Computers & Security (2005) **24**, 160—168

ELSEVIER

Computers & Security

www.elsevier.com/locate/cose

Empirical evaluation of SVM-based masquerade detection using UNIX commands

Han-Sung Kim*, **Sung-Deok Cha**

Division of Computer Science and AITrc/SPIC/IIRTRC, Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), 373-1, Guseong-Dong, Yuseong-Gu, Daejeon, Republic of Korea

Received 20 May 2004; revised 27 July 2004; accepted 18 August 2004

KEYWORDS
Intrusion detection;
Masquerade detection;
Anomaly detection;
Machine learning;
Support vector machine (SVM)

Abstract Masqueraders who impersonate other users pose serious threat to computer security. Unfortunately, firewalls or misuse-based intrusion detection systems are generally ineffective in detecting masquerades. Although anomaly detection techniques have long been considered as an effective approach to complement misuse detection techniques, they are not widely used in practice due to poor accuracy and relatively high degree of false alarms. In this paper, we performed an empirical study investigating the effectiveness of SVM (support vector machine) in detecting masquerade activities using two different UNIX command sets used in previous studies [R. Maxion, N. Townsend, Proceedings of international conference on dependable systems and networks (DSN-02), p. 219—28, June 2002; R. Maxion, Proceedings of international conference on dependable systems and networks (DSN-03), p. 5—14, June 2003]. Concept of ''common commands'' was introduced as a feature to more effectively reflect diverse command patterns exhibited by various users. Though still imperfect, we detected masquerades 80.1% and 94.8% of the time, while the previous studies reported the accuracy of 69.3% and 62.8%, respectively, using the same data set containing only the command names. When command names and arguments were included in the experiment, SVM-based approach detected masquerades 87.3% of the time while the previous study, using the same data set, reported 82.1% of accuracy. These combined experiments convincingly demonstrate that SVM is an effective approach to masquerade detection.

* Corresponding author. Tel.: 82 42 869 3575; fax: 82 42 869 8488.
E-mail addresses: kimhs@salmosa.kaist.ac.kr (H.-S. Kim), cha@salmosa.kaist.ac.kr (S.-D. Cha).

0167-4048/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2004.08.007

Introduction

A masquerader is someone who pretends to be another user while invading target user's accounts, directories, or files (Anderson, 1980). The term can be extended to include legitimate but suspicious use of privilege by insiders. While the UNIX root user is usually authorized to read any file, it is highly unusual and even suspicious if for root user to sequentially read all the files stored in user directories or scan the entire file system searching for files containing specific keywords. In such cases, it is prudent to investigate if the root password has been compromised rather than choose to ignore such activities just because the user signed on as the root.

Violation of security policies by insiders is arguably the most serious security threat as the case of FBI agent Hansen (Webster, 2002) illustrates. Such attacks are difficult to detect since commonly deployed security products such as firewalls are unlikely to be effective. Signature-based intrusion detection systems may detect such attacks only if intruders execute the attack codes whose signatures are known. Besides, there are known techniques to evade IDS products (Ptacek, 2002; Rain Forest Puppy, 1999). Therefore, such misuse detection systems can provide only limited help. Access control mechanisms are unlikely to become an effective solution to insider attacks because an inside attacker can install sniffing software to bypass access control mechanisms in place.

Anomaly detection has long been known as an essential component in securing computers and networks. Basic approaches are well established, and techniques such as statistical analysis (Schonlau and DuMouchel, 2001), data mining (Lee and Stolfo, 1998), and various machine learning techniques (Lane, 2000) have been used. Effective anomaly detection mechanisms can defend attacks initiated by either insiders or outsiders. Masquerade detection is a form of anomaly detection because behavior of a masquerade is assumed to be different from that of the real user. Unfortunately, anomaly detection techniques are not widely used in practice primarily because error rate in anomaly detection is still unacceptably high.

In this paper, we demonstrate that SVM (support vector machine), a machine learning technique, is more effective than Naive Bayes technique by repeating the experiments conducted by Maxion and Townsend (2002) and Maxion (2003) using the same data sets in the similar configurations and by comparing the results objectively and quantitatively.

The remainder of the paper is organized as follows: next section introduces the related work on masquerade detection and SVM. Then, we describe architecture for the SVM-based masquerade detection framework, which is followed by the experimental designs and results of two experiments. Conclusions and future works are presented in last section.

Related work

Masquerade detection experiments

Schonlau and DuMouchel (2001) conducted an experiment on masquerade detection using only the command names collected by UNIX `acct` auditing mechanism. They collected 15,000 "truncated" commands each from 70 users and randomly chose 50 of them in the experiment. Commands entered by the rest, 20 users, were used to simulate masquerade activities. Each command set was decomposed into 150 blocks consisting of 100 commands each, and the first 50 blocks, or 5,000 commands, were used as training data and the rest as test data. Experiment administrators randomly inserted 0∼24 command blocks as a means of approximating incursions by masqueraders.

As shown in Table 1, various approaches to masquerade detection were evaluated, and the accuracy of the most effective masquerade detection technique did not exceed 70%.

Maxion and Townsend (2002) conducted another experiment using the same data and same configuration referred to SEA configuration used in Schonlau and DuMouchel (2001). They analyzed data in a different configuration, referred to as the 1 vs 49 configuration, where the first 5,000 commands were used to build a model, and the first 5,000 commands entered by each of the rest of group, 49 users, were used as test data. In SEA configuration, each user's rest 10,000 commands containing simulated masquerade blocks are used as test data. However, 1 vs 49 configuration had a richer source of masquerade data in that 2450 (i.e., 50 blocks $\times$ 49 users) command blocks were used as masquerader data.

Using Naive Bayes classifier, Maxion and Townsend improved the accuracy[1] of masquerader detection from 39.4% to 61.5% while maintaining

[1] In the SEA configuration, total 231 masquerade blocks were inserted in the test data, and the accuracy was calculated, in percents, as follows: accuracy = (the number of blocks detected among masquerading blocks)/231.

Table 1 Results of masquerader detection experiments using "Schonlau data"

	Techniques	Accuracy (%)	Misses (%)	False alarms (%)
	Uniqueness	39.4	60.6	1.4
Schonlau and DuMouchel (2001)	Bayes 1-step Markov	69.3	30.7	6.7
(SEA configuration)	Hybrid Markov	49.3	50.7	3.2
	Compression	34.2	65.8	5.0
	IPAM	41.1	58.9	2.7
	Sequence match	36.8	63.2	3.7
Maxion and Townsend (2002)	Naive Bayes (updating)	61.5	38.5	1.3
(SEA configuration)	Naive Bayes (no updating)	66.2	33.8	4.6
(1 vs 49 Configuration)	Naive Bayes (no updating)	62.8	37.2	4.6

comparable level of false alarms[2] (i.e., 1.4% and 1.3%, respectively).

Maxion (2003) conducted another experiment using the data set referred to as the Greenberg data (Greenberg, 1988). As opposed to Schonlau data, which contained truncated commands, Greenberg data contain "*enriched commands*" including name, arguments, flag, alias, options, directory, and history. For example, command shown in Table 2 would be "`ar`" when interpreted as a truncated command and "`ar cd; set; ls -Fg`" when treated as an enriched command. Alias name itself is included in the experiment to distinguish the same command sequence entered using different alias.

The original 168 user data are split into four groups comprising 55 novice users, 36 experienced users, 52 computer-scientist, and 25 non-programmer users. Maxion randomly selected 50 users who have more than 2,000 but less than 5,000 commands, and the first 1,000 commands were used as training data whereas the next 1,000 commands were used as test data. Additional 25 users were randomly selected from the remaining pool of 118 users to serve as a source of masquerade commands. To simulate masquerade activities, the last 100 command lines from each of the 25 masquerader users were concatenated to produce a stream of 2,500 command lines. In this experiment, the detection block was reduced into the block of 10 commands each. Thirty of these units were then selected at random and injected at randomly selected positions, without replacement, into the stream of 1,000 self commands lines, thereby resulting in 130 blocks of 10 command lines for testing.

Maxion extracted both "*truncated command*" data set and "*enriched command*" data set from this configuration. 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 (Table 3).

[2] False positive alarm rate was calculated, in percents, as follows : false rate = (the number of detected blocks as masquerades among real user blocks)/(5000 − 231).

Support vector machine (SVM)

Support vector machine refers to a collection of machine learning algorithms designed for binary classification, developed by Vapnik and enhanced by others since 1995. SVM classifies data by determining a set of support vectors, which are members of the set of training inputs that outline a hyper plane in feature space (Burges, 1998) as shown in Fig. 1.

SVM is based on the idea of structural risk minimization, which minimizes the generalization error, i.e. true error on unseen examples. The number of free parameters used in the SVM depends on the margin that separates the data points but not on the number of input features. Therefore, SVM does not require a reduction in the number of features in order to avoid overfitting. SVM provides a generic mechanism to fit the surface of the hyper plane to the data through the use of a kernel function. The user may provide a function, such as a linear, polynomial, or sigmoid curve, to the SVM during the training process, which selects support vectors along the surface of this function. The selection of kernel and parameters plays

Table 2 Example of "Greenberg raw data" (Greenberg, 1988)

S	Fri Mar 6 14:08:30 1987
E	NIL
C	ar
D	/user/cpsc211/l03b32/xxxxxxx
A	cd ass/rough; set prompt = "[$cwd:t]!⇒"; ls -Fg
H	NIL
X	NIL

Table 3 Results for Naive Bayes on Greenberg data, averaged across all users (Maxion, 2003)

Types of data	Truncated	Enriched
Amount of training data	1000	1000
Block size	10	10
Hits (%)	70.9	82.1
Misses (%)	29.1	17.9
False alarm (%)	4.7	5.7

a critical role on the classification performance by SVM.

SVM offers several advantages as a tool for masquerade detection. First, SVM has shown outstanding performance, compared to other machine learning techniques, in several classification problems (Joachims, 2000; Yang and Liu, 1999; Dumais and Chen, 2000). For text categorization, SVM outperformed neural network and Naive Bayesian model. For network intrusion detection, SVM demonstrated superior performance to neural network (Mukkamala et al., 2002). In Fugate and Gattier (2002), performance of Mahalanobis distance-based approach, one of the outlier detection techniques, was reported to be inferior to that of SVM.

Second, SVM is capable of tolerating problems often associated with high dimensions and sparse instance spaces. SVM supports various kernels (e.g., linear function, polynomial function, and radial based function) to solve non-linearly separable and complex problems. For example, in our experiment, 824 unique commands were used. Neural network is incapable of efficiently processing such large number of inputs, and computational complexity associated with building nodes and calculating weights might be prohibitively expensive.

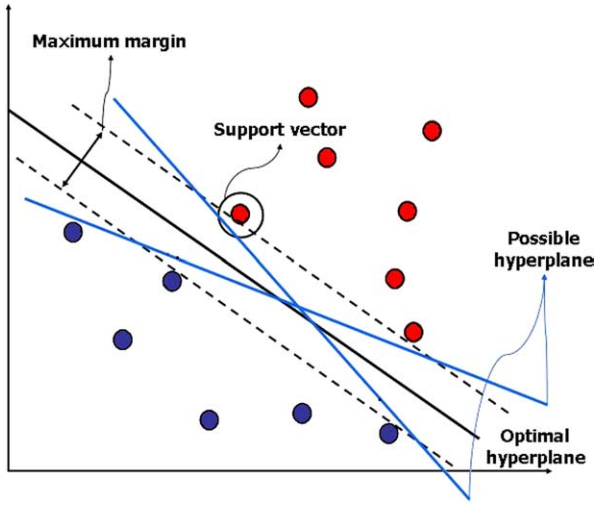


Figure 1 Classification by SVM.

Third, SVM is relatively easy to use compared to other machine learning techniques. For example, with neural network, a user must program details on the number of nodes, weight, and how various nodes are to be connected. With SVM, a user simply needs to select the kernel and parameter values. Although finding the optimal parameter values can be a time consuming task, the values approximating the optimal ones can be determined relatively easily using scaled data and grid-search methods (Hsu et al., 2004). In addition, libSVM (Chang and Lin, LIBSVM) provides a graphical tool to assist users to choose parameter values, and there are several public domain SVM tools (e.g., SVM^{Light}, libSVM, and SVMTorch) which can dynamically update the model during training period if new patterns emerge.

Fourth, SVM is relatively insensitive to the number of data points and can potentially learn a large set of patterns because the complexity of classification problem does not depend on the dimensionality of the feature space (Joachims, 2000).

Lastly, time needed to perform SVM-based modelling and analysis is reasonably short. For example, running SVM^{Light} on a typical Pentium-IV PC required about 70 s for each user to generate a model based on 24,000 data with 824 features. Time interval varied from 0.9 s to 123 s. Once the model is generated, processing of test data to detect masquerade activities took, in most cases, less than a second to process a sequence of 50 commands. Given that commands are usually manually entered from keyboards, such data demonstrate that real-time masquerade detection is possible with SVM.

Proposed approach

Fig. 2 illustrates the overall system architecture for host anomaly detection for UNIX systems (HADUS). It consists of two major components: model learner and anomaly detector. The former develops user profile using audit data. Examples include command history logs, audit data recorded by `acct` mechanism, log data generated by application programs such as `ttywatcher` or Solaris BSM module. While the HADUS architecture is independent of audit data sources,[3] data must first be

[3] Even though our proposed architecture is independent of audit data source, only UNIX command data are used. A fair competition with the previous experiments demands that the same data and configurations are used. Additional information such as process-related logs is likely to improve the classification accuracy.

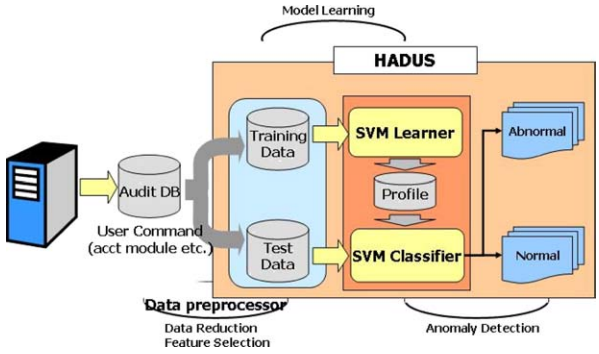


Figure 2 Overall system architecture.

preprocessed so that values corresponding to the selected features can be extracted and stored in database. Once user profile is developed, SVM classifier examines test data and determines if they are to be considered normal or abnormal.

Data preprocessor and feature selection

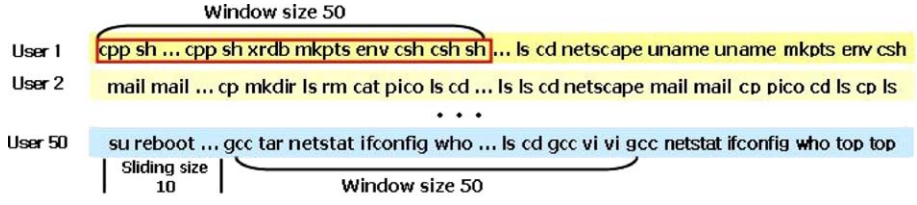
Feature selection is the most critical factor determining the effectiveness of masquerade detection. That is, one must identify a set of characteristics (e.g., types of commands issued) and usage patterns (e.g., frequency and sequence of commands issued) that are likely to be most useful in distinguishing genuine behavior of a user from that of a masquerade.

To build user profile using SVM, "*command frequencies in fixed window size* (*sub-block*)" is used. Decision on the window size is critical. It must be kept as small as possible to allow near-real-time masquerade detection. On the other hand, it must be large enough to make meaningful statistical analysis possible. In our study, we divided the command sequence into a block of 100 commands each as previous studies did, and each block was further divided into sub-blocks of 50 commands. Each command name was given a unique ID, and the frequency of each command appearing in each sub-block was calculated as shown in Fig. 3.

There are commands that are likely to be used quite frequently, such as "`cd`" or "`ls`" command, by most users. On the other hand, some UNIX commands, such as "`reboot`" or "`shutdown`," are most likely to be executed only by system administrators having the root privilege. As one needs to explore ways to better reflect diverse command usage patterns among different users, we introduced the notion of "*common commands*." That is, we decided to treat a set of commands used by more than *X* number of users at the rate exceeding *Y* percent — *X* and *Y* values can be customized to provide the best performance — as common commands. As shown in Fig. 3, some commands including ID 1 and 3 are treated as one command, "*common*."

Classification with voting engine

In the first experiment, blocks of 100 commands were divided into six different sub-blocks, each containing 50 commands with a sliding window of



Examples of Features Window Size 50, Sliding Size 10

user	1	2	3	4	5	6	...	99	100	label
1	5	2	2	1	0	0	...	0	0	1
1	1	1	1	1	1	2	...	0	0	1
:	:	:	:	:	:	:	:	:	:	:
50	0	1	0	1	0	0	...	0	1	-1

Examples of Features Window Size 50, Sliding Size 10 with Common

user	1	2	3	4	5	6	...	99	100	common	label
1	-	2	-	1	0	0	...	0	0	10	1
1	-	1	-	1	1	2	...	0	0	8	1
:	:	:	:	:	:	:	:	:	:	:	:
50	-	1	-	1	0	0	...	0	1	4	-1

Data Base

ID	Name	common
1	cpp	1
2	sh	0
3	xrdb	1
4	mkpts	0
5	env	0
6	csh	0
...		
99	netscape	0
100	uname	0

Figure 3 Feature selection for SVM profiling.

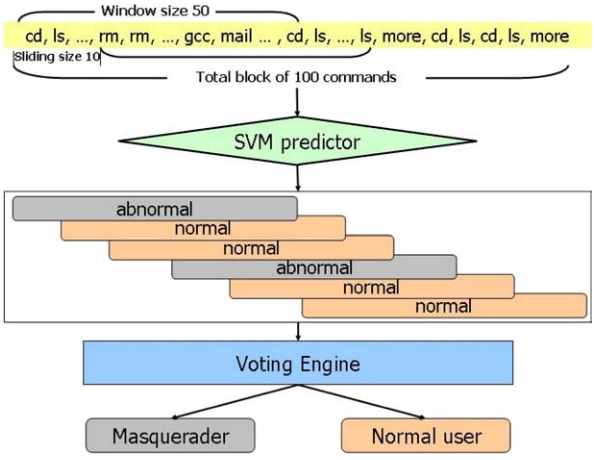


Figure 4 Voting engine in SVM classifier.

size 10. SVM predictor determines if each sub-block is normal or not as shown in Fig. 4. The "*voting engine*" decides if the total block is to be considered as being anomalous. If the number of masquerade sub-blocks exceeds threshold value, the block is considered as masquerade block. The voting engine is designed to reduce false alarm due to data noise.

SVM kernel and parameter selection

Other factors critical to the effectiveness and performance of HADUS are the selection of the SVM kernel and parameter values. As noted earlier, finding the optimal parameter values can be an iterative task. However, there are techniques known to be useful in selecting adequate parameter values without requiring excessive overhead, and tools are available to guide users in the decision process. In order to develop profile for each user, RBF kernel was used and the final parameter values were decided using cross validation technique.

Experiment I: using truncated commands

Experimental design

In this experiment, as noted earlier, we used the same data set and configurations as in previous experiment (Maxion and Townsend, 2002). We used fixed-sized sliding window scheme while using the SEA configuration reported in Schonlau and DuMouchel (2001). That is, we further divided each block into six different sub-blocks, each containing 50 commands structured into a sliding window of size 10. Each block is simply a sequence of commands, and there were no arguments included.

Experimental results

In Fig. 5, the effect of incorporating common commands is shown. For example, label "U35 F30 (26)" indicates that there were 26 commands, out of 824 unique commands appearing in the data set, that are used by more than 35 (out of 50) users at the frequency of exceeding 30%. In terms of SVM modelling, it has the effect of reducing the feature size by 25, thereby making technique more efficient. We repeated experiment using various values of X and Y to investigate the effect of introducing the commonness of commands as additional feature, and we were able to reduce the rate of false positive to as low as 14.04%, as opposed to 21.58% when only the frequency of commands was used in masquerade detection.

Different threshold values were used to evaluate how accurate masquerade detection becomes depending on how sensitive we make the analysis to be. Four different lines in Fig. 5 illustrate how the false positive rate varies if different decision algorithm — voting algorithm — is applied. For example, when the occurrence of up to three, out of six, masquerade sub-blocks are to be treated as noise, as shown with the threshold level 4, the false positive rate was the lowest. If, on the other hand, any occurrence of sub-block which is classified to be that of masquerade activities is sufficient to make the entire 100 command block as commands entered by masquerades, the false positive rate was shown to be the highest.

As further illustrated in Fig. 6, when we increase the minimum number of sub-blocks required to be suspected of containing commands issued by masquerades, the rate of false positive

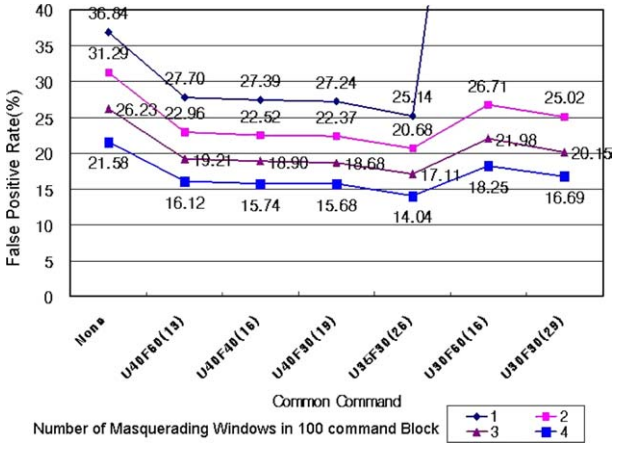


Figure 5 The effect of common command.

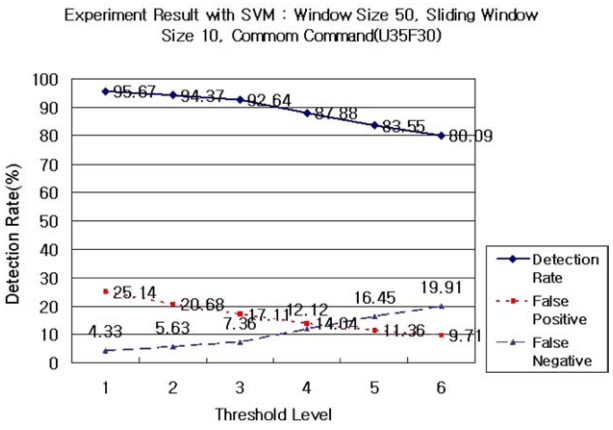


Figure 6 The effect of voting engine in SEA configuration.

could be further reduced from 14.04% to 9.71%. Unfortunately, stricter requirements also resulted in the loss of accuracy in the masquerader detection from 95.67% to 80.09%.

It must be noted that in our experiment, the sum of accurate masquerader detection rate and the rates of false detection, positives and negatives included, does not always result in 100% because we followed the same methodology of computing the rates as used in the earlier experiments (Schonlau and DuMouchel, 2001; Maxion and Townsend, 2002).

To get the results shown in Table 4, we used SVM^{Light} using radial based function (RBF) kernel. Our experiment using SEA configuration demonstrates that SVM is superior to Naive Bayes in terms of accuracy but inferior when false alarm rate is compared. Also we repeated the experiment conducted by Maxion and Townsend, in the 1 vs 49 configuration, and compared the effectiveness of SVM-based masquerader detection to that of Naive Bayes technique. SVM, in the 1 vs 49 configuration, demonstrated exceptional performance in all aspects compared to Naive Bayes method. Both experiments were conducted in the experimentally optimal setting, previously labelled as U30 F35 (26). It must be emphasized that this experiment does not prove that SVM-based masquerade detection is always free from false positives but that SVM is more effective than other machine learning technique.

Table 4 Experimental results using SVM

Data configuration	Methods	Accuracy (%)	Misses (%)	FA (%)
SEA	SVM	80.1	19.9	9.7
configuration	Naive Bayes	61.5	38.5	1.3
1 vs 49	SVM	94.8	5.2	0.0
Configuration	Naive Bayes	66.2	33.8	4.6

Experiment II: using enriched commands

Experimental design

In the subsequent experiment, we used the Greenberg data set and configurations used in previous experiment (Maxion, 2003). From original data, 50 users are selected for training and testing, and truncated command lines and enriched command lines were extracted for the different two experiments. Each user data contains 2,300 commands divided into blocks of 10 commands each. One thousand command lines in the beginning were used for training and the next 1,300 command lines were used for classification. In order to develop profile for each user, RBF kernel was used and the parameter values were decided by cross validation technique.

During the experiment, we used fixed-sized sliding window scheme. That is, we further divided each block into six different sub-blocks, each containing five commands structured into a sliding window of size 1.

In the first phase of this experiment, we used the truncated command and chose a simple feature of counting how many times each unique command was used. Based only on frequency information, SVM classifier decides if each of the six sub-blocks represents masquerader activities or not. Majority voting algorithm was applied to determine if each 10 command sequence is to be treated as commands entered by masqueraders.

In the next phase, we attempted to use enriched command lines as features to improve accuracy of masquerader detection. An enriched command is a concatenation of the whole command line typed, including flags, arguments and items of shell grammar (such as & or ≫) together with the expansion of any alias employed.

Experimental results

ROC curves on the truncated and the enriched version of the Greenberg's data are presented in Fig. 7. The ROC curves were obtained by stepping the value of the threshold for the voting engine. The dotted curve applies to the truncated data; the bold curve applies to the enriched data. Lenient decision criteria allow a higher hit rate, but also a higher false alarm rate. More stringent criteria tend to reduce both rates. Each point on

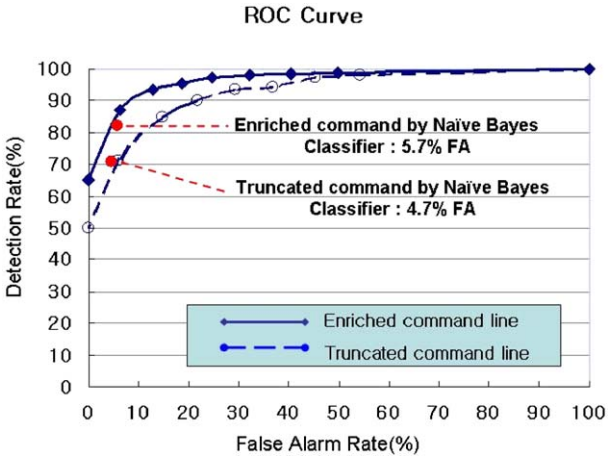


Figure 7 Comparison of ROC curve by SVM^{light} using RBF kernel.

the curves indicates a particular trade-off between hits and false alarms. The curve for the enriched data shows better performance than that of the curve for the truncated data.

Fig. 7 illustrates that, as was the case in the previous experiment, SVM-based approach was more effective than Naive Bayes-based technique. From the figure, it may appear that, in the view point of hit rate and false alarm, the performance of SVM appears almost the same or better than that of Naive Bayes classifier. When the truncated command used, the result of Naive Bayes classifier is slightly better than that of SVM.

Table 5 summarizes the results of detection methods according to the data type. When enriched commands were used, accuracy of masquerader detection improved from 82.1% to 87.3%. SVM is more accurate than Naive Bayes, and chances of failing to detect masquerade — misses — are lower (i.e., 17.9% vs 12.7%).

Although the notion of common commands were also used in the second experiment, it did not make tangible contribution in improving the accuracy of masquerade detection. In the case of Schonlau data, of the 824 unique commands, 26 of them (or 3.2%) were commonly used (e.g., by more than 35 of 50 users with the frequency exceeding 30%). In the Greenberg data, however, commands used by users were less common. Only 13 of more than 2,251 truncated (or 3,737 enriched) commands were considered common by the same criteria. This finding further emphasizes the importance of feature selection in the masquerader detection.

Table 5 Results comparison for Naive Bayes and SVM on Greenberg data, averaged across all users

Data type	Truncated		Enriched	
Method	Naive (Maxion, 2003)	SVM	Naive (Maxion, 2003)	SVM
Training data size	1000		1000	
Block size	10		10	
Hits (%)	70.9	71.1	82.1	87.3
Misses (%)	29.1	28.9	17.9	12.7
FA (%)	4.7	6.0	5.7	6.4

Conclusion and future work

In this paper, we demonstrated the effectiveness of SVM-based masquerade detection architecture. In the first experiment in which truncated UNIX commands were used, SVM was shown to be more effective than Naive Bayes. The second experiment using enriched commands, including alias and argument values, demonstrates that having additional information improved accuracy of masquerade detection while lowering the rate of false alarm. SVM was again shown to be superior to Naive Bayes method in detecting masquerades.

While our experiment convincingly demonstrated that SVM is the most effective masquerade detection method available to date, it is still far from being perfect. To deploy masquerade detection system in practice, we have to further reduce the false alarm and increase the detection rate by carefully analyzing security threat and defining effective feature sets. Additional data such as CPU usage, memory usage, or file accesses may be helpful. Unfortunately, there is no publicly available data set using which empirical studies can be performed.

However, it does not necessarily mean that the technique is not useful to security administrators whose responsibilities are to securely manage computer systems from the threats of outside attacks as well as insider abuses. Manual analysis of huge amount of log data is simply impractical, and the proposed technique can significantly increase productivity of security engineers.

As for false alarms, not all the alarms need to be treated equally serious. Alarms and related log data provide invaluable suggestions as to which incidents must be further analyzed using other techniques including manual investigation. When implementing selective investigation practices, "interpreted and educated" alarms, though still imperfect, the proposed technique would make critical contributions.

Topics worthy of further research include: (1) approaches to update user behavior model without

causing excessive number of false alarms when genuine usage profile of a user changes; (2) trade-off between the variety and effectiveness of features to be analyzed including session or time information.

Acknowledgement

Authors would like to thank Dr. Jahwan Kim of KAIST for his comment on our experiment and paper. Research reported in this paper has been funded, in part, by research funding to AITrc (http://aitrc.kaist.ac.kr), SPIC (http://spic.kaist.ac.kr) and IIRTRC (http://iitrc.cnu.ac.kr).

References

Anderson JP. Computer security threat monitoring and surveillance Technical report. Fort Washington, Pennsylvania: James P. Anderson Company; April 1980.

Burges Christopher JC. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 1998;2(2):121—67.

Chang Chih-Chung, Lin Chih-Jen. LIBSVM — a library for support vector machines, <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.

Dumais Susan T, Chen Hao. Hierarchical classification of Web content. In: Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval; 2000. p. 256—63.

Fugate Mike, Gattier James R. Anomaly detection enhanced classification in computer intrusion detection. SVM 2002, LNCS 2388; May 2002. p. 186—97.

Greenberg Saul. Using Unix: collected traces of 168 users. Research report 88/333/45, Department Computer Science, University Calgary, Calgary, Canada; 1988.

Hsu Chih-Wei, Chang Chih-Chung, Lin Chih-Jen. A practical guide to support vector classification, <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>; 2004.

Joachims Thorsten. Estimating the generalization performance of a SVM efficiently. In: Proceedings of ICML-00, 17th International Conference on Machine Learning; 2000. p. 431—8.

Lane Terran D. Machine learning techniques for the computer security domain of anomaly detection. Ph.D. thesis, Department of Electrical and Computer Engineering, Purdue University; Aug. 2000.

Lee Wenke, Stolfo Salvatore. Data mining approaches for intrusion detection. In: Proceedings of 7th USENIX Security Symposium, San Antonio, TX; 1998.

Maxion Roy. Masquerade detection using enriched command lines. In: Proceedings of international conference on dependable systems and networks (DSN-03); June 2003. p. 5—14.

Maxion Roy, Townsend Tahilia N. Masquerade detection using truncated command line. In: Proceedings of international conference on dependable systems and networks (DSN-02); June, 2002. p. 219—28.

Mukkamala S, Janowski G, Sung AH. Intrusion detection using support vector machines. In: Proceedings of IEEE international joint conference neural networks; May 2002. p. 1702—7.

Ptacek Thomas H. Insertion, evasion, and denial of service: eluding network intrusion detection, <http://secinf.net/info/ids/idspaper/idspaper.html>; Oct. 16, 2002.

Rain Forest Puppy. A look at whisker's anti-IDS tactics, <http://www.wiretrip.net/rfp/txt/whiskerids.html>; Dec. 1999.

Schonlau Matthias, DuMouchel William, Ju Wen-Hua, Karr Alan F, Theus Martin, Vardi Yehuda. Computer intrusion: detecting masquerades. Statistical Science Feb. 2001;1(16):58—74.

Webster William H, and others. A review of FBI security programs, U.S. Dept. Justice; Mar. 2002.

Yang Yiming, Liu Xin. A re-examination of text categorization methods. In: Proceedings of SIGIR-99, 22nd ACM international conference on research and development in information retrieval; 1999. p. 42—9.

Han-Sung Kim received BSc degree in Computer Science from Korea Military Academy in 1990. He also received MSc of Computer Science from the University of Western Ontario, Canada, in 1995. He is currently a PhD candidate at KAIST. His research interests include software engineering and computer security, especially in intrusion detection system.

Sung-deok Cha received the BS, MS and PhD degrees in Information and Computer Science from the University of California, Irvine, in 1983, 1986, and 1991, respectively. From 1990 to 1994, he was a member of the technical staff at Hughes Aircraft Company, Ground Systems Group, and the Aerospace Corporation, where he worked on various projects on software safety and computer security. In 1994, he became a faculty member of the Korea Advanced Institute of Science and Technology (KAIST), Electrical Engineering and Computer Science Department. His research interest includes software safety, formal methods, and computer security.