Semi-supervised Learning for False Alarm Reduction

Chien-Yi Chiu[1], Yuh-Jye Lee[1], Chien-Chung Chang[1], Wen-Yang Luo[2], and Hsiu-Chuan Huang[2]

[1] Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology
Taipei, 10607 Taiwan

[2] Information & Communication Security Lab,
Chunghwa Telecom Laboratories

Abstract. Intrusion Detection Systems (IDSs) which have been deployed in computer networks to detect a wide variety of attacks are suffering how to manage of a large number of triggered alerts. Thus, reducing false alarms efficiently has become the most important issue in IDS. In this paper, we introduce the semi-supervised learning mechanism to build an alert filter, which will reduce up to 85% false alarms and still keep a high detection rate. In our semi-supervised learning approach, we only need a very small amount of label information. This will save a huge security officer's effort and make the alert filter be more practical for the real systems. Numerical comparison with conventional supervised learning approach with the same small portion labeled data, our method has significantly superior detection rate as well as in the false alarm reduction rate.

Keywords: Machine Learning, Semi-Supervised Learning, False Alarm Reduction, Intrusion Detection.

1 Introduction

In recent years, the rapidly increasing rate of cyber attacks make intrusion detection become a critical issue of network security. By the growth of the Internet and the large amount of network users, network traffic is horribly increasing. This phenomenon leads a result that alarms of intrusion detection system (IDS) become overwhelming for the analysts. Here we introduce a method of using the information of network connections to reduce false alarms. As we know, more and more network applications rely on the TCP protocol, especially the services over the World-Wide-Web (or say, over the http protocol). More and more users shop over Internet, such as booking a ticket or ordering dishes. The great volume of transactions lead the network criminals change their target from end users to popular web servers. Attackers try to embed malicious scripts or malwares into the web server to indirectly attack the great amount of web users. In this case, all the attacks over the http protocol are based on TCP protocol. By comparing with UDP and ICMP, TCP is a connection-oriented protocol. It has some additional information of the connections, such as the connection duration, bytes sent by source host, bytes sent by destination host, and so on. All of the information is used in intrusion detection [9] and holding a KDD'99 cup competition [7].

Different from general purpose IDSs which are using signatures to detect malicious information in packet payload, we use connections to analyze an alarm is suspicious or not. First, we use connection information to aggregate alerts together. After aggregating the alerts, we could directly classify a connection as suspicious one or not. By the classification results of the connections, we could determine whether the aggregated alerts are malicious or not.

In the previous work, analyst has to collect enough labeled data for the machine training. As we know, labeled data is expansive and hard to collect. In our experience, the alerts of the IDSs are very easy to collect. In contrast, to label an alert as true attack or false alarm is hard and expensive. That leads a problem, that we never know the amount of labeled data is enough or not. For using the limited labeled data, the performance of supervised learning techniques is not good enough as it could be. We propose using semi-supervised learning technique, named after *Two-Teachers-One-Student* (2T1S) [5] to solve this problem. With the corresponding connection information of the alerts, we could use the large amount of unlabeled data with few labeled data to enhance the IDSs' performance, just as a supervised learning technique could do with enough labeled data.

1.1 Contributions

Our target is to build a system, which can reduce the great amount of false alarms by corresponding TCP connection information. Moreover, for improving the performance, we use a semi-supervised learning technique 2T1S to gain more useful information from the large amount of unlabeled data.

For achieving the goals, we built a experimental system to test our ideas. Along the experimental results, we conclude the following contributions:

- Successful reducing false alarms with connection information
- Using semi-supervised learning technique 2T1S to improve the performance while only a few labeled data are available.

The remainder of the paper is organized as follows. In Section 2, we review related works, including alert classification and machine learning based intrusion detection system. In Section 3, we introduce the framework of our method. Section 4 describes the numerical experiments and details the results. Section 5 contains some concluding remarks.

2 Related Work

Machine learning techniques used on reducing false positives is not new. Tadeusz Pietraszek [15] using adaptive alert classification for supporting human analyst by classifying alerts into true positives and false positives. He addresses the false alarm problem by building a classifier, which is so-called Alert Classifier that tells true from false alarms. This kind of method is known as *Alert classification.* Another way to apply machine learning techniques on reduce false alarms is named after *Alert Sequence Classification.* For example, Alharby [2] proposed a method to characterize a “normal” stream of alarms. He developed an algorithm for detecting anomalies with continuous and discontinuous sequential patterns.

Except using classification methods to reduce false alarms, machine learning techniques is also used for building a system to detect network attacks. This kind of works are rich and widespread, dating back to at least 1980 with Anderson’s [3] initial proposal for such system. Lee [9] developed a methodology to construct additional features using data mining. He used the additional features to classify if a connection is malicious or not. In recent years, semi-supervised learning is brought in this category. Lane [8] proposed a model to fuse misuse detection with anomaly detection and to exploit strengths of both. Chen et al. [6] proposed two semi-supervised classification methods, Spectral Graph Transducer and Gaussian Fields Approach, to detect unknown attacks. They also developed a semi-supervised clustering method, MPCK-means to improve the performances of the traditional purely unsupervised clustering methods. Mao et al. [12] proposed a co-training method framework for intrusion detection, which is a semi-supervised learning method to utilize unlabeled data and combine multi-view data.

To the best of our knowledge, semi-supervised learning technique has not previously used in building classifier to reducing the false alarms. However, some concepts we apply here have been successfully used in intrusion detection and related domains. We adapt Lee’s method to construct statistical features of connections, and use these features to judge corresponding alerts are suspicious or not. We also adapt the algorithm, *Two-Teachers-One-Student* (2T1S), proposed by Chang et al. [5] for improving the performance with the unlabeled data.

3 Methodology

In this section, we first describe our motivation below. We construct the statistical connection features for reducing the false alarms, and try to use semi-supervised learning technique to improve the performance by utilizing the unlabeled data. For constructing the statistical features, we have a NCF instance extractor to extract connections from network traffic and compute statistical features with a two seconds time window. We also adapt a semi-supervised learning algorithm, 2T1S, in our machine learning based analysis engine to improve the performance via including the information of unlabeled data.

Fig. 1. illustrates the framework of our proposed system. Except the intrusion detection system as a sensor, our system has an additional sensor to create the TCP connection database, and mapping the alerts generated by the IDS with the connection records to compute the statistical features. All the alerts mapped to the same connection will be aggregate as a small cluster, and share the same

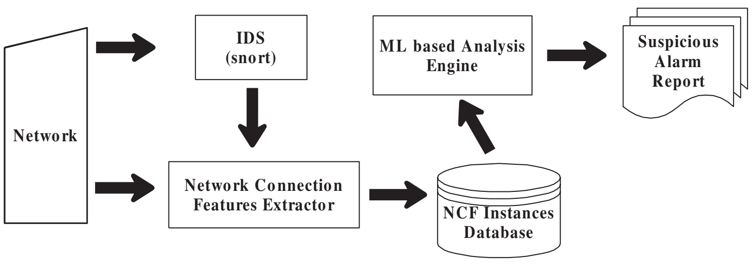

Fig. 1. The System Architecture

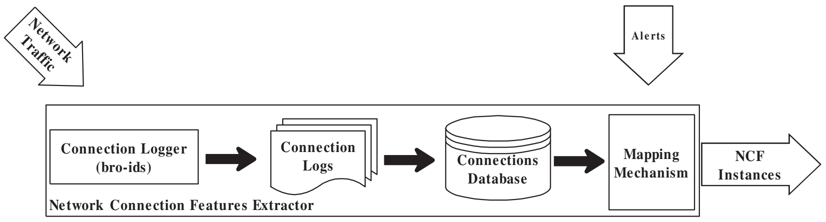

Fig. 2. The Network Connection Features Extractor

connection features. The mapped connection with the corresponding statistical features will be look as a Network Connection Features instance (NCF instance), which is stored in NCF Instance Database. Our machine learning based analysis engine will directly read the NCF instances as input data for analyzing corresponding alerts are true attacks or false alarms. Below, we describe the detail of our system framework.

3.1 IDS(Snort)

We use snort [16] as our IDS sensor due to its popularity and open source. Snort is a libpcap-based packet sniffer/ packet logger/ signature-based IDS developed by Marty Roesch. Snort was originally intended to be a packet sniffer, Roesch added the signature-based analysis (also known as rule-based analysis within the snort community) to be a rules-matching IDS. As time progressed, the size of the latest rules is increasing with the number of exploits available. That is also the reason why snort will generate large amount of false alarms.

3.2 Network Connection Features Extractor

The Network Connection Features Extractor (NCF Extractor) is composed by a TCP connection logger and a mapping mechanism. The architecture is shown in Fig. 2.

Most of the signature-based IDSs use the packet-signature to generate alerts. It means a packet is malicious or not just depends on the packet's header and payload information. However, for the network security managers, the information from only one alert is not helpful for determining whether it is a true alarm or not. The managers need more information of the hosts, including the source host information and the target host information, which triggers the alert. All of the required information for labeling alerts as attack or not is just like the service type, which the hosts provide for, or the alerts triggered by the same hosts. For the purpose, we think the statistical TCP information could make up the needed information.

For generating the connection statistical features, the TCP connection information is needful. We set a TCP connection logger to extract the connections from network traffic, and output all the connections by a connection log. We use Bro [14] as our TCP connection logger. Bro provides the function to log the parsed TCP connection semantics, which we used as the basic features of TCP connection. The summary of the TCP connection information [1] is introduced as follows:

- start: the connection's start time.
- duration: the connection's duration.
- local IP & remote IP: local and remote addresses that participated in the connection.
- service: connection's service, as defined by service.
- local port & remote port: the ports used by the connection.
- org bytes sent & res bytes sent: the number of bytes sent by the originator and responder, respectively. These correspond to the size fields of the corresponding endpoint records.
- state: the state of the connection at the time the summary was written (which is usually either when the connection terminated, or when Bro terminated).
- flags: a set of additional binary state associated with the connection.
- tag: reference tag to log lines containing additional information associated with the connection in other log files(e.g.: http.log).

We provide another viewpoint to judge the alerts are malicious or not. Instead of reading the alarms' information directly, we aggregate alerts by the TCP connections. The alerts belong to the same connection; we set they have the same characteristics on the connection viewpoint. If a connection contains at least one malicious alarm, we set this connection as a malicious one. The alerts belong to the malicious connections; we look them all as suspicious alarms.

When Bro extracts the TCP connections information into logs, we will restore the connection records into a connections database. With the incoming alerts, we use the TCP pairs (Source IP, Destination IP, Source Port and Destination Port) and the trigger time to find if a matching connection exists. If the connection exists, we will compute the statistical features with a two seconds time window. The statistical features will be combined with the basic connection information to generate a NCF instance. We also store the mapping relationship between alerts and connections into database for constructing the alert clusters. After all, the NCF instance will be send to the NCF Instance Database, each NCF instance represent the alert cluster, which mapped to corresponding connection.

3.3 Machine Learning Based Analysis Engine

Machine Learning Based Analysis Engine is used for learning and predicting the NCF instances. In the following statements, we describe the learner, which we used in the machine learning based analysis engine. Beyond what the supervised learning can offer, many real applications need to deal with both labeled and unlabeled data simultaneously. Usually, the amount of labeled data is insufficient and obtaining it is expensive. In contrast, unlabeled data is abundant and easy to collect. For example, we may need to categorize a number of web documents, but only a few of them may be correctly labeled. In another example, determining the functions of biological strings is expensive, and only a small portion of them have been studied (labeled) to date. Semi-supervised learning can help researchers deal with these kinds of problems because it takes advantage of knowing two kinds of data; 1) it uses labeled data to identify the decision boundary between data with different labels; and 2) it uses unlabeled data to determine the data's density, i.e., the data *metric*.

Among the various semi-supervised learning algorithms that have been proposed, the multi-view approach is one of the most widely used. This kind of methods split data attributes into several attribute subsets, called *views*, to improve the learning performance. In the *co-training* algorithm [4], classifiers of different views learn about the decision boundaries from each other. Based on this concept, a number of variants have been developed, e.g., the *tri-training* algorithm [17]. On the other hand, the classifiers of different views can be combined to form an *ensemble* classifier with a high level of confidence. We call this approach *consensus training*.

In this paper, we use a semi-supervised algorithm, Two-Teachers-One-Student (2T1S) [5], as the learner of our machine learning based analysis engine. 2T1S is a multi-view algorithm. Different from regular multi-view methods, 2T1S selects different views in the feature space rather than in the input space. 2T1S elegantly blends the concepts of co-training and consensus training. Through co-training, the classifier generated by one view can "teach" other classifiers constructed from other views to learn, and vice versa; and by consensus training, predictions from more than one view can give us higher confidence for labeling unlabeled data. In practice, given three different views, 2T1S choose two views as teachers for consensus training and the remaining view as the co-training partner. The classification answers from two classifiers (two teachers) represent the consensus result, which is used to teach the third view (the student) to learn the labels for unlabeled data. This process is performed for each choice of teachers-student combination. After the student learns the data, the newly learned labeled data is added to the student's original labeled data set, as the set of guessed labeled data can be included for training in the next step if it is part of the teachers' sets in the next step. The whole process is run iteratively and alternately until some stopping criteria are satisfied. We describe the 2T1S algorithm with the pseudo code in Algorithm 1.

Algorithm 1. The *2T1S* Algorithm

```
Input:
  Initial labeled data $\mathcal{D}_L=\{(\mathbf{x}^i, y_i)\}_{i=1}^{\ell}$, $\mathbf{x}^i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$.
  Initial unlabeled data $\mathcal{D}_U=\{(\mathbf{x}^i)\}_{i=\ell+1}^{m=\ell+u}$, $\mathbf{x}^i \in \mathbb{R}^n$.
  Initial classifiers $f_1(\mathbf{x})$, $f_2(\mathbf{x})$, $f_3(\mathbf{x})$.
  Initial consensus level $0 \leq \varepsilon \leq 1$.
Output:
  The final discriminant model $f(\mathbf{x})$.
$\mathcal{D}_{L_i} \leftarrow \mathcal{D}_L, i = 1, \ldots, 3.$;
$iter \leftarrow 1.$;
$\mathcal{D}_L^{(0)} \leftarrow \mathcal{D}_L.$;
repeat
    for $i \leftarrow 1$ to 3 do
        for $j \leftarrow 1$ to $u$ do
            $t_1 \leftarrow mod(i-1, 3) + 1$;
            $t_2 \leftarrow mod(i, 3) + 1$;
            $s \leftarrow mod(i+1, 3) + 1$;
            if $(f_{t_1}(\mathbf{x}^j) \geq \varepsilon$ and $f_{t_2}(\mathbf{x}^j) \geq \varepsilon)$ or;
              $(f_{t_1}(\mathbf{x}^j) \leq -\varepsilon$ and $f_{t_2}(\mathbf{x}^j) \leq -\varepsilon)$;
            then
                $\mathcal{D}_{L_s} \leftarrow \mathcal{D}_{L_s} \cup \mathbf{x}^j$;
                $\mathcal{D}_L \leftarrow \mathcal{D}_L \cup \mathbf{x}^j$;
                $\mathcal{D}_U \leftarrow \mathcal{D}_U \setminus \mathbf{x}^j$;
        Retrain the classifier $f_s(\mathbf{x})$ with $\mathcal{D}_{L_s}.$;
    $\mathcal{D}_L^{(iter)} \leftarrow \mathcal{D}_L.$;
    $iter \leftarrow iter + 1.$;
until $\mathcal{D}_L^{(iter)} = \mathcal{D}_L^{(iter-1)}$ ;
Construct an RSVM classifier $f(\mathbf{x})$ with the final labeled data set $\mathcal{D}_L.$;
Return $f(\mathbf{x}).$;
```

Here we need to emphasize the difference between network packets and connections. Most IDSs generate alerts by packet-signature, and the communicating packets between two hosts form a connection. For long duration connections, one connection may contain several alerts in it. If anyone of the alerts, which was contained by a connection is true attack, we label the connection as malicious. All the alerts mapped to the malicious connection will be concerned as suspicious alerts. When NCF Extractor generates NCF instances, we record the relationship between the alerts and connections. Alerts belong to the same connection will share the same features and classification result of NCF instance. In learning phase, we will use the NCF instances to learn a model or so-called a classifier. The model will be used for predicting a new incoming NCF instance in predicting phase. All the suspicious connections predicted by the model will be output in a suspicious alarm report.

4 Experiments

4.1 Dataset Description

DARPA intrusion detection evaluation dataset, which is sponsored by Defense Advanced Research Projects Agency and Air Force Research Laboratory, and managed by MIT Lincoln Laboratory since 1998. The available datasets, including DARPA1998, DARPA1999, and DARPA2000 datasets, generated in a simulated environment; however, they have some flaws identified both in simulation as well as the evaluation procedures [11][13]. We adopt the DARPA1999 dataset for experiments because it provides entire contents of attack database and attack truth files for labeling. For extracting TCP connections to analyze, we need the raw-traffic data as the input to our connection sensor. DARPA1999 dataset contains five weeks inside and outside sniffing data, fulfilling our requirement for extracting connection logs. All the dataset is separated into two parts. The first three weeks of the sniffing data are treated as training data, and the other two weeks are used as testing data.

In our experiment, we combine the inside and outside alerts and NCF instances together. We use Bro 1.4 as our connection logger and Snort 2.8.4.1 as our IDS. Table 1 summarizes the statistics of the datasets. Bro Connections stand for the connection amount extracted by Bro. Snort TCP Alerts means the alerts generated by snort and belong to TCP protocol. The false alarm rate is the false positive rate in training and testing set.

Table 1. Dataset Statistics

	Training set	Testing set
Bro Connections	1640157	1116166
Snort TCP Alerts	13912	16966
False Alarm Rate	92.33%	98.22%

4.2 Evaluation

For evaluating the experiment results, we consider two metrics to assess the performance of the learning methods. The first metric is detection rate, which is used for showing the missing rate of true attacks. The second one is the reduction rate, which stands for displaying the rate of the filtered alarm.

We use RSVM [10] as our supervised learner to test our approach of using connection information to reduce the false alarm. Before we use the dataset to learn a model, we perform three preprocessing works:

- Feature selection using information gain and gain ratio.
- Pick 50 both positive and negative training points as our reduced set to build the kernel matrix.
- Over-sampling the positive points to reform the unbalanced dataset.

Table 2. Testing Result of Supervised Learning and Semi-Supervised Learning with partial labeled data. (repeat 10 times).

	Supervised Learning		Semi-Supervised Learning	
Ratio of Labeled Data	Detection Rate	Reduction Rate	Detection Rate	Reduction Rate
1%	0.6766±0.057	0.8482±0.049	0.7046±0.072	0.7748±0.145
3%	0.7425±0.044	0.6725±0.154	0.7822±0.053	0.7814±0.113
5%	0.7508±0.055	0.6272±0.114	0.7912±0.067	0.8128±0.076
7%	0.7790±0.033	0.5829±0.012	0.8321±0.082	0.7934±0.094
10%	0.7460±0.041	0.6451±0.112	0.8607±0.046	0.8141±0.085
30%	0.8613±0.064	0.8430±0.016	0.8917±0.048	0.8527±0.041
50%	0.8417±0.070	0.8603±0.022	0.8861±0.037	0.8263±0.018
70%	0.9091±0.043	0.8401±0.026	0.8963±0.029	0.8532±0.015

We got a result of reducing 66.5% false alarms by missing 4.4% true attacks on testing data. It stands for filtering 11693 alerts and only less than 0.1% alarms belong to malicious. The result shows the connection features work well with a supervised learner, RSVM.

After we got the previous results, we began to test if the performance of supervised learning technique is affected by the size of labeled data. We random pick a small portion of the training data as our training set to build model, and test if the model could classify the testing data correctly. This process will be repeated 10 times for calculating the mean and standard variation of the results. The results are shown in Table 2.

From the results, we could easily know the performance is affected seriously by the size of the labeled data. The detection rate rise and down between 67.7% to 90%, and the reduction rate is varied from 58% to 86%. Here we need to emphasize the trade off between the detection rate and the reduction rate. When we tuning the RSVM parameters for learning a better model, we could easily find a model with great detection rate but awful reduction rate. It means the model almost classifies all the alerts as malicious. In contrast, a model with high reduction rate but very low detection rate also exists. It means the classifier tell us most of the alerts are benign. In Table 2, we choose a relatively good model in both detection rate and reduction rate. That also the reason why the detection rate does not always monotonic increase with the increase in percent of labeled data. For improving the performance, we attempt to use semi-supervised learning techniques to test if the unlabeled data will be helpful on building the model.

We choose 2T1S as our semi-supervised learner, and make some modification to let it be suitable to apply here. The modification is list as following:

– Feature selection using information gain and gain ratio.
– Over-sampling positive points before base learner training the classifier.
– Apply different parameters on each base learner.

The results of classifying the testing data using the model learned from 2T1S are also shown in Table 2. With 10% partial labeled data, supervised learning can merely reduce 64.5% alerts and detect 74.6% true attacks. At the same time, semi-supervised learning can detect 86.1% true attacks and reduce 81.4% alerts, both detection rate and reduction rate of semi-supervised learning is significantly better than supervised method. In most cases, semi-supervised learning also has better results than supervised one. These results strongly support our ideas that exploiting the information of unlabeled data could improve the performance.

5 Conclusion

In this paper, we successfully using the connection features to reduce false alarms by both supervised and semi-supervised learning techniques.

We use Network Connection Feature instance (NCF instance) to represent corresponding cluster of alerts. NCF instances will be fed to the machine learning based analysis engine to learn a model for classifying alerts into suspicious alerts or false alarms. In our experiments, we use RSVM as our supervised learning algorithm to test whether our framework works well in reducing false alarms by the NCF instances. The results show that we could filter out 65% false alarms and only miss less than 0.1% true attacks in the filtered alarms. However, having entire labeled data to build the alert filter is not practical. Thus, we introduced semi-supervised learning technique in this work. The numerical results show that both detection rate and reduction rate can be improved with very limited labeled data points. While only use small portion of labeled data in supervised learning will not have satisfied the results.

Because of the connection features used with the supervised learner and semi-supervised learner are the same, we believed that the semi-supervised learning technique could bring the improvement with the unlabeled data.

Acknowledgement

We would like to thank the anonymous referees for providing constructive comments. The work has been supported in part by Telecommunication Laboratories, Chunghwa Telecom Co., Ltd. under the grant 98AG054 and 99AG064, and by Ministry of Economic Affairs, R.O.C. under grant 98EC17A02S20137.

References

1. Bro reference manual: Analyzers and events (April 2007), `http://www.bro-ids.org/wiki/index.php/Reference_Manual:_Analyzers_and_Events`
2. Alharby, A., Imai, H.: IDS false alarm reduction using continuous and discontinuous patterns. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 192–205. Springer, Heidelberg (2005)
3. Anderson, J.P.: Computer security threat monitoring and surveillance. Technical report, Computer Security Division of the Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD (1980)

4. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: COLT: Proceedings of the Workshop on Computational Learning Theory, pp. 92–100. Morgan Kaufmann Publishers, San Francisco (1998)
5. Chang, C.-C., Pao, H.-K., Lee, Y.-J.: An RSVM based two-teachers-one-student semi-supervised learning algorithm. Pattern Recognition (under submission)
6. Chen, C., Gong, Y., Tian, Y.: Semi-supervised learning methods for network intrusion detection. In: Proceeding of IEEE International Conference on Systems, Man and Cybernetics, October 2008, pp. 2603–2608 (2008)
7. Hettich, S., Bay, S.D.: The uci kdd archive (1999), `http://kdd.ics.uci.edu/`
8. Lane, T.: A decision-theoretic, semi-supervised model for intrusion detection. In: Machine learning and data mining for computer security: Methods and applications, number 978-1-84628-029-0 (Print) 978-1-84628-253-9 (Online). Advanced Information and Knowledge Processing, pp. 157–177. Springer, Heidelberg (2006)
9. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM Transactions on Information and System Security (TISSEC) 3(4), 227–261 (2000)
10. Lee, Y.-J., Mangasarian, O.L.: RSVM: Reduced support vector machines. In: Proceedings of the First SIAM International Conference on Data Mining (2001)
11. Mahoney, M.V., Chan, P.K.: An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. LNCS, pp. 220–238. Springer, Heidelberg (2003)
12. Mao, C.H., Lee, H.M., Parikh, D., Chen, T., Huang, S.Y.: Semi-supervised co-training and active learning based approach for multi-view intrusion detection. In: Proceedings of the 2009 ACM symposium on Applied Computing, pp. 2042–2048. ACM, New York (2009)
13. McHugh, J.: Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Transactions on Information and System Security 3(4), 262–294 (2000)
14. Paxson, V.: Bro: A system for detecting network intruders in real-time. In: USENIX (ed.) Seventh USENIX Security Symposium proceedings: conference proceedings, San Antonio, Texas, January 26-29. USENIX (1998)
15. Pietraszek, T.: Using adaptive alert classification to reduce false positives in intrusion detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 102–124. Springer, Heidelberg (2004)
16. Roesch, M.: Snort - lightweight intrusion detection for networks. In: Large Installation System Administration Conference (LISA Conference), pp. 229–238 (1999)
17. Zhou, Z.-H., Li, M.: Tri-training: Exploiting unlabeled data using three classifiers. IEEE Transactions on Knowledge and Data Engineering 17(11), 1529–1541 (2005)