Online Boosting Based Intrusion Detection in Changing Environments

Yan-guo Wang
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
100080 Beijing, China
ygwang@nlpr.ia.ac.cn

Weiming Hu
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
100080 Beijing, China
wmhu@nlpr.ia.ac.cn

Xiaoqin Zhang
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
100080 Beijing, China
xqzhang@nlpr.ia.ac.cn

ABSTRACT

Intrusion detection is an active research field in the development of reliable web-based information systems, where many artificial intelligence techniques are exploited to fit the specific application. Although some detection algorithms have been developed, they lack the adaptability to the frequently changing network environments, since they are mostly trained in batch mode.

In this paper, we propose an online boosting based intrusion detection method, which has the ability of efficient online learning of new network intrusions. The detection can be performed in real-time with high detection accuracy. Experimental results show the advantage of the method in the intrusion detection application.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Invasive software (e.g., viruses, worms, Trojan horses), Unauthorized access (e.g., hacking, phreaking)*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*

General Terms

Security, Algorithms

Keywords

Intrusion detection, online boosting, pattern recognition

1. INTRODUCTION

With the increasing of network intrusions, information security becomes crucial in the development of web-based information systems, such as e-business and e-government. It provides us a reliable environment for communication, information integrity and privacy protection. As the complementary measures of intrusion preventions such as user authentication and encryption, intrusion detection techniques attract more and more attention from researchers and engineers.

Intrusion detection has been an active research field ever since its first proposed by Denning [2] in 1987. It is an effective tool for protecting our systems against various types of network attack. Generally, intrusion detection systems (IDSs) can be classified into two categories: host-based IDSs and network-based IDSs [10]. Host-based IDSs make use of the system log of the target host machines, and some rule-based detection algorithms can be derived. However, it may be late when an intrusion is detected by host-based IDSs, as damage to the system may have already occured. Moreover, it is difficult for host-based IDSs to detect distributed network attacks that aim to consume the system resources. Network-based IDSs perform detection at network nodes such as switchers and rooters. They exploit the information of separate IP packages and detect packages that are of potential harm to machines on the network. Within the architecture of network-based IDSs, the detection burden in the host machines is greatly lightened, and security measurements can be taken before the attack reaches the host machines. Furthermore, network-based IDSs are able to detect distributed network attacks.

There are various ways of intrusion detection that have been developed. A statistical method is proposed in [2], where statistical profiles for normal behaviors are constructed and used to detect anomalous behaviors as intrusions. Data mining techniques are also widely used in intrusion detection [13, 22]. The concepts of "association rules" and "frequent episodes" are introduced into the intrusion detection task to describe the network activities. In [15] a distributed outlier detection architecture is constructed to detect network attacks.

Recently, methods in machine learning and pattern recognition become popular in the intrusion detection research. Various new algorithms are introduced and studied in IDSs. For supervised learning, Bivens et al. [1] used neural networks for intrusion detection. Another algorithm with great generalization ability, SVM, is also applied in [3, 7]. As for unsupervised learning, an effective clustering tool, self-organizing map, attracts great attention in [9, 11, 18]. There are also many other pieces of work on this research field [12,

19].

Although many intrusion detection algorithms have been developed, there are still some problems in practical uses. First, there are many new intrusion types produced every week. Most of existing algorithms have to retrain the whole detector in order to detect the new types of attack, since they are trained in batch mode. Second, the training data set for intrusion detection is often very huge, so it becomes impractical for frequently retraining the detector to adapt to a dynamic environment. Third, the variety of attribute of network data is also a difficult issue. There are various types of attributes for network data, including both categorical and continuous ones. Furthermore, the value ranges for different attributes differ greatly, from $[0, 1]$ to $[0, 10^7]$. This brings more difficulties for many detection methods and limits their performance.

In this paper, the online boosting algorithm proposed by Oza [16] is applied into network-based intrusion detection. The online learning framework provides the ability of quick adaption to the changing environments. The carefully designed strategies for training weak classifiers make the learning efficient, and produce great performance for the final ensemble classifier. Experimental results show that the proposed method quickly adapts to the changing environments, and can perform real-time intrusion detection with high detection accuracy.

The rest of the paper is organized as follows. In Section 2 we introduce the online boosting based intrusion detection algorithm, and provide its relation to the batch boosting based detection scheme. In Section 3 some experimental results are presented which show the advantage of the proposed method. Then we draw the conclusions in the last section.

2. ONLINE BOOSTING BASED INTRUSION DETECTION

We begin with a formulation of the intrusion detection task. After a brief introduction of the ideas of batch and online boosting, we present our online boosting based intrusion detection algorithm. Then an analysis of the computational complexity is followed.

2.1 Problem Formulation

In the network-based IDSs, the training and detection are performed at network nodes such as switchers and routers. Three groups of features are extracted from each network connection:

- basic features of individual TCP connections;
- content features within a connection suggested by domain knowledge;
- traffic features computed using as two-second time window.

The framework for constructing the above features for intrusion detection can be found in [14].

For each network connection, the feature values extracted above form a vector

$$\mathrm{x} = [x_1, x_2, ..., x_d], \tag{1}$$

where d is the number of features extracted. The label y indicates the binary class of the network connection:

$$\begin{cases} y = 1 & \text{normal connection} \\ y = -1 & \text{network intrusion} \end{cases}. \tag{2}$$

The intrusion detection algorithm is expected to train a classifier H from the labeled training data set, then use the classifier H to predict the binary label $\tilde{y} = H(\mathrm{x})$ for a new network connection.

2.2 Adaboost Algorithm

Adaboost [5] is one of the most popular machine learning algorithms developed in recent years, which has been successfully used in many applications, such as face detection [21], and image retrieval [20].

The classical Adaboost algorithm is trained in batch mode, which is showed in Table 1. Note that N is the number of training samples, M is the number of weak classifiers to generate, and L_b is the base model learning algorithm, such as Naive Bayes and decision stumps. A sequence of weak classifiers are learned based on the evolving sampling distribution of the training data set. The final strong classifier is an ensemble of the weak classifiers, and the voting weights are derived from the classification errors of these weak classifiers.

The evolving weight $w_n^{(m)}$ plays a key role in Adaboost. It indicates the importance of the n-th training sample while generating the m-th weak classifier. The weight $w_n^{(m)}$ is updated in the following way:

$$w_n^{(m+1)} = w_n^{(m)} \times \begin{cases} \frac{1}{2(1-\epsilon^{(m)})} & \text{if } h^{(m)}(\mathrm{x_n}) = y_n \\ \frac{1}{2\epsilon^{(m)}} & \text{if } h^{(m)}(\mathrm{x_n}) \neq y_n \end{cases}. \tag{3}$$

The weights of samples that are wrongly classified by the current weak classifier are increased, the others decreased, so that more attention is paid to the samples that are difficult to classify while selecting the next weak classifier.

Theoretical proof has been given in [5], which implies the convergence of the weighted classification error for the final strong classifier:

$$\Sigma_{n:H(\mathrm{x_n})\neq y_n} w_n^{(1)} \to 0, \quad \text{as } M \to \infty. \tag{4}$$

2.3 Online Boosting

In many applications, learning process need to be performed in online mode. For example, when the training data are generated as data streams, or the size of the training data set is too huge for memory resources, it is impractical to give the entire training data set to the learning algorithm at a time. Training a classifier in online mode is necessary in these cases. This opens a hot research field called "online learning" or "incremental learning". Online learning means that we process a training sample, then discard it after updating the classifier. There is likely some difference between

Table 1: Adaboost Algorithm

Input: $\{(\mathrm{x}_1, y_1), ..., (\mathrm{x}_N, y_N)\}, M, L_b$.

Initialization: $w_n^{(1)} = 1/N, \quad n = 1, ..., N$.

For $m = 1, 2, ..., M$

$h^{(m)} = L_b(\{(\mathrm{x}_1, y_1), ..., (\mathrm{x}_N, y_N)\}, w^{(m)})$

Calculate the weighted error of $h^{(m)}$:
$\epsilon^{(m)} = \Sigma_{n:h^{(m)}(\mathrm{x_n}) \neq y_n} w_n^{(m)}$
If$\epsilon^{(m)} \geq 1/2$, then
set $M = m - 1$ and stop loop
end if

Update the weights:
$$w_n^{(m+1)} = w_n^{(m)} \times \begin{cases} \frac{1}{2(1-\epsilon^{(m)})} & \text{if } h^{(m)}(\mathrm{x}_n) = y_n \\ \frac{1}{2\epsilon^{(m)}} & \text{if } h^{(m)}(\mathrm{x_n}) \neq y_n \end{cases}$$

Output the final strong classifier:
$H(\mathrm{x}) = \mathrm{sign}(\Sigma_{m=1}^{M} h^{(m)}(\mathrm{x}) \cdot \lg \frac{1-\epsilon^{(m)}}{\epsilon^{(m)}})$.

the two classifiers that trained in batch and online modes, since the online learning algorithm can only make use of the information supplied by part of the training data set and the processed data can not be retrieved. Thus, the key issue of online learning research is how to control the above difference while training a classifier online.

In order to adapt Adaboost to the data streams environment, Oza proposed an online version of Adaboost in [16], and the convergence proof for the online version was also given. Recently, Grabner and Bischof [6] successfully introduced the online boosting algorithm into computer vision field to select features online.

The detailed online boosting algorithm is presented in Table 2. Here h is the set of weak classifiers to be updated online, L_o is the online base model learning algorithm. Note that in the batch Adaboost algorithm, the sum of sample weights remains 1:

$$\sum_{n=1}^{N} w_n^{(m)} = 1, \quad m = 1, ..., M, \tag{5}$$

where the definition of $w_n^{(m)}$ is already given in Section 2.2. While in online boosting, the weight λ evolves individually for each training sample. As to the weighted classification error of $h^{(m)}$, an approximation is used:

$$\epsilon^{(m)} = \frac{\lambda_m^{sw}}{\lambda_m^{sc} + \lambda_m^{sw}}, \tag{6}$$

which involves only samples already seen. Moreover, the number of weak classifiers is not fixed in Adaboost; while in online boosting, the number of weak classifiers is fixed beforehand, and the weak classifiers are all learned online.

Although it may differ greatly from that learned in batch mode when only a few training samples have been processed,

Table 2: Online Boosting Algorithm

Input: $\{(\mathrm{x}_1, y_1), ..., (\mathrm{x}_N, y_N)\}, M, \mathrm{h}, L_o$.

Initialization: $\lambda_m^{sc} = 0, \lambda_m^{sw} = 0, m = 1, ..., M$.

For each new training sample (x, y)

Initialize weight of the current sample $\lambda = 1$

For $m = 1, 2, ..., M$

Set k according to $Poission(\lambda)$
Do k times
$h^{(m)} = L_o((\mathrm{x}, y), h^{(m)})$

If $h^{(m)}(\mathrm{x}) = y$, then
$\lambda_m^{sc} = \lambda_m^{sc} + \lambda$
$\epsilon^{(m)} = \frac{\lambda_m^{sw}}{\lambda_m^{sc} + \lambda_m^{sw}}$
$\lambda = \lambda(\frac{1}{2(1-\epsilon^{(m)})})$
else
$\lambda_m^{sw} = \lambda_m^{sw} + \lambda$
$\epsilon^{(m)} = \frac{\lambda_m^{sw}}{\lambda_m^{sc} + \lambda_m^{sw}}$
$\lambda = \lambda(\frac{1}{2\epsilon^{(m)}})$
end if

Output the final strong classifier:
$H(\mathrm{x}) = \mathrm{sign}(\Sigma_{m=1}^{M} h^{(m)}(\mathrm{x}) \cdot \lg \frac{1-\epsilon^{(m)}}{\epsilon^{(m)}})$.

the online ensemble classifier converges statistically to the ensemble generated in batch mode, as the number of training samples increases [16]. The good performance of the online boosting algorithm is also showed in many experiments in [17].

2.4 Online Boosting Based Intrusion Detection

An intrusion detection algorithm is expected to fill mainly the following three requirements in order to be suitable for practical uses:

- the detection should be perform in real-time;
- the detection accuracy should be as high as possible, which means a high detection rate to guarantee the system security, and a low false alarm rate to decrease unnecessary human burden;
- the detector should adapt quickly to the changing network environments, which implies the ability to accurately detect any new type of attack soon after its emergence.

In order to make the updating of intrusion detector efficient, the training of the detector should not be time-consuming, which prevents the use of some complex classifiers; on the other hand, the strict requirement of detection performance makes the direct use of simple classifiers impractical. Consider the variety of attribute of network connection data, the

situation is even worse. To successfully apply the idea of online boosting into intrusion detection, the above difficulties for intrusion detection should be considered carefully in the design of weak classifiers.

We choose the weak classifiers to be the set of decision stumps on each feature dimension. Thus the number of weak classifiers M is fixed, which equals to the number of features d in (1). These weak classifiers are learned online, and the approximated weighted classification error (6) is updated when a new training sample comes.

For a categorical feature f, the set of attribute values $\mathcal{C}^f$ is divided into two subsets $\mathcal{C}_p^f$ and $\mathcal{C}_n^f$ with no intersection, and the decision stump takes the form as

$$h_f(\mathrm{x}) = \begin{cases} 1 & \mathrm{x_f} \in \mathcal{C}_p^f \\ -1 & \mathrm{x_f} \in \mathcal{C}_n^f \end{cases}, \tag{7}$$

where $\mathrm{x_f}$ indicates the attribute value of x on the feature f. To avoid the combinatorial computation of examining all possible decision stumps, the division of $\mathcal{C}^f$ is efficiently constructed in the following way:

$$z \in \begin{cases} \mathcal{C}_p^f & \sum_{\mathrm{x_f}=z} \delta(y=1) \geq \sum_{\mathrm{x_f}=z} \delta(y=-1) \\ \mathcal{C}_n^f & \sum_{\mathrm{x_f}=z} \delta(y=1) < \sum_{\mathrm{x_f}=z} \delta(y=-1) \end{cases}, \tag{8}$$

where z is an attribute value on the feature f in the training data set, $\delta(\cdot)$ equals to 1 if condition $(\cdot)$ satisfies, equals to -1 otherwise.

For a continuous feature f, the range of attribute values is split by a threshold v, and the decision stump takes the following form:

$$h_f(\mathrm{x}) = \begin{cases} 1 & \mathrm{x_f} \geq v \\ -1 & \mathrm{x_f} < v \end{cases} \text{ or } h_f(\mathrm{x}) = \begin{cases} -1 & \mathrm{x_f} \geq v \\ 1 & \mathrm{x_f} < v \end{cases}. \tag{9}$$

The threshold v and the above two cases are chosen to minimize the weighted classification error

$$\epsilon = \Sigma_{n:h_f(\mathrm{x_n}) \neq y_n} w_n. \tag{10}$$

The above design of weak classifiers has the following advantages in the intrusion detection task:

- the weak classifiers (7) and (9) only operate on individual feature dimensions, which avoids the difficulty caused by the large distinction of value ranges for different feature dimensions;
- the training of the decision stumps is simple, and the online updating can be efficiently implemented;
- the detection performance is guaranteed by the final ensemble of weak classifiers.

Thus, the proposed online boosting based method for intrusion detection seems suitable for practical uses.

2.5 Computational Complexity

Computational complexity is an important aspect for intrusion detection algorithms, since a fast response to network intrusions is necessary for taking subsequent security measurements in time. Particularly, when a new type of attack appears, the characteristic of the new pattern should be learned as soon as possible, and the ability of accurate detection of the new attack should be quickly incorporated into the detector.

For the proposed online boosting based intrusion detection method, the learning and detection can be done in parallel. When a new sample comes, we only need to update the decision stumps and the related ensemble weights for each feature dimension, so the computational complexity of the online boosting learning is only $O(d)$, where d is the number of features of each network connection. Thus the updating of the intrusion detector can be implemented very efficiently, and the detector is expected to response quickly to new types of attack. Moreover, the detection of the ensemble classifier also has a computational complexity of $O(d)$, which can be implemented in real-time.

3. EXPERIMENTS

3.1 Intrusion Data Set

The KDD Cup 1999 data set [4] is used in our experiments, since it is a widely used benchmark data set for many network-based intrusion detection algorithms. It was used for the 1998 DARPA intrusion detection evaluation program, which was prepared and managed by MIT Lincoln Labs. A network environment was set up to simulate a typical U.S. Air Force LAN, where a wide variety of intrusions were simulated like in a real military network. Nine weeks of TCP/IP connection data were collected, and they were labeled for testing intrusion detection algorithms.

For each network connection, 41 features are extracted, including 9 categorical and 32 continuous features. Attacks in the data set fall into four main categories:

- DOS: denial-of-service;
- R2L: unauthorized access from a remote machine, e.g., guessing password;
- U2R: unauthorized access to local superuser (root) privileges;
- Probing: surveillance and other probing, e.g., port scanning.

In each of the four categories, there are many low level attack types.

The numbers of normal connections and each category of intrusions in the training and test data sets are listed in Table 3. Note that the test data is not from the same distribution as the training data, and it includes some attack types not existing in the training data. This makes the task more realistic. For more details of the intrusion data set, please refer to [4].

3.2 Online Boosting Based Intrusion Detection

We use the following two measures to evaluate the performance of the detection algorithm:

$$\text{detection rate:} \quad \mathrm{DR} = \frac{N_{\mathrm{detected}}}{N_{\mathrm{attack}}} \times 100\%, \tag{11}$$

Table 3: The KDD Cup 1999 Data Set

Categories	Training data	Test data
Normal	97278	60593
DOS	391458	223298
R2L	1126	5993
U2R	52	39
Probing	4107	2377
Others	0	18729
Total	494021	311029

$$\text{false alarm rate:} \quad \text{FAR} = \frac{N_{\text{false}}}{N_{\text{normal}}} \times 100\%, \tag{12}$$

where N_{detected} denotes the number of attacks correctly detected, N_{attack} denotes the total number of attacks in the data set, N_{false} denotes the number of normal connections that are wrongly detected as attacks, and N_{normal} denotes the total number of normal connections.

We use the proposed online boosting based method to train a detector on the training data, then apply the detector to classify the test data. A high DR of 91.2756% is obtained on the test data. However, the FAR is 8.3805%, which is a little high for intrusion detection.

Generally, the more negative samples are concerned, the higher DR is obtained, with a higher FAR. Note that the normal samples only occupy 20% of the training data, which makes the detector pay too much attention to the attack samples. In fact, with the identical initial weight $\lambda = 1$ for each training sample in online boosting, the importance of each binary class relates to the number of positive and negative samples in the training data.

In order to balance the requirements of DR and FAR, we introduce a parameter $r \in (0, 1)$ in the setting of initial weight λ for each training sample:

$$\lambda = \begin{cases} \frac{N_{\text{normal}}+N_{\text{attack}}}{N_{\text{normal}}} \cdot r & \text{normal connection} \\ \frac{N_{\text{normal}}+N_{\text{attack}}}{N_{\text{attack}}} \cdot (1-r) & \text{network intrusion} \end{cases}.$$

Through adjusting the parameter r, we can change the importance of positive and negative samples in the training process, and then get a balance between DR and FAR. The selection of r depends on the proportion of normal samples in the training data, and the requirements of DR/FAR in the specific application.

After testing of some values between 0.05 and 0.95, we finally get an appropriate setting of $r = 0.35$. With this value of r, a much smaller FAR of 2.2374% is obtained, with a still acceptable DR of 90.1329%. This result is more suitable for the intrusion detection application.

For comparison, we also test the batch Adaboost algorithm on the intrusion data set. The detection results of the above algorithms on the test data are listed in Table 4. Note that although learned in online mode, the detection accuracy of the online boosting based method is comparable with that of Adaboost which is learned in batch mode, and an ideal small FAR can be obtained through appropriately adjusting of the parameter r.

Table 4: Detection Results on the Test Data

Algorithms	DR(%)	FAR(%)
Online boosting	91.2756	8.3805
Online boosting (with $r = 0.35$)	**90.1329**	**2.2374**
Adaboost	92.6568	2.6686

To focus on some specific intrusion types, Jirapummin et al. [8] employed a hybrid neural network model to detect TCP SYN flooding and port scan attacks. In order to examine the detection ability of the specific types of attack, a smaller data set that contains only three attack types is constructed. The samples are randomly selected from the KDD Cup 1999 data set, and the numbers of samples are listed in Table 5. Detection results on the test data are presented in Table 6, including also the results given by Sarasamma et al. [18] for comparison. We can see that the specific types of intrusions are accurately detected by the online boosting based method, and a smaller FAR is obtained compared with the other two methods.

3.3 Fast Adaption to Changing Environments

In the online boosting based intrusion detection method, the characteristic of a new type of attack is quickly learned in online mode, and the ability to detect the new attack is soon incorporated into the ensemble classifier. For example, the learning processes of two attack types, "guess-passwd"(a R2L attack) and "back"(a DOS attack), are showed in Figure 1 and Figure 2. The horizontal axis indicates the number of samples of this attack, and the vertical axis indicates the class label $\tilde{y}$ predicted by the detector. Note that $\tilde{y} = 1$ means the sample is classified as a normal connection, and $\tilde{y} = -1$ means a network intrusion. We can see that the new types of attack are accurately detected soon after online learning on a few samples.

3.4 Comparison with Some Other Algorithms

We also compare the proposed method with some other recent progresses on the intrusion detection research. For example, Eskin et al. proposed a geometric framework for unsupervised anomaly detection in [3], and three methods are presented to detect anomalies in the data set: a cluster-based approach, a k-nearest neighbor-based approach, and a SVM-based approach. Unlike most intrusion detection methods which use the whole 41 features, Kayacik et al. [9] used only 6 of the most basic features, and they performed intrusion detection based on self-organizing feature maps. Recently, a multi-layer hierarchical K-Map was proposed by Sarasamma et al. [18], and different combinations of feature subsets can be selected to detect intrusions in the hierarchical framework.

Detection results of the above methods are listed in Table 7. Note that these results are all obtained using the KDD Cup 1999 data set. We can see that the detection performance of the online boosting based method is very comparable with those of the other methods. Moreover, the online boosting based method outperforms the other methods in the ability of efficient online learning of new intrusion types, which is an attractive characteristic for intrusion detection in the frequently changing network environments.

Table 5: A Smaller Data Set

Categories	Training data	Test data
Normal	20676	60593
Neptune	22783	58001
Satan	316	1633
Portsweep	225	354
Total	44000	120581

Table 6: Detection Results for Some Specific Attacks

Algorithms	DR(%)	FAR(%)
Hybrid Neural Networks [8]	90-99.72	0.06-4.5
Hierarchical K-Map [18]	99.17-99.63	0.34-1.41
Online Boosting	**99.55**	**0.17**

4. CONCLUSIONS

In this paper, an efficient online boosting based intrusion detection method is proposed. The method outperforms existing intrusion detection algorithms in the adaptability to the changing environments. The ability to detect new types of attack is quickly incorporated into the ensemble classifier without retraining the whole detector, and the detection can be performed in real-time with high accuracy. Experimental results show the potential usage of the method in the intrusion detection application.

5. ACKNOWLEDGMENTS

This work is partly supported by NSFC (Grant No. 60520120099 and 60672040) and the National 863 High-Tech R&D Program of China (Grant No. 2006AA01Z453).

6. REFERENCES

[1] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, and M. Embrechts. Network-based intrusion detection using neural networks. In *Proc. of Artificial Neural Networks In Engineering*, November 2002.

[2] D. Denning. An intrusion-detection model. *IEEE Trans. on Software Engineering*, 13(2):222–232, February 1987.

[3] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Applications of Data Mining in Computer Security*, 2002.

[4] S. S. et al. The third international knowledege discovery and data mining tools competition [online]. available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

Table 7: Comparison with Some Other Algorithms

Algorithms	DR(%)	FAR(%)
Clustering [3]	93	10
K-NN [3]	91	8
SVM [3]	91-98	6-10
SOM [9]	89-90.6	4.6-7.6
Hierarchical K-Map [18]	90.94-93.46	2.19-3.99
Online Boosting	**90.1329**	**2.2374**

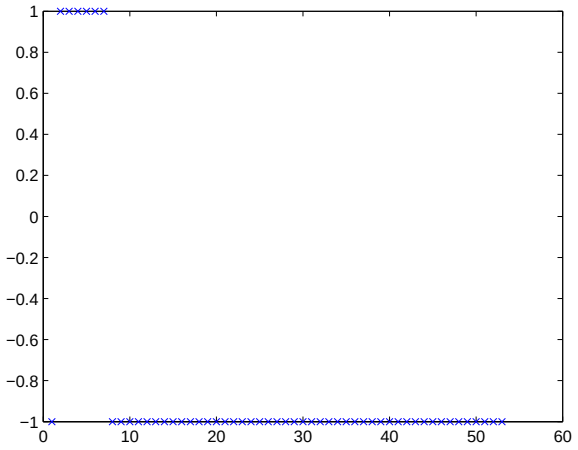

Figure 1: Online learning of "guess-passwd" attack.

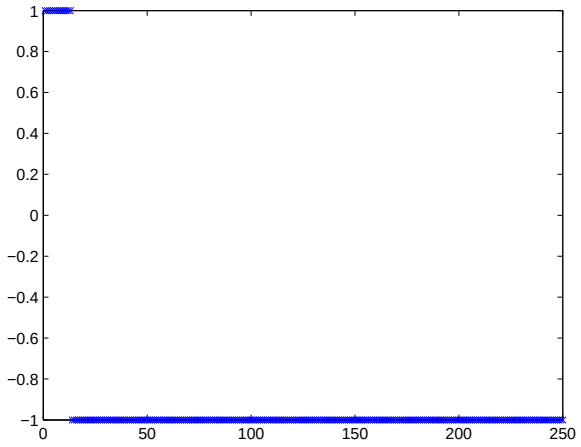

Figure 2: Online learning of "back" attack.

[5] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[6] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. of Computer Vision and Pattern Recognition*, pages 260–267, 2006.

[7] P. Hong, D. Zhang, and T. Wu. An intrusion detection method based on rough set and svm algorithm. In *Proc. of Int. Conf. on Communications, Circuits and Systems*, pages 1127–1130, June 2004.

[8] C. Jirapummin, N. Wattanapongsakorn, and P. Kanthamanon. Hybrid neural networks for intrusion detection systems. In *Proc. of The 2002 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2002)*, pages 928–931, July 2002.

[9] H. G. Kayacik, A. Zincir-Heywood, and M. Heywood. On the capability of an som based intrusion detection system. In *Proc. of Int. Joint Conf. on Neural Networks*, pages 1808–1813, 2003.

[10] R. A. Kemmerer and G. Vigna. Intrusion detection: A brief history and overview. *Computer*, 35(4):27–30, April 2002.

[11] K. Labib and R. Vemuri. Nsom: A real-time network-based intrusion detection system using self-organizing maps. *Networks and Security*, 2002.

[12] A. Lazarevic, A. Ozgur, L. Ertoz, J. Srivastava, and V. Kumar. A comparative study of anomaly detection schemes in network intrusion detection. In *Proc. of SIAM Conf. on Data Mining*, 2003.

[13] W. Lee. A data mining framework for building intrusion detection models. In *Proc. of IEEE Symposium on Security and Privacy*, pages 120–132, May 1999.

[14] W. Lee and S. J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Trans. on Information and System Security*, 3(4):227–261, November 2000.

[15] M. Otey, A. Ghoting, and S. Parthasarathy. Fast distributed outlier detection in mixed attribute data sets. *Data Mining and Knowledge Discovery*, 12(2-3):203–228, May 2006.

[16] N. Oza. *Online Ensemble Learning*. PhD thesis, University of California, Berkeley, 2001.

[17] N. Oza and S. Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *Proc. of ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2001.

[18] S. T. Sarasamma, Q. A. Zhu, and J. Huff. Hierarchical kohonenen net for anomaly detection in network security. *IEEE Trans. on Systems, Man, and Cybernetics(B)*, 35(2):302–312, April 2005.

[19] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. Training genetic programming on half a million patterns: An example from anomaly detection. *IEEE Trans. on Evolutionary Computation*, 9(3):225–239, June 2005.

[20] K. Tieu and R. Viola. Boosting image retrieval. *International Journal of Computer Vision*, 56(1-2):17–36, 2004.

[21] R. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[22] S. Zanero and S. M. Savaresi. Unsupervised learning techniques for an intrusion detection system. In *Proc. of ACM Symposium on Applied Computing*, 2004.