perform malicious attacks from the inside.

In the recent years "CSI/FBI Computer Crime and Security Survey", inside abuse of network access was the second most cited forms of attacks while virus incidents ranked the most attacks with no surprise [6]. To prevent a system from attacks due to identity theft, the effective approach is to monitor user behavior and report any suspicious activities. Alarms are alerted when a user behaves out of characters and a large deviation with the behavior profile is detected.

To distinguish a masquerader from genuine users is a challenging task due to the problem of concept drift, where the observed user behavior may change with different tasks, time, general knowledge level and such other uncertain elements [7]. In this paper, we introduce a data mining technique naïve Bayes classification to detect masqueraders based upon user command sequences generated. The naïve Bayes classification is applied to measure the degree of deviation from the normal behavior profiles built in the past. A fuzzy system is presented to integrate multiple commands execution to evaluate the overall threat of the possible masquerader existence.

The rest of this paper is organized as follows. Section 2 is the literature review. Section 3 presents the naïve Bayes classification to measure the degree of deviation. Section 4 introduces a fuzzy system to evaluate the overall threat evaluation of a sequence of commands execution. Experimental results are presented in Section 5. The paper concludes with Section 6.

2. Literature Review

2.1. Masquerader Detection Methods

Access control and authentication are not sufficient to prevent potential intrusions from masquerader which already got the authorization to access system resources by obtaining an authorized user identity illegally. Traditionally user behavior in a system is characterized by parameters such as login frequency, location frequency, last login, session elapsed time, password fails, location fails, amount of network traffic, resources used by user in a session and so on [8]. Users shell commands sequence concatenated by date order can be used to build behavior profiles for user classification, predict future behavior, and detect masquerades. Commands are still the predominating way to access UNIX systems. A genuine user shows the relatively static characteristic of command usage.

Machine learning and statistical methods have been widely used in the literature for the behavior profiling from the analysis of command sequences. Davison and Hirsh developed a model called IPAM (incremental probabilistic action modeling) to predict sequences of user actions in which single-step command transition probability is estimated from training data [9]. Balajinath introduced GBID (Genetic Based Intrusion Detector) to model individual user behavior with a 3-tuple vector which is learnt later by a genetic algorithm [10]. Ryan used a back propagation neural network NNID (Neural Network Intrusion Detector) to identify users simply by what commands and how often they use, called the 'print' of a user [11].

Lane and Brodley selected a machine learning algorithm IBL (instance based learning) to measure the similarity between the most recent 10 commands of a user and the profile extracted from the past [12]. The similarity measure is the count of matches of a new sequence with the sequences from a user's commands history, with a greater weight assigned to adjacent matches. Schonlau selected several statistics-based

methods to detect masqueraders, including uniqueness, Bayes one-step Markov, Compression, Multi-step Markov chain etc. [13].

2.2. Fuzzy Reasoning System

Fuzzy systems, including fuzzy logic and fuzzy set theory, provide a rich and meaningful addition to standard logic. Fuzzy logic is much closer to the human thinking and natural language than the traditional logical system to represent imprecise information. The central concept of fuzzy logic is the concept of fuzzy sets, which provides a systematic way for manipulation of vague and imprecise information. This concept is briefly reviewed in the following sections, followed by a discussion of fuzzy logic and inference system.

The concepts of fuzzy sets and fuzzy logic have been used in rule-based fuzzy systems, in which relationships between variables are represented by fuzzy IF-THEN rules. These systems have been successfully applied on fuzzy controllers and a wide range of consumer products. It uses the IF-THEN rules associated with the fact to extrapolate imprecise propositions.

Since the fuzzy rules can be considered as a relation defined on the universe of discourse, the composition of rules will be used in fuzzy reasoning as multiple antecedents. Fuzzy reasoning includes several different forms. The simplest form will just involve a single rule with single antecedent.

If *x* is A, then *y* is B.
associated with the fact
x is A′.
The obtained consequent is
y is B′.
Another form is a single rule with multiple antecedents
If *x* is A and y is B, then *z* is C.
associated with the fact
x is A′ and *y* is B′.
We can get the consequent
z is c′.
In practice, most application will involve multiple rules with multiple antecedents, which has the form of antecedent:
If *x* is A1 and y is B1, then z is C1.
If *x* is A2 and y is B2, then z is C2.
associated with the fact
x is A′ and y is B′.
The corresponding consequent is
z is C′.

Fig. 1 shows a typical fuzzy system, composed of an inference engine and a fuzzy rule base. The inference process includes the following five phases:

Find the degree of membership functions of facts with respect to each antecedent.

Combine the degree of membership unction in a rule using fuzzy operation.

Fire a rule to get the qualified consequent degree of membership.

Aggregate the consequent membership of the rules (multiple rules reasoning) to get the overall output degree of membership.

Defuzzification to extract a crisp value from the output membership function to replace a fuzzy set with a proper single value.

The inference engine performs an inference process by use of approximate reasoning which is the

backbone of the fuzzy system. The fuzzy rule base is the knowledge base, which consists of a collection of fuzzy IF-THEN rules. The inference engine based on the collection of fuzzy rules conducts a mapping from input fuzzy sets to output fuzzy sets. The fuzzy rules has the following form

R(k): IF x1 is A1K and *x*2 is A2k and … and *xn* is Ank, then *y* is Bk

for $k = 1, \ldots, N$, where $x1, \ldots, xn$, and y are linguistic variables, corresponding to input and output, respectively. A1K, …, Ank, and Bk are elements of the term-sets of linguistic variables values. A fuzzifier and a defuzzifier are used in order to obtain a system with crisp (non-fuzzy) inputs and crisp outputs. This is necessary in most real systems where the inputs and outputs are real-values variables.

The fuzzy system is used to predict the future behavior based on the past behavior. To model a real system using fuzzy inference mechanism, generally it should include the following steps:

Select input and output variables.

Choose a specific fuzzy inference system (Mamdani, TSK or Sugeno). For most of the linguistic application, it usually will select Mamdani fuzzy inference model, the input and output are all described by fuzzy membership function definition.

Define the linguistic terms associated with input and output variables.

Select membership functions for input and output variables that correspond to linguistic terms.

Design a collection of fuzzy if-then rules.

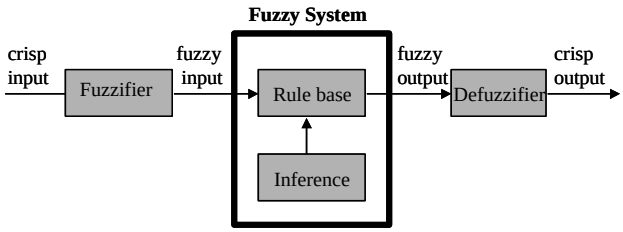

Fig. 1. General structure of a fuzzy logic system.

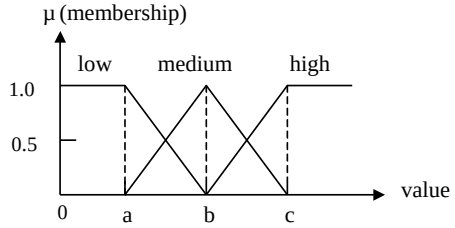

Fig. 2. Triangular membership definition.

3. Anomaly Detection Using Data Mining Techniques

To detect masqueraders, we analyze user activities in the form of commands sequences. This approach is more effective in UNIX/Linux system where most of the activities involve the shell commands execution. In this section, we present the data mining technique - naïve Bayes classification to capture the deviation of each user command to classify as normal from a genuine user or abnormal from masqueraders.

3.1. The Naïve Bayes Classification Model

Bayesian classifiers are statistical classifiers that are based on Bayes' theorem. They can be used to predict class membership probabilities that a given tuple belongs to a particular class. Naive Bayes classifier [10] is a simple probabilistic classifier which assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. The naïve Bayes classifiers are known for the inherent robustness to noise and the fast learning time. They have been used successfully in text classification where the task is to assign a document to a particular class.

In the context of this research, our model first assumes that a user will generate a sequence of commands under normal conditions, each with a fixed probability which is independent of the command preceding it and is thus called "naïve" classifier. The probability of a command from a particular user is based on the frequency with which it appeared in the training data, and is given by:

$$P_{c,user} = \frac{\text{Training Count}_{c,user} + \alpha}{\text{Training Length} + (\alpha \times A)} \quad (1)$$

where α is a real number larger than zero to ensure that there are no zero count in the classifier. A is the number of distinct commands in the training data for a specific user. For the illustration, based on the naïve Bayes classifier, the probability (P) that a sequence of six commands sequence "ababcc" (in which each letter represents a unique Unix/Linux command), was generated by a particular user (e.g., user1) is:

$$P_{ababcc,user1} = P_{a,user1} \times P_{b,user1} \times P_{a,user11} \times P_{b,user1} \times P_{c,user1} \times P_{c,user1} \quad (2)$$

Or

$$P_{ababcc,user1} = {P_{a,user1}}^2 \times {P_{b,user1}}^2 \times {P_{c,user1}}^2 \quad (3)$$

In addition, the probability (P') of a system call sequence that is NOT generated by a program can be calculated in the same way in (4).

$$P'_{ababcc,user1} = (1 - P_{a,user1})^2 \times (1 - P_{b,user1})^2 \times \left(1 - P_{c,user1}\right)^2 \quad (4)$$

The ratio R of the probability P' that the sequence of NOT originating from user1 to the probability P of NOT originating can be compared with some defined threshold values for the classification.

$$R = \frac{P'}{P} \quad (5)$$

The R value is used to measure the degree of deviation for each command. To detect a masquerader in a real system, we can assume that the alert should be reported after the execution of a series of commands (e.g. 20 commands). The sum of the R values from each command in a sequence can be compared to a predefined threshold values to classify the sequence as normal from a genuine user or abnormal from a masquerader.

We have found that this approach usually will introduce high false alarms due to the reason that the assumptions of threshold values may be incorrect. Even if the threshold limits are not predetermined, it is difficult to tune the threshold limits to fit all different distributions for users. In Section 4, we use a fuzzy system to measure the overall threat to avoid the high false alarms.

4. Behavior Deviation Evaluation Using Fuzzy Logic

High false alarm rates are the most significant drawback of anomaly detection systems. In the statistical approaches that are used in most of the early versions of intrusion detection systems, a predetermined threshold limit is set for each user in the system being monitored.

In this paper, we have chosen a fuzzy system for the overall threat evaluation of user behavior deviation instead of selecting a predefined threshold. Fuzzy systems will be appropriate considering the facts of uncertainties in security area from unknown characteristics of attacks and system vulnerabilities.

Current intrusion detection systems are still far from mature, as evidenced by voluminous reported incidents and high false alarms. This is partly due to uncertain situations in the security area, like unknown characteristics of attacks, system vulnerabilities.

For anomaly intrusion detection, it may not be the best choice to evaluate the threat precisely, though such approaches usually work acceptably well for misuse intrusion detection systems. In some cases, system or user behavior patterns are difficult to be classified as normal or anomalous, partly due to concept drift problem, where the observed behavior changes slowly over time.

The fuzzy logic based threat evaluation approach presented in this chapter detects anomalous program

or user behavior, while trying not to introduce high false alarms. The fuzzy inference system is applied first to get the membership functions associated with different fuzzy sets. The similarity information of a test case with the behavior profile appears in the premises of fuzzy rules. Then the defuzzification process is done to calculate crisp values of the degree of deviation. The numerical values of the multiple commands deviation are combined into the premises of another fuzzy inference system to obtain the final degree of deviation of the test case.

We have chosen three fuzzy sets "low"', "medium" and "high" to measure the degree of deviation. Three triangular membership functions are used to calculate the membership values based on the ration R value in equation (1).

As it is described, a testing case will be compared with the profile to obtain the value of deviation with the behavior profile. Three fuzzy sets "low"', "medium" and "high" are selected to measure the magnitude. The common triangular membership function is chosen in the immunology model (Fig. 2).

There is no overlap between fuzzy sets "low" and "high" and the point "b" splits the section "ac". The point at which two fuzzy sets intersect has the membership value of 0.5 with both fuzzy sets. A case may be assigned to one or two categories from the fuzzy membership function definition.

Another three fuzzy sets "Normal", "Potential", and "Anomalous" are selected to measure the degree of behavior deviation and the corresponding possibility of potential intrusion to the system in the immunology model. The same triangular membership function with different splitting points is selected. For a test case, if there is no obvious deviation compared with the behavior profile, it belongs to the fuzzy set "Normal". "Potential" means that the user or program actions are suspicious in some degree. "Anomalous" means that the behavior deviation is large enough to report alarms.

The following fuzzy rules are used to infer the degree of deviation of a user command.

Rule 1. If the ratio R is low, then the deviation of the command is low.

Rule 2. If the ratio R is medium, then the deviation of the command is medium.

Rule 3. If the ratio R is high, then the deviation of the command is high.

After the membership values of facts with respect to each antecedent in a rule are determined, the standard MAX-MIN method is applied to measure the impact of fuzzy rules and the highest membership is elected. In our fuzzy system, two rules can be fired if a case falls into two categories with different membership values.

The overall deviation needs to be converted to a numerical value (Δ threat), which is calculated using the center of area (COA) defuzzification method. The COA method is used to obtain a single value of the control output including every element of the number of occurrence for events and its corresponding degree of membership function. The $\Delta threat$ values are stored for each command in a sequence.

The center of area (COA) defuzzification method is applied to get a single non-fuzzy crisp output of final threat ($\Delta Threat$) to measure the degree of the deviation with the behavior profile. The COA formula is:

$$\Delta Threat = \frac{\sum_{k=1}^{n} \mu_k \times center(k)}{\sum_{k=1}^{n} \mu_k} \tag{6}$$

where n is the number of fired rules, μk is the degree of membership of rule k, and center(k) is the peak-value where the fuzzy set for the rule k has the maximum membership values. This final ΔThreat value can be compared with a predefined threshold value to determine the degree of the threat to the system for a certain test case.

The fuzzy system only checks each individual command deviation from the user profile. To get a more accurate estimation of potential threat, we should consider after a series of commands (e.g., 20 commands) have been executed. Each command execution will cause the fuzzy system to fire one or two rules defined. The total threat evaluation level (T) that corresponds to multiple commands in a sequence, It is calculated by multiplying the command's weight values by their $\Delta threat$ values.

In general, if a particular command occurs frequently in the training phase, it will have a high impact and accordingly a high weight in the calculation. In this paper, for the identification purpose of either as a genuine user or as a masquerader, we have used a threshold value 60% to classify the data. Other approaches of linguistic terms can be used to indicate the degree of potential threat level.

5. Experimental Results

The model is applied to detect masqueraders using the data generated in a network laboratory. User commands of ten students are collected with the permission when they worked on Linux systems over a period of several months. The data set is 5000 commands for each of the ten users. The original commands sequence was preprocessed and only the names of these commands were kept. The data from two users are selected to serve as intrusion targets.

For eight of the ten users, the first 3000 commands sequence is kept as training data to build the user profiles. The remaining 2,000 commands sequence is divided into 100 blocks of 20 commands in each for the classification test. The masquerader data is generated from the other two users using cross-validation technique. The 100 blocks from the two masqueraders are chosen to randomly substitute test cases of the eight users. Test blocks will be contaminated either completely or not at all, so that no mixed blocks exist.

To estimate the overall performance, the detection and the false alarm rate are considered simultaneously. The detection rate is defined as the percentage of masquerader data that has been correctly identified as anomalous. The false alarm rate is the percentage of normal data that has been labeled as anomalous and not classified correctly.

For each user, the test data as generated from masqueraders is 200 blocks. The model achieves a high masquerader detection rate in a range between 75% and 88% for each user and the overall detection rate is 81.5%. We also tested the 100 normal cases for each user to ensure that the model does not incorrectly do the classification. If a normal case is classified incorrectly as abnormal, a false alarm is generated. On the average, the false alarm rate is 5.7%.

In conclusion, it is always the ultimate goal for IDSs to detect masqueraders within a short time interval and alert the system earlier to prevent further loss. Based on the results, the interval of about 20 commands execution achieves a high detection rate with an acceptable false alarm rate.

When a larger block size is selected, the model may introduce a higher false alarm rate. In practice, an appropriate size can be selected based upon specific security policies for an organization. In general, it is fairly reasonable and effective for an anomaly IDS to detect potential masqueraders after about 20 or 30 commands execution.

6. Conclusion

Anomaly intrusion detection systems have the capability to detect new or unknown attacks. A user's activities in a UNIX/Linux system can be recorded by the commands sequence. System behavior can be captured from system call sequences generated by privileged processes, which can be manipulated to exploit the vulnerabilities of a computer system.

In this paper, we introduce the Naïve Bayes classification and a fuzzy system to detect masquerades. To do the classification from user activities of commands sequences, a data mining technique Naïve Bayes

classification is used to measure the deviation from the user profiles. New command blocks are compared to calculate the deviation as memberships with fuzzy sets.

In the future, we want to extend the current research of masquerader detection to Windows system where the user activities can be captured by the services or applications initiated by the user clicking using the mouse instead of commands executed. The behavior models from this kind of activity will differ from those discussed in this paper. Future work would address these questions.

References

[1] Bace, R. (2000). *Intrusion Detection*. 2nd Edition, Macmillan Technical Publishing, Indiana.

[2] Donn, B. P. (1994). Demonstrating the elements of information security with threats. *Proceedings of the 17th National Computer Security Conference* (pp. 421-430).

[3] James, P. A. (1980). Computer security threat monitoring and surveillance. *Technical Report*, James, P. Anderson Co., Fort Washington, PA.

[4] Dorothy, E. D. (1987). An intrusion detection model. *IEEE Transactions on Software Engineering*, *13(2)*, 222–232.

[5] Eugene, H. S., & Diego, Z. (2000). Intrusion detection using autonomous agents. *Computer Networks*, *34(4)*, 547–570.

[6] *Computer Security Institute and Federal Bureau of Investigation*. (2013). CSI/FBI Computer Crime and Security Survey, Computer Security Institute Publication.

[7] Terran, L. (2000). Machine learning techniques for the computer security domain of anomaly Detection. PhD thesis, Purdue University, W. Lafayette, IN.

[8] Davison, B., & Hirsh, H. (1998). Predicting sequences of user actions. *Predicting the Future: AI Approaches to Time-Series Problems*, AAAI Workshop, Madison, Wisconsin, 5–12.

[9] Balajinath, & B., Raghavan. V. S. (2001). Intrusion detection through learning behavior model. *Computer Communication*, *24*, 1202-1212.

[10] Ryan, J., & Lin, M., & Miikkulainen, R. (1998). Intrusion detection with neural networks. *Advances in Neural Information Processing Systems.* Cambridge, MA, MIT Press

[11] Lane, T., & Brodley, C. E. (1990), Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, *2(3)*, 295-331.

[12] Schonlau, M., DuMouchel, W., Ju, W., Karr, A., Theus, M., & Vardi, Y. (2001). Computer intrusion: detecting masquerades. *Statistical Science, 16(1)*, 58-74.

[13] J. Han, M.K amber, J. Pei (2012). Data Mining Concepts and Techniques (3ed). Morgan Kaufmann Publisher.

Yingbing Yu received his PhD degree in computer science and engineering from University of Louisville, USA in 2005. His research interests include computer security, network security, soft computing, intelligent system, wireless networking and security.