Detecting Masqueraders Using One-class Classification Methods

Alejandro Pasos[†], V. Rao Vemuri[*†] and Yihua Liao[*]

[*]Department of Computer Science
[†]Department of Applied Science
University of California, Davis
One Shields Ave, Davis
CA 95616, USA

Email: {apasos, rvemuri, yhliao}@ucdavis.edu

Abstract

Detecting masqueraders is one of the most difficult problems in intrusion detection. The masqueraders use a password from a trusted user which makes the attack difficult to detect by traditional security mechanism. This work presents a new alternative to handle this problem: one-class classification methods. These methods are based on having a good knowledge of the normal class to discriminate everything different. Three methods were presented: k-nearest neighbor (k=1), evolving clustering method and autoassociators. All methods were applied to a known Unix dataset that contains the profiles of 50 users. The three methods run with different thresholds and the best results were chosen. In the three cases the one-class classification methods improved the previous results on this dataset.

Detecting Masqueraders Using One-class Classification Methods

I. Introduction

Detecting masqueraders in one of the most difficult problems in intrusion detection. A masquerader is someone who enters the system using the password of a trusted user. He usually guesses or steals that password.

One approach to handle this problem is the normal behavior presented in [1]. This approach, uses examples of normal behavior to build user profile. When some activity is highly different from the profile it is considered as abnormal. Therefore, the main goal to this approach is to define the normal behavior. One of the best methods to define only one class are the one-class classification methods. Usually, these methods utilize thresholds to decide if some activity belongs to a class or not.

We applied the one-class classification methods to a Unix dataset presented by M. Schonlau in [2]. The one-class classification methods improved the performance from 60-67% of detection to 70-80% with an slight increase of the false positive rate from 1% to 2-4%. However, there are two problems that remain in applying one-class classification methods : The first is the threshold selection. Different thresholds were applied and the best was selected. In a real world application, the threshold selection should be automatic. The second problem is the dimensionality of the data. All methods presented here utilize vectors of fixed dimensionality. If in the future the dimensionality increases or decreases that would be a problem.

The rest of the paper is organized as follows: In section 2 we will show some related work, in section 3 the one-class classification approach will be presented, in section 4 the dataset applied here will be explained, in section 5 the results will be discussed and finally section 6 will present the conclusions of this work.

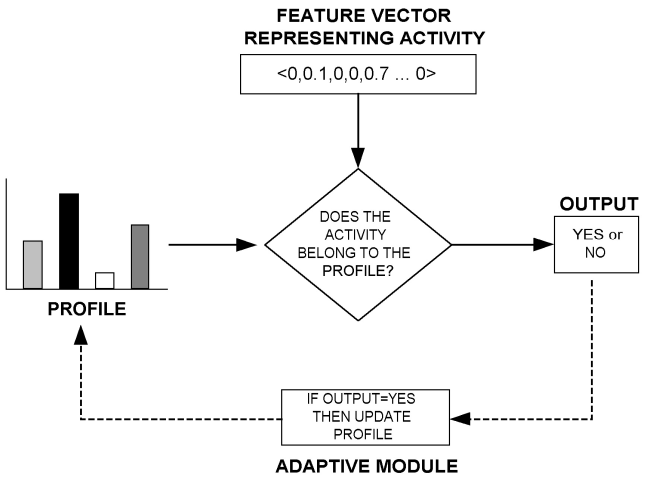

FEATURE VECTOR REPRESENTING ACTIVITY
<0,0.1,0,0,0.7 ... 0>
DOES THE ACTIVITY BELONG TO THE PROFILE?
OUTPUT
YES or NO
PROFILE
IF OUTPUT=YES THEN UPDATE PROFILE
ADAPTIVE MODULE

Fig. 1. General overview of the approach presented.

II. Related Work

The first work on user profile for intrusion detection was introduced in [3]. They presented different levels of granularity to monitor users: keyboard level, command level, session level and group level. They concluded that command level was the best for user profile because they are closer to user's activity. This work will use that assumption and will work in the command level. They applied neural networks to predict the next sequence of user commands. If the neural network fails in predicting the next command, then that sequence was considered abnormal.

The work in [4] utilized an algorithm that measures the similarity of two sequences of user commands. In addition, they updated the stored sequences based on the Last Recent Strategy. Their results showed that sequences from the same user have higher similarity from the sequences than other users. In [5] Hidden Markov Models were applied in the same data.

[2] introduced statistical approaches to this problem. The methods applied were: Uniqueness(if there is a command which was not present in the database, then the sequence was abnormal), one-step markov(the conditional probability of the next command given the previous command), multi-step markov(the probability depends on more than one previous command), sequence match from [4]. In [6] naive bayes classifier was applied to the same dataset and improved the performance.

III. One-Class Classification Approach

As mentioned in the previous section, this work will focus on detecting masqueraders(intruders that impersonate normal users). In [7] four classes of threats in computer systems were defined : Unauthorized access to information (disclosure), acceptance of false data (deception), interruption or prevention of correct operation (disruption) and unauthorized control of some part of the system (usurpation). In this context, masquerading was classified as a form of deception and usurpation.

There are some classical security approaches to avoid masqueraders, for example, password, biometrics etc. However, when those approaches fail, the systems cannot defend against the intrusion. The approach presented here could be applied as a second barrier of security when the first failed.

The first step to a masquerader's attack is to get a password from some user. When the masquerader logs in, the system cannot identify that he is not the user. However, the masquerader's behavior is different from the regular user. Any proposed method in this area must be able to detect this variation between the normal behavior of the user and the masquerader.

The approach used here will the behavior model presented in [1]. Figure III gives a general overview of the approach. There is a predefined profile which is created only with examples of one class. The method inputs new activity that represents the current behavior of the user. Comparing the new activity with the profile, it decides if the activity and the profile belongs to the same user or not. In addition, when the activity belongs to the profile, this profile is updated to store this new behavior (adaptive module).

There are two possibilities to build a profile: Store all possible abnormal behaviors or profile the normal behavior. Because of it is easier to get normal behavior than abnormal behavior, the profile will be built with normal behavior. In addition, the profile will be built using one-class classification [8] methods. One-class classification methods sometimes are called thresholded methods because they need that parameter to assign the activity to the normal class or not. Everything that is not assigned to the normal class is considered abnormal. In addition, there are two more requirements that are not related to one-class classification methods but they are needed in the masquerader problem:

- Online/Adaptive Learning.- The system will be running always which means that there is no separation of training and testing. When it collects enough information, it compares the information with the stored profile to determine if the user is acting as expected or not. Also, the system must be able to detect changes in the user behavior. One main assumption is that the user changes this behavior briefly. If an abrupt change of behavior is found, it will be declared as abnormal.
- Computationally inexpensive and transparent to the user.- The system will be constantly making decisions about user activity and updating the profile if it is necessary. However, all this activity must be transparent to the user. Therefore, it must be computationally inexpensive to avoid any interference with the user's work.

	AUTOAS.	ECM	k-NN
Training	Expensive	-	Inexpensive
Testing	Inexpensive	Inexpensive	Expensive
Memory	Medium	Inexpensive	Expensive

TABLE I
OVERALL CHARACTERISTICS OF THE THREE METHODS. NOTE THAT ECM DOES NOT HAVE A TRAINING PHASE.

In the rest of this section the three methods utilized in this work will be presented. The main characteristics of the methods are summarized on table I. In addition, two important aspects of these one-class classification methods will be discussed: the distance metric and the decision of the threshold.

A. K-Nearest Neighbor

k-Nearest neighbor is one example of instance based learning classifiers [9]. It is one of the simplest but effective algorithms used. It assigns an instance to the class of its closest neighbor based on a distance measure. Given that we are dealing with one class classifiers(k=1), it is necessary to assign an instance to the class based on a threshold.

The k-NN algorithm is considered static learning because the stored examples are not updated over time. In addition, the process of calculating the distance with all stored examples is time consuming.

B. Evolving Clustering Method

ECM is a one-pass, clustering method introduced in [10]. In ECM, there is no separation between testing and training. It clusters the data sequentially and the algorithm is simple enough that allows to execute it online. These characteristics make the method attractive to this problem.

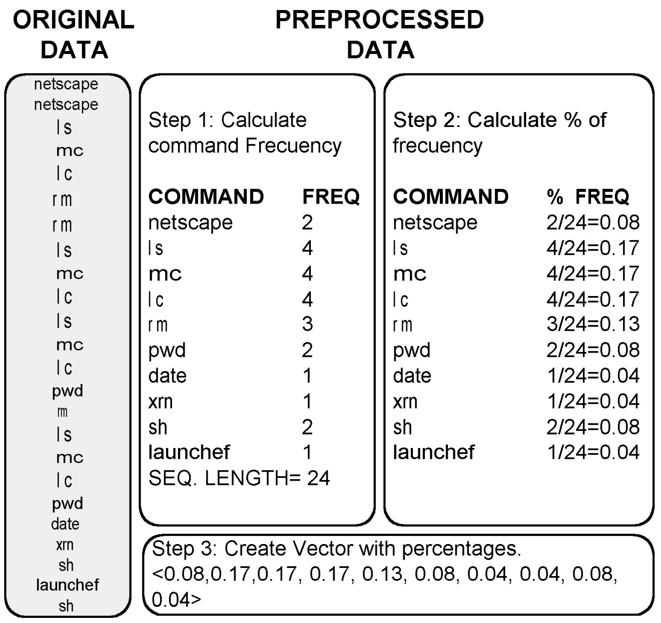
The basic algorithm of ECM is:

1) Calculate the distance between a new vector and all created cluster centers. Choose the minimum of all those distances.
2) If there are no clusters, then create a new cluster with center equal to the new vector.
3) If the distance is less than a threshold assign the vector to that cluster and if is necessary update the center and radius of the cluster

4) If the distance is great than a threshold then create a new cluster with the center equal to the new vector.

The ECM only creates clusters. In order to utilize this algorithm on intrusion detection it is important to add a new step. After N vectors are clustered, all created clusters are analyzed. If any vector contains small number of elements, then that cluster and all its elements are considered as anomalous. This process adds a drawback to this method: it delays the process of detection because we have to wait to cluster N vectors before making a decision. The decision of the parameter N is important because a small N means faster detection but it could leads to increase the false positive rate. On the other hand, a big N would delay the process allow the intruder to finish his attack. In this work this process will be executed after all input vectors are clustered

Other important aspect is how small should be the cluster to be considered as an intrusive. All clusters with less than the 15% of the total number of vectors clustered are considered abnormal.



Step 1: Calculate command Frecuency

COMMAND	FREQ
netscape	2
ls	4
mc	4
lc	4
rm	3
pwd	2
date	1
xrn	1
sh	2
launchef	1

SEQ. LENGTH= 24

Step 2: Calculate % of frecuency

COMMAND	% FREQ
netscape	2/24=0.08
ls	4/24=0.17
mc	4/24=0.17
lc	4/24=0.17
rm	3/24=0.13
pwd	2/24=0.08
date	1/24=0.04
xrn	1/24=0.04
sh	2/24=0.08
launchef	1/24=0.04

Step 3: Create Vector with percentages.
<0.08,0.17,0.17, 0.17, 0.13, 0.08, 0.04, 0.04, 0.08, 0.04>

Fig. 2. Example of how the original files were preprocessed.

C. Autoassociators

Autoassociators are Neural Networks with the number of inputs equal to the number of outputs [8]. The goal of the network is to output the same vector as the input. Based how similar are the input and the output it is decided if the vector belongs to the class or not. The autoassociators tries to store in the hidden layers the principal patterns of the learned vectors. For classification problems, autoassociators utilize one network for each class. Each network learns only from vectors of the same class. To test a new vector, this vector is propagated through all networks and is classified to the class whose network reproduce the most similar output. Usually, the autoassociators uses the backpropagation algorithm with online gradient descendent. The number of hidden units is usually smaller than the inputs.

D. Threshold selection

All classifiers introduced have in common the need of a threshold. We assume that the threshold is specific for each user. All approaches compared the results using different thresholds and selecting the threshold that optimizes the results for each user. However, an optimal approach should have an automatic way to select this threshold.

E. Distance Metric

Another important aspect to these methods is how to calculate the distances between vectors. On other words, What is the best distance metric for this problem?. We considered the following metrics:

1) Euclidean:

$$Dist_e(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2} \tag{1}$$

2) Cosine:

$$Dist_c(\mathbf{x},\mathbf{y}) = \frac{\sum_{k=1}^{n} x_k \cdot y_k}{\sqrt{\sum_{k=1}^{n} x_k^2} \cdot \sqrt{\sum_{k=1}^{n} y_k^2}} \tag{2}$$

3) Hamming:

$$Dist_h(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{n} \mid x_k - y_k \mid \tag{3}$$

IV. Dataset

Schonlau's dataset was presented in [2]. The original files consist of 50 unique users, each user information is stored into a text file consisting of a sequence of 1500 commands in a *tcsh* shell. Each sequence of 100 commands was considered a session. Therefore, each user has 150 sessions. Given that it is difficult to obtain intrusive sequences, sequences from other users were used as intrusions. The first 50 sequences contain commands only from the same user and the remaining 100 commands were contaminated based on the following rule: If the current sequence is normal, the probability that the next sequence is an intrusion is 1%, otherwise the probability is 80%.

	AUTO1	AUTO2	AUTO3	AUTO4
Distance	Cosine	Cosine	Hamming	Hamming
Error	0.009	0.009	0.009	0.09
Hid.Layer	75%	50%	50%	75%

TABLE II
CHARACTERISTICS OF THE FOUR AUTOASSOCIATORS. THE HIDDEN LAYER IS SHOWN AS A PERCENTAGE OF THE INPUT LAYER

Using that strategy leads to have different kind of data, there are users with no intrusions at all, users with few intrusions and users with many sequences of intrusive behavior.

In order to apply the data to the one-class classification methods the original files were preprocessed. Instead of the original sequence of commands, this work used feature vectors. For a sequence of commands in the original files the frequency of each command was calculated. Then, we created vectors based on those command frequencies. Figure IV shows an example of how the data was preprocessed: On the left side there is a sequence of 24 commands. The middle square shows the first step that is add the frequency of all commands. The right square (step 2) calculate the percentage of frequency dividing the frequency of each command with the sequence length . Finally, the square at the bottom (step 3) shows the feature vector which is only the listed percentage of frequencies of step 2. In order to compare our results with the previous results, we decided to use the sequence length of 100.

V. RESULTS

The best results applied over the Schonlau's dataset were One-Step Markov (1S-MV) from [2] and Naive Bayes from [6]. Both results are presented on table III.

We run the k-NN using the three metrics introduced on section III-E. KNN-C, KNN-E, and KNN-H on table III are the results applying K Nearest Neighbor method with cosine, euclidian and hamming metrics respectively. For ECM the Euclidean distance was used. Finally, for the autoassociator we combine the distance metrics, the convergence error and the number of hidden layers. Table II summarizes the four networks topology. The hidden layers are shown as a percentage of the input layer. For example if there are 100 input layers, the first autoassociator of table II will have 100x0.75 = 75 hidden layers.

There are only two possible outcomes on our classification: normal or abnormal. Therefore, we have four possible situations:

- Normal classified as Normal (True negative).
- Normal classified as Abnormal (False positive).
- Abnormal classified as Abnormal (True positive).
- Abnormal classified as Normal (False negative).

	Previous Work		Current Work		
	1S-MV.	N.BAYES	KNN-C	KNN-E	KNN-H
HITS	69.3	61.5	80.5	73.1	80.0
F. P.	6.7	1.3	2.5	4.7	2.9

	Current Work				
	ECM	AUTO1	AUTO2	AUTO3	AUTO4
HITS	77.0	71.4	77.4	70.9	72.7
F. P.	4.4	3.2	4.2	3.6	4.4

TABLE III
RESULTS OF ALL METHODS. THE FIRST TWO METHODS CORRESPOND TO THE PREVIOUS WORKS.

To evaluate the results, the two most important aspects are the hit rate (true positives) and the false positive rate. They can be calculated as follows:

$$FalsePositive = \frac{FalsePositive}{TrueNegative+FalsePositive}$$

$$HitRate = \frac{TruePositive}{TruePositive+FalseNegative}$$

The hit rate is important because it defines how effective is the method detecting intruders. The false positive rate measures how well the system could discriminate the normal from abnormal. The goal is to maximize the hit rate and minimize the false positives. The best result possible is a 100% of hit rate and 0% of false positive rate.

The results are summarized on table III. The k-NN method produced the best results overall. However, it is important to note that k-NN is a computationally expensive method because it compares each new vector with all vectors in training data. In addition, it requires storing all those vectors in memory, which consume many system resources. These are the reasons that difficult to implement k-NN online. Therefore, k-NN method should be considered for learning in batch (off-line) mode.

The best results of autoassociator were using cosine metric. The autoassociator only require memory to store all the weights of the neural network. In addition, the process of detection is faster because it only needs to propagate the new vector and compare the similarity between the input and the output. The main problem of this method is that the training phase is time consuming because the neural network must converge to a minimum error.

Finally, ECM result was good considering that it is a computationally inexpensive method in execution and in memory. The main problem with ECM is that it must delay the detection in order to be sure that suspicious activity is abnormal.

VI. Conclusions

The three methods presented here improved all previous results. In addition, the characteristics of these methods make them more attractive because they consume less system resources and can be executed adaptively (except k-NN). The two most important problems to face are the threshold selection and the dimensionality of the vectors.

All one-class classification methods chose the threshold that improved the results. However, in a real application the threshold selection must be done automatically by the system. Future work on these methods must implement some way to have an automatic threshold selection that optimizes the results.

The dimensionality is another important aspect. Each element of the vectors represents each program executed by the user. However, if in the future the user executes a new program (adding a new feature) this will become a problem. There are two alternatives to handle this problem: Modify the methods to allow variable vector dimensionality or classify programs into a fixed set of predefined features. The first solution can affect the performance of the system. In addition, the new programs executed by the user could grow infinitely leading the methods to the curse of dimensionality problem. The second solution is better because a good classification of these programs could help to a better classification of users. However, it requires user intervention and knowledge to pre-classify each program to a feature and to store a database of each program belonging to each class.

References

[1] E. Amoroso, *Intrusion Detection*, 1st ed. Intrusion.Net Books, 1999.

[2] M. Schonlau, W. DuMouchel, W. Ju, A. Karr, M. Theus, and Y. Vardi, “Computer intrusion: Detecting masquerades,” *Statistical Science.*, vol. 16, no. 1, pp. 1–17, 2001. [Online]. Available: citeseer.nj.nec.com/schonlau01computer.html

[3] M. B. H. Debar and D. Siboni, “A neural network component for an intrusion detection system,” in *IEEE Symp. on Research in Computer Security and Privacy*, 1992, pp. 240–250.

[4] T. Lane and C. E. Brodley, “Detecting the abnormal: Machine learning in computer security,” Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, Tech. Rep., Jan. 1997. [Online]. Available: citeseer.nj.nec.com/lane97detecting.html

[5] T. Lane, “Hidden markov models for human/computer interface modeling,” 1999. [Online]. Available: citeseer.nj.nec.com/lane99hidden.html

[6] R. A. Maxion and T. N. Townsend, “Masquerade detection using truncated command lines,” in *International Conference on Dependable Systems and Networks*, Bethesda, Maryland, 2002.

[7] M. Bishop, *Computer Security: Art and Science*, 1st ed. Addison Wesley, 2002.

[8] D. Tax, “One-class classification,” Ph.D. dissertation, Delft University of Technology, 2001.

[9] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[10] Q. Song and N. Kasabov, “Ecm - a novel on-line, evolving clustering method and its applications.” [Online]. Available: citeseer.nj.nec.com/526700.html