Masquerade Detection Based on One Class SVM

Yuxin DING[1] Ping SUN[1] Xiuyue CHEN[1] Changan LIU[1]

[1]Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China

Abstract

Masqueraders invade into users'system and impersonate the real users to do whatever they want. Unfortunately, firewalls or misuse-based intrusion detection systems are generally ineffective in detecting masquerades. In this paper an abnormal detection method based on one class SVM are presented to detect masquerade activities using UNIX command sets. Firstly the performance of binary SVM classifier are studied to illustrated why one class SVM are adopted, then to improve the performance of one class SVM different feature selection methods are studied, experimental results show that for abnormal detection using UNIX command simplifying raw data and decreasing the dimensions of feature space is an effective approach to improve the performance of SVM classifiers for masquerade detection.

Keywords: anomaly detection, SVM, Shell command

1. Introduction

A masquerader is someone who pretends to be another user while invading target user's accounts. For example, a masquerader invades an UNIX user's account and execute shell commands on victim's machine. Unfortunately firewalls and misuse-based intrusion detection system haven't the ability to detect such intrusion. In general masquerade detection is a form of anomaly detection because what we know is the normal behavior of each user, the behavior of a masquerader is unknown. We assume a masquerader's behavior is different from that of the real user.

Schonlau and DuMouchel[1] made an experiment on masquerade detection using the UNIX commands. Various masquerade detection methods, such as hybrid markov, sequence match, were evaluated, and the accuracy of the most effective masquerade detection technique did not exceed 70%. Maxion and Townsend[2] conducted another experiment using the same data, they used an updated Naive Bayes classifier, the accuracy of masquerader detection was improved from 39.4% to 61.5% while maintaining comparable level of false alarms rate. Maxion[3] conducted another experiment using the data set referred to as the Greenberg data. Different from Schonlau data, Greenberg data contain ''enriched commands'' including name, arguments, flag, alias, options, directory, and history. He compared the result from truncated data set using Naive Bayes classifier with that of enriched command data set using same classifier. As expected, accuracy of masquerader detection improved when enriched commands were used. Kim[4] adopted SVM, it is proved that that SVM is more effective than Naive Bayes technique by repeating the experiments conducted by Maxion and Townsend[2] and Maxion[3] using the same data sets in the similar configurations.

In this paper we also adopt SVM to detect masqueraders as the experiments of Schonlau and Maxion. Experiments data we adopt is UNIX shell commands, we build normal model for each user. For

one user, UNIX commands of other users can be seen as masquerade activities. We mainly focus on the following problems : Which one, two class SVM or one class SVM, is more appropriate for masquerade detection?
How do we extract data features to improve SVM's discrimination ability?

2. Support Vector Machine (SVM)

In this section, we give a very brief review of SVM and refer the details to [5] [6]. Consider *N* training samples$\{x_1,y_1\},...,\{x_n,y_n\}$ where x_i is a m-dimensional feature vector representing the *i*th training sample, and y_i is 1or -1, it is the class label of xi. A hyperplane in the feature space can be described as the equation $w^T x + b = 0$, where $w \in R_m$ and b is a scalar. When the training samples are linearly separable, SVM yields the optimal hyperplane that separates two classes with no training error, and maximizes the minimum distance from the training samples to the hyperplane.
It is easy to find that the parameter pair (*w; b*) corresponding to the optimal hyperplane is the solution to the following optimization problem:
minimize : $L(w) = 1/2\|w\|^2$
subject to : $y_i(w^T x_i + b) \geqslant 1 \quad i = 1..N$
For linearly nonseparable cases, there is no such a hyperplane that is able to classify every training sample correctly. From above we can see that SVM is originally designed for binary classification.
For One class SVM, the training data don't require to be labled. Only positive examples are used in training and testing. One class SVM map the data into the feature space and then try to use a hypersphere to describe the data in feature space and put most of the data into the hypersphere. This can be formulated into an optimization problem. We want the ball to be as small as possible while at the same time, including most of the training data. We only consider the positive points and get the objective function in the following form (primal form):
min $R^2+(1/vl)\sum_i.\zeta_i$
s t $\|\Phi(X_i)\text{-}c\| \leqslant R^2+\zeta_i$
The trade off between the radius of the hyper-sphere and the number of training samples that it can hold is set by the parameter $v \in [0,1]$. When v is small, we try to put more data into the "ball". When v is larger, we try to squeeze the size of the "ball".

3. Masquerade Detection

3.1. Two class SVM vs. one class SVM

Masquerade detection is a multi-class problem. Each user is a class, classifiers should be able to discriminate whether data are created by a certain user, if the data can't be classified into any user class, it is classified as abnormal data or masquerade data.
SVM is originally designed for binary classification, so we need to extend two class SVM to the multi-class scenario. The conventional way is to decompose the *M*-class problem into a series of two-class problems and construct several binary classifiers. the most widely used implementations are the one-against-one method and the one-against-all method, the one-against-all method constructs *M* SVM classifiers with the *i*th one separating class *i* from all the remaining classes. To clearly show the performance of two class SVM for masquerade detection, in this paper the one-against-one method is adopted, the one-against-one method constructs M(M-1)/2 SVM classifiers, each of them can only discriminate two classes. For convenience, classifier(i,j) means this classifier is built from $user_i$ and $user_j$, and can discriminate $user_i$ from $user_j$. For a testing data, if there exist M classifiers and the M classifiers assign it to the

same user, we decide this data belongs to this user class, or it is a masquerade data.
For one class SVM, classifier construction is easy, we only need to construct one classifier for each user, this classifier can discriminate this user from masqueraders, if a testing data is not accepted by a user's classifier, it is classified as masquerade data. In section 4 the performance of those 2 type of SVM are compared and analyzed.

3.2. Feature Selection

The key point for masquerade detection is how to extract data features from training data. Before presenting feature selection methods, we give a description of the experimental data. We use the data provided by the Purdue University that collect together eight UNIX users' history files. Those files record the history shell commands used by user. The data in each file consist of shell commands sessions. Each session is a command sequence used by a user from his log in until his log out. Firstly the data is preprocessed in the following way:
1.Reserving the name, mark, and meta characters of the shell commands.
2.Replacing the filename, hostname, directory, website and other information with the unified identifiers <n>, where n denotes the number of them.
3.Inserting the identifiers at the beginning and the end of a session.
4.ordering the shell command tokens according to the times that they appear in a session, then connecting each session one by one.
5.Not appending the timestamps to the data.
The preprocessed data looks like below:
SOF， cd， <1>， ls， -laF， |， more， cat， <3>， >， <1>， exit， **EOF**， **SOF**， cd， <1>， xquake， &， fg， vi， <1>， mailx， <1>， exit， **EOF**
The **SOF** and the **EOF** are the beginning and ending symbols respectively,<1> is the identifiers of the filename, directory, and other information. For convenience, each command is represented by a token, the token is the same for the same command.
According the above data format, 3 feature selection approached are designed:
1.Fix-sized Random Feature Extraction: a fix-sized window is used to slide on each session data, when window moving, the command sequence in the window is a feature vector for training or testing. In this method two parameters need to be considered, one is the window size, the other is the sliding step of the window. The property of this approach is that feature extraction is simple, but the meaning for each dimension of a feature vector is random.
2.Fix-sized Frequency Feature Extraction: this approach is very similar with approach one, a fix-sized sliding window is also used, the frequency of each command appearing in the window is calculated. In this approach the length of a feature vector is the number of token types, each dimension of a feature vector represents the frequency of a token (or command) appearing in the window. The property of this method is that the dimensions of the feature vector is high, it is the number of all distinct tokens presented in all training and testing sets, but each dimension has a fixed meaning.
3.Class-based Frequency Feature Extraction: one problem of approach 2 is that the dimensions of feature vectors is high, it is the number of all tokens, in our data sets the number of tokens is 1936. In pattern recognition we usually think adding more features can improve the performance of classifier, but if the new adding features represent the feature of noise, this will easily lead classifier overfits training data and the classifier can't be generalized. So we take some measures to

decrease the dimensions of feature vectors. In this paper according the functions of UNIX shell commands they are divided into 135 categories. By this approach each dimension of a feature vector represents the frequency of a token class (or command class) appearing in the window.

In section 4 experiments which adopt different feature selection approaches are made, and the experimental results are analyzed.

4. Experiments

To evaluate the model's performance, we define two criteria: True Detection Rate (TDR) and Fault Detection Rate (FDR). TDR is the ratio of the number of true masquerade data detected by classifier to the total number of true masquerade data. The FDR is the ratio of the number of a normal user data classified as masquerade data to the total number of a normal user data. For a high accuracy classifier, the TDR should be high and the FDR should be small.

4.1. Experiments of two class SVM

We first test the performance of the two class SVM with the one-against-one method. We select six users whose command session number is bigger than 1000. two third of the data is used as the training data, the other is testing data. window size is 30, the moving step of window is 6. The result is shown in table 4-1.

Here we give a short illustration on table 4-1, for example in the first line the test user is user1, we use user1's data to test classifiers classifier(u1,u2), …. classifier(u1,u8). For one-against-one classifier, only a feature vector of user1 is classified by all classifiers of this line, that is classifier(u1,u2) ,….classifier(u1,u8),as user1, this vector is recognized as user1's data. that is to say classifiers are interdependency, if one classifier is overfit, the whole classification accuracy will be affected greatly. In table 4-1 only the testing data of user5 and user6 can be better recognized by classifiers. Other user's data almost are recognized as masquerade data. That is the reason why we choose one class SVM. In the following experiments we all use one class SVM.

Table 4-1 results of two class SVM

Test user	The TDR of each one-against-one classifier					
	U1	U2	U4	U5	U6	U8
U1		99.8	62.6	23.2	83.4	43.4
U2	52.4		96.5	20.8	0	68.1
U4	100	100		100	0	100
U5	100	100	92.5		96.7	94.1
U6	87.7	100	100	91.2		96.6
U8	100	100	0	100	88.9	

4.2. Experiments for 3 feature extraction approaches

Table 4-2 SVM results for approach 1

test user	TDR for Each One Class SVM (Diagonal is FDR)					
	M1	M2	M4	M5	M6	M8
U1	**51.2**	48.4	50.4	54.3	49.7	51.2
U2	50.45	**49.3**	53.9	63.8	51.8	53.2
U4	56.0	53.9	**44.8**	60.8	43.4	45.6
U5	46.0	45.3	44.8	**38.6**	45.1	48.3
U6	60.4	58.9	61.4	70.4	**39.2**	64.3
U8	36.2	34.7	37.2	43.5	35.8	**74.3**
AV TDR	49.8	48.2	49.5	58.5	45.1	52.5

Table 4-3 SVM results for approach 2

test user	TDR for Each One Class SVM (Diagonal is FDR)					
	M1	M2	M4	M5	M6	M8
U1	**12.9**	62.4	35.6	51.9	13.2	64.9
U2	50.3	**1.1**	26.3	22.3	17.3	43.8
U4	38.9	33.3	**11.2**	83.7	28.7	46.4
U5	49.1	21.9	73.4	**9.4**	28.2	32.8
U6	23.0	55.8	66.0	65.8	**6.1**	23.8
U8	82.3	77.2	54.5	45.6	14.8	**6.9**
AV TDR	48.7	50.1	51.1	53.8	20.4	42.3

Table 4-4 SVM results for approach 3

test user	TDR for Each One Class SVM (Diagonal is FDR)					
	M1	M2	M4	M5	M6	M8
U1	**7.6**	55.1	27.3	40.7	20.9	73.9
U2	48.4	**1.9**	15.7	30.6	34.2	58.6
U4	27.6	16.7	**6.4**	80.5	61.5	39.9
U5	41.2	30.3	73.9	**11.7**	48.4	27.2
U6	30.6	44.4	69.7	56.9	**6.9**	16.7
U8	82.8	71.5	43.9	30.8	16.7	**2.9**
AV TDR	46.1	36.6	46.6	52.1	41.2	43.2

On the above tables Mi represents the classifier of user i. Ui represents the testing data of user i. the number at line Ui and row Mi is the FDR of classifier Mi. For classifier Mi, all the testing data Uj (j!=i) can be seen as the masquerade data, the number at line Mi and Uj(j!=i) is the TDR of classifier Mi corresponding to masquerade data set Uj. AV TDR is the average TDR of each classifier, it is the average of the TDR in each row. In table 4-2 total average FDR is 49.5%, total average TDR is 50.6%. In table 4-3 the total average FDR is 7.9%, total average TDR is 44.4%. In table 4-4 total average FDR is 6.2%, total average TDR is 44.3%.
From above table it can be seen that classifiers using feature extraction approach 1 almost haven't the ability to detect masquerade data. The performance of classifiers using feature extraction approach 2 is better than that of approach 1. Though it is TDR is relatively low, it is FDR is desirable, lower FDR means lower false alarm rates which is important in practice. Classifiers in table 4-4 basically have the same performance as classifiers in table 4-3, but the dimensions of feature vectors of classifiers in table 4-4 is 135, while the dimensions of feature vectors of classifiers in table 4-3 is 1936. This proved that approach 3 is more effective compared with other approaches.

5. Conclusion

In this paper SVM are applied to detect masquerade activities. The performance of two type of SVM, two class SVM and one class SVM, are compared. Due to the interdependency among one-against-one SVM classifiers, we think one class SVM is more appropriate for masquerade detection. To improve the accuracy of one class SVM, 3 feature extraction methods are presented, the experiments results show that Class-based Frequency Feature Extraction method can effectively decrease the dimension of feature space and lower the computation complexity of SVM.

6. References

[1] Schonlau Matthias, etc, "Computer intrusion: detecting masquerades", Statistical Science ,vol.16, no.1, 2001, pp.58-74.

[2] Maxion Roy, Townsend Tahilia, "Masquerade detection using truncated command line", Proc. of international conference on dependable systems and networks, pp. 219-228,2002.

[3] Maxion Roy, "Masquerade detection using enriched command lines", Proc. of international conference on dependable systems and networks , pp. 5-14, 2003.

[4] H.S. Kim, S.D. Cha "Empirical evaluation of SVM-based masquerade detection using UNIX commands" Computers & Security, vol.24, no.2, pp.160-168,2005.

[5] V.N. Vapnik, "An Overview of Statistical Learning Theory", IEEE Trans. on Neural Networks, vol. 10, no. 5, pp. 988-999, Sept. 1999.

[6] C. Cortes and V.N. Vapnik, "Support Vector Networks", Machine Learning, vol. 20, no. 3, pp. 273-297, 1995.