IEICE TRANS. INF. & SYST., VOL.E87–D, NO.11 NOVEMBER 2004

PAPER

Efficient Masquerade Detection Using SVM Based on Common Command Frequency in Sliding Windows

Han-Sung KIM[†a)], *Student Member and* **Sung-Deok CHA**[†b)], *Nonmember*

SUMMARY Masqueraders who impersonate other users pose serious threat to computer security. Unfortunately, firewalls or misuse-based intrusion detection systems are generally ineffective in detecting masqueraders. Anomaly detection techniques have been proposed as a complementary approach to overcome such limitations. However, they are not accurate enough in detection, and the rate of false alarm is too high for the technique to be applied in practice. For example, recent empirical studies on masquerade detection using UNIX commands found the accuracy to be below 70%. In this research, we performed a comparative study to investigate the effectiveness of SVM (Support Vector Machine) technique using the same data set and configuration reported in the previous experiments. In order to improve accuracy of masquerade detection, we used command frequencies in sliding windows as feature sets. In addition, we chose to ignore commands commonly used by all the users and introduce the concept of voting engine. Though still imperfect, we were able to improve the accuracy of masquerade detection to 80.1% and 94.8%, whereas previous studies reported accuracy of 69.3% and 62.8% in the same configurations. This study convincingly demonstrates that SVM is useful as an anomaly detection technique and that there are several advantages SVM offers as a tool to detect masqueraders.

key words: *intrusion detection, masquerade detection, anomaly detection, machine learning, SVM (Support Vector Machine), user command*

1. Introduction

A masquerader is someone who pretends to be another user while invading target user's accounts, directories, or files [1]. The term can be extended to include legitimate but suspicious use of privilege by insiders. For example, a user who logged on as root to a UNIX system is authorized to read any file. However, it is highly unusual for a root user to sequentially read all the files stored in a directory — especially if they belong to an ordinary user — 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 has 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 [2] 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 can detect such attacks only if intruders execute the attack codes whose signatures are known. Techniques to evade IDS products have been published [3], [4], and misuse detection systems can provide only limited help. Access control mechanism is unlikely to be an effective solution to insider attacks if an inside attacker can install sniffing software or rely on social engineering techniques. In fact, according to the most recent CSI/FBI annual survey on computer crime, insider abuse of net access and unauthorized access by insiders were reported by 78% and 38% of the respondents, respectively [5].

Anomaly detection has long been known as an essential component in securing computers and networks. Basic approaches to anomaly detection are well established, and techniques such as statistical analysis [6], data mining [7], and various machine learning techniques [8] have been used. Masquerade detection is a form of anomaly detection because behavior of a masquerader is assumed to be different from that of a real user. Unfortunately, anomaly detection techniques are not widely used in practice primarily because error rate is still too high for the technique to become practical.

In this paper, we empirically demonstrate that SVM, a machine learning technique, is useful in detecting masquerade activities. SVM has been successfully used in applications such as bioinformatics, text categorization, character recognition, handwriting recognition, facial and object detection, and intrusion detection [9]. We repeated the earlier experiments, performed by Schonlau et al. [6], using the same data set and the configuration so that effectiveness of SVM can be objectively and quantitatively compared.

The rest of this paper is organized as follows. In Sect. 2, we briefly explain related work on masquerade detection. In particular, following a brief introduction to controlled experiments on masquerade detection, we discuss several advantages SVM offers over other approaches used in anomaly detection and review how SVM has been used in intrusion detection research. In Sect. 3, we propose an architecture for the SVM-based masquerade detection. In Sect. 4, we briefly explain our experimental design with emphasis on feature selection. In addition, our experimental results are compared to those reported in previous experiments in which the same data set was used. Section 5 concludes the paper.

Manuscript received August 4, 2003.
Manuscript revised February 26, 2004.

[†]The authors are with the Division of Computer Science, Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), 373–1 Guseong-Dong, Yuseong-Gu, Daejeon 305–701, Korea.
a) E-mail: kimhs@salmosa.kaist.ac.kr
b) E-mail: cha@salmosa.kaist.ac.kr

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

	Techniques	Accuracy (%)	Misses (%)	False Alarms (%)
Schonlau et al. [6] (SEA Conf.)	Uniqueness	39.4	60.6	1.4
	Bayes 1-step Markov	69.3	30.7	6.7
	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 et al. [10]	Naive Bayes (updating)	61.5	38.5	1.3
	Naive Bayes (no Upd.)	66.2	33.8	4.6

2. Related Works

2.1 Masquerade Detection Experiments

Schonlau et al. [6] conducted an experiment on masquerade detection using user commands collected by UNIX acct auditing mechanism. Only the command names were used in the experiment, and various attributes associated with commands (e.g., timestamps, CPU usage, etc.) were ignored. They collected 15,000 commands from each of 70 users and randomly chose 50 of them as target users. Commands entered by the rest, 20 users, were used to simulate masquerade activities. Each command set was decomposed into 150 blocks 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 to 24 command blocks as means of simulating logs left by masqueraders. As shown in Table 1, six different approaches to masquerade detection were evaluated, and the accuracy of the most effective masquerade detection technique failed to exceed 70%. In addition, there were too many false alarms for the technique to be trusted[†].

Maxion and Townsend [10] conducted another experiment using the same data. However, they used a different configuration, referred to as the 1vs49 configuration, as opposed to SEA configuration used in [6]. In the 1vs49 configuration, the first 5,000 commands entered by each user 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. 1vs49 configuration provides a richer set of masquerade data than the SEA configuration does in that. Whereas the SEA configuration used 0 to 24 blocks of commands, drawn from up to three users, 2,450 (i.e., 50 blocks $\times$ 49 usersx) command blocks were used as masquerader data in the 1vs49 configuration. Using Naive Bayes classifier, Maxion and Townsend were able to improve accuracy[††] of masquerade detection from 39.4% to 61.5% while maintaining comparable level of false alarms[†††] (i.e., 1.4% and 1.3%, respectively). Unfortunately, it is still not accurate enough.

2.2 Advantages of SVM-Based Anomaly Detection

SVM refers to a collection of machine learning algorithms developed by Vapnik and enhanced by others. Designed for binary classification, 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 [11]–[13]. For text categorization, SVM outperformed neural network and Naive Bayesian model. For network intrusion detection, SVM demonstrated superior performance to neural network [14]. In [15], 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. Based on the concept of structural risk minimization, 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 as features. 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.

Third, SVM is relatively easy to use compared to other machine learning techniques such as neural networks. 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 [17]. For example, libSVM [18] provides a graphical tool to assist users to choose parameter values. In addition, unlike most machine learning techniques which require hard coding, 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

[†][6] argues that the rate of false alarm must not exceed 1% for the technique to be considered reliable enough to be routinely used in practice.

[††]In the SEA configuration, total 231 masquerader 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.

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

Table 2 Related work details.

	Audit Data	Detection Methods	# of Features	Results (%)	Remarks
New Mexico Institute of Mining and Tech. [14]	DARPA KDD 99 Data	Neural networks (3, 4-layer feed-forward NN)	41	99.25	
		SVM (RBF)	41	99.50	SVM^{Light}
NMT [16]	Command and HTTP	SVM	8	94.00	
Los Alamos Lab [15]	DARPA KDD 99 Data	Mahalanobis Outlier detection	41	90.30	
		SVM (RBF)	41		SVM^{Light}, libSVM

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 [11].

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 seconds for each user to generate a model based on 24,000 data with 824 features. Time interval varied from 0.9 second to 123 seconds. 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 convincingly demonstrates that real-time masquerade detection is possible with SVM.

2.3 Use of SVM in Intrusion Detection Research

As reported in [14]–[16] and summarized in Table 2, SVM has been used in several intrusion detection studies.

Experiment reported in [14] was conducted using DARPA KDD 99 data, available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. This data set has 41 attributes on network packets, and there are 500,000 training data and 300,000 test data. Performance of three different configurations of feed-forward neural networks[†] and that of SVM was compared. The study found both approaches to be accurate and compatible in intrusion detection capability. However, the average training time required for SVM, 17.8 seconds, was significantly shorter than that of the neural networks.

In [16], UNIX command and HTTP audit data collected by a Web server were analyzed and grouped into predefined categories. Different weights were assigned depending on the command or type and category of fingerprints produced by the web requests. Analysis of simulated intrusive activities revealed that SVM is an effective tool for intrusion detection having accurately detected 94% of intrusions.

In [15], Mahalanobis outlier detection technique, one of the distance-based outlier method, was compared against SVM using the same DARPA KDD data. In this experiment, profiles corresponding to normal usage were developed as well as those of various intrusive activities such as denial of service attacks and probing. They reported that different types of attacks resulted in different profiles and that two techniques demonstrated comparable level of performance. Experiments reported in [14] and [15] obtained different results even though they used the same data set and SVM software tool because (1) they simulated different types of intrusive activities; and (2) parameter values used in the experiments were different.

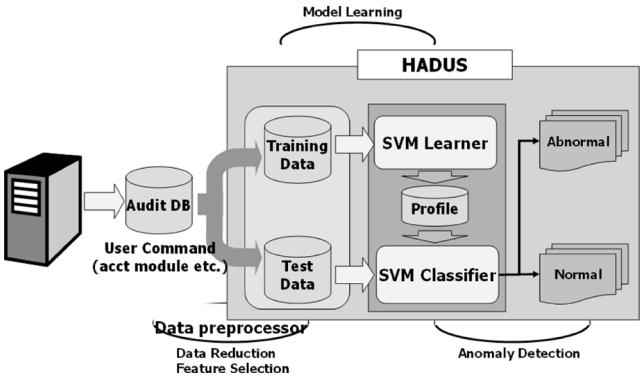


Fig. 1 Overall system architecture.

3. Proposed Approach: HADUS

Figure 1 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 (i.e., training 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, data must first be 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

[†]Three different feed-forward neural networks with the following architectures were used.:
Network A: 4-layer, 41-20-20-20-1,
Network B: 3-layer, 41-40-40-1,
Network C: 3-layer, 41-25-20-1.

they are to be considered normal or abnormal.

3.1 Data Preprocessor and Feature Selection

Feature selection is the most critical factor in 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 useful in distinguishing behavior of an authorized user from that of a masquerader. For example, a secretary who spends most of working hours performing text processing tasks is highly unlikely to issue commands often used by programmers (e.g., "`gcc`") or by system administrators (e.g., "`ping`"). However, there certainly exists a possibility that a secretary issues such commands out of curiosity or as a part of learning activities related to personal career development goals.

The data preprocessor performs feature selection process as shown in Fig. 2. To build user profile using SVM, "*command frequencies in fixed window (sub-block) size*" is used. Decision on the window size is critical. A preliminary study, in which command frequencies of 100 command blocks as a unit, resulted 100% of false positives for all 50 users due to insufficient training data. In principle, the window size must be kept as small as possible to allow near-real-time masquerade detection. On the other hand, it must be large enough to allow meaningful statistical analysis from a sequence of commands issued. In our study, we divided the command sequence into a block of 100 commands each as previous studies did so that objective and quantitative comparison can be made. Each block is further divided into sub-blocks of 50 commands. Each command name is given an unique ID, and the frequencies of each command appearing in each sub-block were calculated as shown in Fig. 2. The shaded part in the examples of features window size 50, sliding size 10 represents that user 1's first block has 5 `cpp`s, 2 `sh`s, 3 `xrdb`s, etc.

In addition to frequencies, we introduced the notion of "*common commands*" as an additional feature. There are commands that are likely to be used quite frequently, such as "`cd`" or "`ls`" commands, by most users. On the other hand, some UNIX commands, such as "`reboot`" or "`shutdown`," are most likely to be executed only by a system administrator having the root privilege. The notion of "*common commands*" provides a mechanism to better reflect diverse command usage patterns among different users. 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. 2, some commands including ID 1 and 3 are treated as one command, "*common*."

3.2 Classification with Voting Engine

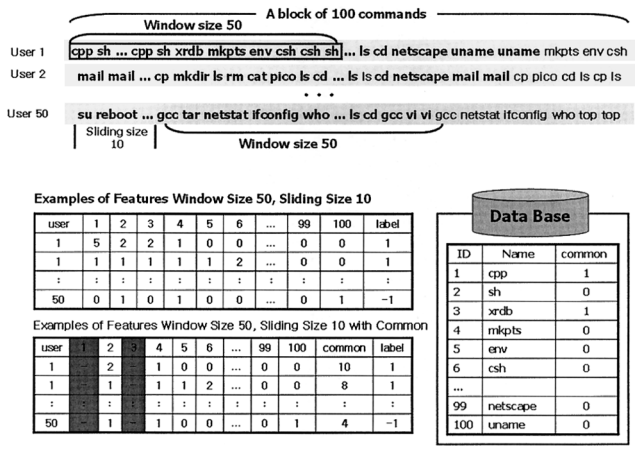
In the first experiment, each block of 100 commands was divided into six different sub-blocks, each containing 50 commands with a sliding window of size 10. SVM predictor determines if each sub-block is normal or not as shown in Fig. 3. The "*voting engine*" decides if the total block is to be considered as being anomalous. If the number of masquerader sub-blocks exceed threshold value, the block is considered as masquerader block. The voting engine is designed to reduce false alarm due to data noise.

Other factors critical to the effectiveness and performance of HADUS is 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 too much overhead, and tools are available to guide users in the decision process.

4. Experimental Design and Results

4.1 Experimental Design

An experiment on masquerade detection should ideally be performed using "live" data containing real, as opposed to simulated, masquerade activities. Unfortunately, such data is practically impossible to obtain. As an alternative, we



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

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

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

Fig. 2 Feature selection for SVM profiling.

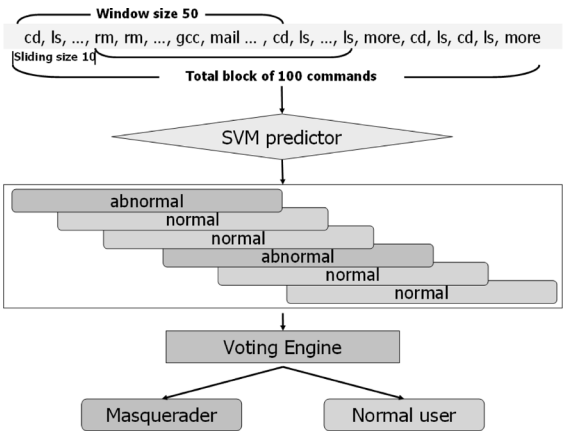


Fig. 3 Voting engine in SVM classifier.

have decided to use the command set that has been used in the previous experiment [6], [10] and is available in the public domain. In this research, we used the same data set and configurations so that objective and quantitative comparison can be made. In order to develop a profile for each user, RBF kernel was used and the parameter values were decided by cross validation technique. However, how each block of 100 commands was analyzed is different from the previous studies.

In the first phase of experiment, we used the SEA configuration reported in [6] and chose frequency of commands used in each block as the only feature. That is, we further divided each block into six different sub-blocks, each containing 50 commands structured into a sliding window of size 10. For example, if commands in each block were numbered 1 through 100, commands 1 through 50 constitute the first sub-block, 11 through 60 the second sub-block, etc. Each block is simply a sequence of commands, and there were no arguments provided for privacy reasons. Based only on frequency information, SVM classifier decides if each of the six sub-blocks represent masquerade activities or not. Majority voting algorithm was applied to determine if each 100 command sequence is to be treated as commands entered by masqueraders. Unfortunately, command frequency alone, as expected, is a poor feature in detecting masqueraders in that the rate of false positive is 21.58% which is unacceptably high in practice.

In order to improve accuracy of anomaly detection while lowering the rate of false alarms, we chose to introduce the notion of "*common commands*" as an additional feature in the classification process as discussed before. We revised our feature set to include only the commands that are likely to better reflect usage patterns distinct among different users.

4.2 Experimental Results

4.2.1 SEA Results

In Fig. 4, 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%. We repeated experiment using various values of X and Y to investigate the effect of introducing common 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. 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. 4 illustrate how the false positive rate varies if different decision voting algorithm is applied. For example, when the occurrence of up to three, out of six, masquerader sub-blocks are to be treated as noise, as shown with the threshold level 4, the false positive rate was the lowest. If occurrence of any anomalous sub-block causes the entire block to be the false declared suspicious, positive rate was the highest.

As further illustrated in Fig. 5, as we increase the minimum number of sub-blocks required to be suspected of containing suspicious commands, the rate of false positive was reduced from 14.04% to 9.71%. Unfortunately, stricter requirements also resulted in the loss of accuracy from 95.67% to 80.09%. As expected, there is a trade-off between the accuracy of masquerade detection and the rate of false positives, and each site needs to determine the circumstances that best serve its security needs.

It must be noted that the sum of accurate masquerade detection rate and the rates of false detection, positive and negative included, does not add up to 100% because we followed the same methodology of computing the rates as used in the earlier experiments [6], [10]. We applied the same approach in order to make quantitative comparison of performance meaningful.

In summary, using the SEA configuration, we were able to achieve 15.6% improvement in the accuracy of masquerade detection. That is, while minimizing the rate of false positives, we were able to improve the accuracy to 80.09% compared to 69.3% reported by Schonlau et al. using Bayes 1-step Markov modelling. In the same configuration, SVM-based approach resulted in 19.91% of false neg-

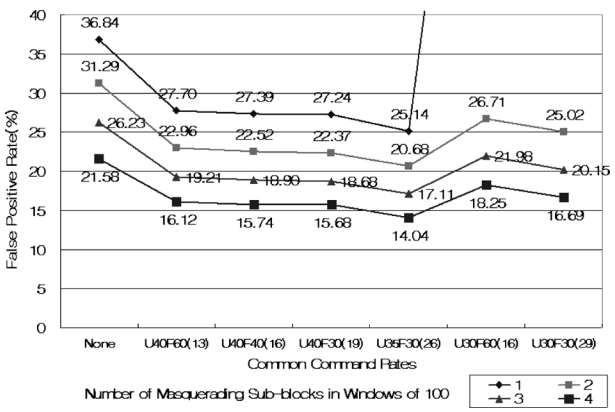


Fig. 4 The effect of common command.

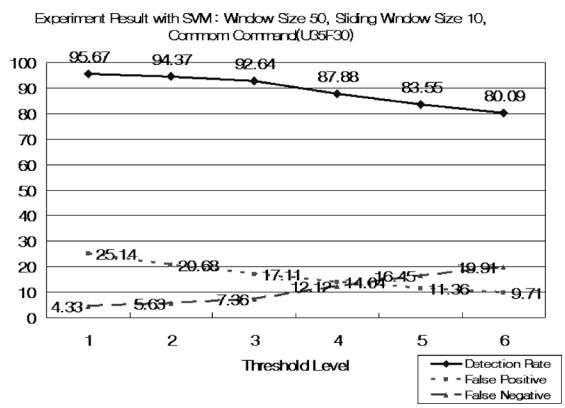


Fig. 5 The effect of decision maker in SEA configuration.

Table 3 Confusion matrix by SVM.

Victim	Intruder											Intrusion Missed
	1	2	3	4	5	6	7	8	9	10	...	
1	0(0)	0(2)	1(40)	18(47)	6(41)	0(1)	5(40)	0(3)	3(50)	0(9)	...	240 (1296)
2	0(0)	0(0)	3(3)	3(3)	0(4)	0(0)	0(4)	0(6)	0(0)	1(1)	...	13 (359)
3	5(20)	0(0)	0(0)	0(15)	1(12)	0(1)	3(16)	0(1)	0(3)	0(9)	...	100 (549)
4	8(35)	1(3)	0(2)	0(0)	3(44)	0(0)	2(36)	1(14)	0(48)	0(5)	...	199 (1110)
5	1(10)	0(3)	1(2)	1(18)	0(0)	0(0)	6(32)	0(3)	1(34)	0(2)	...	244 (1028)
6	0(13)	0(0)	0(0)	0(1)	0(0)	0(0)	0(9)	0(0)	0(9)	0(5)	...	34 (212)
7	4(36)	0(3)	1(4)	2(44)	2(50)	0(2)	0(0)	0(3)	2(49)	0(4)	...	319 (1299)
8	0(20)	0(1)	0(2)	0(38)	0(36)	0(0)	1(9)	0(0)	0(14)	0(0)	...	49 (611)
9	0(37)	0(3)	0(2)	1(45)	3(46)	0(1)	1(4)	0(2)	0(0)	0(14)	...	113 (1162)
10	2(9)	0(0)	0(2)	0(25)	0(0)	0(3)	0(4)	0(2)	0(26)	0(0)	...	37 (365)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Intrusion Succeeded	103 (1124)	9 (402)	34 (339)	117 (1405)	221 (1617)	33 (74)	121 (1379)	56 (360)	49 (1359)	3 (270)	...	

Table 4 Experimental results using SVM.

Data Configuration	Methods	Accuracy (%)	Misses (%)	False Alarms (%)
SEA Configuration	SVM	80.1	19.9	9.7
	Naive Bayes	61.5	38.5	1.3
1vs49 Configuration	SVM	94.8	5.2	0.0
	Naive Bayes	66.2	33.8	4.6

atives where masquerade activities went undetected. However, in the same configuration, in [6], the corresponding rate was 30.7%.

4.2.2 1vs49 Results

We also repeated the experiment conducted by Maxion and Townsend, in the 1vs49 configuration, and compared the effectiveness of SVM-based masquerade detection to that of Naive Bayes technique. This experiment was conducted in the experimentally optimal setting, previously labelled as U30 F35 (26). For each user, 2,450 (i.e., 49×50) masquerading blocks were simulated, and Table 3 describes partial confusion matrix. Numbers in parenthesis were reported in [10]. For the user 1, SVM-based approach missed 240 masquerader blocks, while masquerade detection using Naive Bayes missed 1,296 blocks.

As summarized in Table 4, in the 1vs49 configuration, SVM-based approach achieved 94.8% accuracy in masquerade detection, and there were no false alarms[†].

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.

5. Conclusion and Future Works

Research on masquerade detection is still at an early stage, and researchers are still trying to identify approaches that are reliable enough to be usable in practice while providing a low enough rate of false accusation. Given the current state of art in masquerade detection technology, an organization must be prepared to invest necessary resource into investigating incidents that might turn out to be authorized (perhaps a bit unusual) activities of legitimate users if potential security threat caused by masqueraders is too great a risk to tolerate. In order to have an effective masquerade detection capability in place, an organization must carefully analyze security threat, define useful feature set to be monitored, and continuously collect and analyze log data.

In the experiment reported in the paper, we introduced the notions of "common command", "sub-blocks" with sliding windows and "voting engines." Including additional attributes could further improve accuracy of masquerade detection while lowering the rate of false alarm. For example, session information as well as the time each command was issued might contribute in developing accurate user profile.

Despite limitations in the experimental design and still-too-high rate of false alarm, SVM is shown to be a useful tool in detecting masqueraders. Our experiment demonstrates that in small-to-medium sized organizations, near real-time masquerade detection is possible with SVM.

While our experiment obtained the best results available to date, it is still far from being perfect. Topics worthy

[†]Full details on experimental designs and results are available at http://salmosa.kaist.ac.kr/~kimhs/research

of further research include: 1) Approaches to update user behavior model without causing excessive number of false alarms when genuine usage profile; 2) Trade-off analysis between variety and effectiveness of features because more features do not necessarily improve accuracy of masquerade detection.

Acknowledgement

Authors would like to thank Prof. Roy Maxion of CMU for his cooperation of sharing the full confusion matrix. Research reported in this paper has been funded, in part, by research funding to AITrc (http://aitrc.kaist.ac.kr) to SPIC (http://spic.kaist.ac.kr) and to IIRTRC (http://iirtrc.cnu.ac.kr).

References

[1] J.P. Anderson, "Computer security threat monitoring and surveillance," Technical Report, James P. Anderson Company, Fort Washington, PA, April 1980.
[2] W. Webster, C. Alexander, G. Bell, W. Cohen, R. Fiske, T. Foley, and C. Hills, "A review of FBI security programs," March 2002.
[3] T.H. Ptacek, "Insertion, evasion, and denial of service: Eluding network intrusion detection," http://secinf.net/info/ids/idspaper/idspaper.html, Oct. 2002.
[4] R.F. Puppy, "A look at whisker's anti-IDS tactics," http://www.wiretrip.net/rfp/txt/whiskerids.html, Dec. 1999.
[5] CSI and FBI, "Computer security issues and trends: 2002 CSI/FBI computer crime and security survey," Computer Security Inst., 2002.
[6] M. Schonlau, W. DuMouchel, W.H. Ju, A.F. Karr, M. Theus, and Y. Vardi, "Computer intrusion: Detecting masquerades," Statistical Science, vol.16, no.1, pp.58–74, Feb. 2001.
[7] W. Lee and S. Stolfo, "Data mining approaches for intrusion detection," Proc. 7th USENIX Security Symp., pp.79–94, San Antonio, TX, 1998.
[8] T.D. Lane, Machine Learning Techniques for the Computer Security Domain of Anomaly Detection, Ph.D. Thesis, Department of Electrical and Computer Engineering, Purdue University, Aug. 2000.
[9] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition," Data Mining and Knowledge Discovery, vol.2, no.2, pp.121–167, 1998.
[10] R. Maxion and T. Townsend, "Masquerade detection using truncated command lines," Proc. Int'l Conf. Dependable Systems and Networks (DSN-02), pp.219–228, June 2002.
[11] T. Joachims, "Estimating the generalization performance of a SVM efficiently," pp.431–438, 2000.
[12] Y. Yang and X. Liu, "A re-examination of text categorization methods," Proc. SIGIR-99, 22nd ACM Int'l Conf. Research and Development in Information Retrieval, pp.42–49, 1999.
[13] S.T. Dumais and H. Chen, "Hierarchical classification of Web content," Proc. SIGIR-00, 23rd ACM Int'l Conf. Research and Development in Information Retrieval, pp.256–263, 2000.
[14] S. Mukkamala, G. Janowski, and A.H. Sung, "Intrusion detection using support vector machines," Proc. IEEE Int'l Joint Conf. Neural Networks, pp.1702–1707, May 2002.
[15] M. Fugate and J.R. Gattier, "Anomaly detection enhanced classification in computer intrusion detection," SVM 2002, LNCS 2388, pp.186–197, May 2002.
[16] S. Mukkamala, G. Janowski, and A.H. Sung, "Intrusion detection using support vector machines," Proc. High Performance Computing Symp.-HPC 2002, pp.178–183, April 2002.
[17] C.W. Hsu, C.C. Chang, and C.J. Lin, "A practical guide to support vector classification," http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, 2004.
[18] C.C. Chang and C.J. Lin, "Libsvm – A library for support vector machines," http://www.csie.ntu.edu.tw/~cjlin/libsvm/

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

Sung-Deok Cha received the B.S., M.S. and Ph.D. 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, Electrical Engineering and Computer Science Department. His research interest includes software safety, formal methods and computer security.