Network Anomalous Attack Detection Based on Clustering and Classifier

Hongyu Yang[1,2], Feng Xie[3], and Yi Lu[4]

[1] Information Technology Research Base, Civil Aviation University of China
Tianjin 300300, China
[2] Tianjin Key Lab for Advanced Signal Processing, Civil Aviation University of China
Tianjin 300300, China
yhyxlx@hotmail.com
[3] Software Division, Inst. of Computing Tech., Chinese Academy of Science
Beijing 100080, China
xiefeng@software.ict.ac.cn
[4] Security and Cryptography Laboratory, Swiss Federal Institute of Technology
(EPFL), CH-1015 Lausanne, Switzerland
yi.lu@epfl.ch

Abstract. A new approach to detect anomalous behaviors in network traffic is presented. The network connection records were mapped into different feature spaces according to their protocols and services. Then performed clustering to group training data points into clusters, from which some clusters were selected as normal and known-attack profile. For those training data excluded from the profile, we used them to build a specific classifier. The classifier has two distinct characteristics: one is that it regards each data point in the feature space with the limited influence scope, which is served as the decisive bounds of the classifier, and the other is that it has the "default" label to recognize those novel attacks. The new method was tested on the KDD Cup 1999 data. Experimental results show that it is superior to other data mining based approaches in detection performance, especially in detection of PROBE and U2R attacks.

1 Introduction

The goal of intrusion detection is to detect security violations in information systems. It is a passive approach to security as it monitors information systems and raises alarms when security violations are detected. There are generally two types of approaches taken toward network intrusion detection: *misuse detection* and *anomaly detection*.

In supervised anomaly detection, given a set of normal data to train from, the goal is to determine whether the test data belongs to normal or to an anomalous behavior. Recently, there have been several efforts in designing supervised network-based anomaly detection algorithms, such as ADAM [1]. Unlike supervised anomaly detection where the models are built only according to the normal behavior on the network, unsupervised anomaly detection attempts to

Y. Wang, Y. Cheung, and H. Liu (Eds.): CIS 2006, LNAI 4456, pp. 672–682, 2007.

detect anomalous behavior without using any knowledge about the training data. Unsupervised anomaly detection approaches are usually based on statistical approaches [2], clustering [3,4,5,6], outlier detection schemes [7,8], etc.

In this paper, we introduce a novel data mining based framework for anomaly detection, which uses clustering and classification algorithms to automatically detect the known and new attacks against computer networks and systems. We evaluated our system over KDD Cup 1999 data [9], which is a very popular and widely used intrusion attack data set. Experimental results show that our approach is really very competitive with respect to other approaches

2 System Design

Our aim is to justify the network connections normal or intrusive, which means we would reconstruct the network packets and extract features that describe the higher-level interactions between end hosts. Our scheme is divided into two phases. In training phase, we construct the normal profile and known-attack profile from the labeled training data, respectively. When detecting, our system classify the incoming connection as normal, known-attack or anomaly.

2.1 Framework

We use a combination of clustering and classification to discover attacks in a tcpdump audit trail. In our framework, the training set needs to be labeled or attack-free. If the data set includes the labeled attacks, we could get the known-attack profile. Otherwise, we only have the normal profile. When training is finished and detection model is built, we can use it to discriminate the new incoming connection on line. The purpose of clustering is to model the normal and known-attack network behaviors. We think the connections of the same type are more resemble statistically, which means these data are more easily clustered together. Therefore, we can use the centroid of a cluster to represent all members within that cluster, which will reduce the mass of raw data markedly.

For those ambiguous data in sparse space, we need a classifier to deal with them. Other than traditional classifiers, our classifier has the ability to classify a connection record as "anomaly". It is important to note that there is no "anomaly" class in training set, in which all examples belong to either "normal" class or "known-attack" class. Generally speaking, the traditional classifier only labels the data as known categories that are presented in the training set. However, we let the classifier include a "default" label by which the classifier expresses its inability to recognize the class of the connection as one of the known classes. Of course, the "default" label is "anomaly" in our algorithm. Our later experimental results will show that this is a very efficient way to detect the novel attacks that keep unseen before.

Thus, the system is ready to detect intrusions. First, the raw network packets are reconstructed to a connection and correspondingly preprocessed according to its protocol and service. Then it is compared with the profile modeled in the

training phase. If it exists in the profile, we will label it as the matched type. Otherwise, it will be fed to the classifier, which will label it as normal, known attack, or anomaly. Finally, when the number of data labeled as known-attack or anomaly surpasses a threshold, an analysis module using the association algorithms will deal with the vast data in order to extract the frequent episodes and rules.

2.2 Feature Space and Attributes Deals

Feature Spaces

We map the connection records from the audit stream to a feature space. The feature space is a vector space of high dimension. Thus, a connection is transformed into a feature vector. We adopt 8 feature spaces according to the protocol and service of the connection. That is, we choose the different attributes for the connections with different services. An important reason is that different services usually have the specific security-related features. For example, the attributes of an HTTP connection are different from those of an FTP connection. The eight services include HTTP, FTP, SMTP, TELNET, FINGER, UDP, ICMP and OTHER, in which OTHER service is default. So, even if a new service occurs in the data stream for the first time, it can be simply regarded as OTHER service without reconfiguring the system.

Distance Function

In order to describe the similarity of two feature vectors, we use the Euclidean distance as our measure function:

$$d(v_i, v_j) = \sqrt{\sum_{k=1}^{\|v_i\|} (v_i^{(k)} - v_j^{(k)})^2} \tag{1}$$

where both v_i and v_j are feature vectors in the vector space $\Re^n$ of the same dimensions. $v_i^{(k)}$ represents the kth component of vector v_i, and $\|v_i\|$ means the dimensions of v_i, i.e. n.

Apparently, the distance between the two vectors is in reverse proportion to the similarity between them. For simplicity, we suppose each component of a vector is the same weight.

Discrete and Continuous Attributes

There are two attribute types in our connection records. One is discrete, i.e. nominal, and the other is continuous. Since the number of normal instances usually vastly outnumbers the number of anomalies in training data set and in anomaly detection values that are observed more frequently are less likely to be anomalous, we represent a discrete value by its frequency. As a result, discrete attributes are transformed to continuous ones. For a continuous attribute, we adopt the "cosine" normalization to quantize the values. Furthermore, the values of each attribute are normalized to the range [0,1] to avoid potential scale problem. The whole normalization processes include two steps: the first step is normalization of each continuous attribute,

$$v_i^{(k)'} = \frac{v_i^{(k)}}{\sqrt{\sum_{j=1}^{\|D\|} (v_j^{2(k)})}} \tag{2}$$

where $\|D\|$ represents the total number of vectors in the training set D. And the second step is the normalization of the feature vector. Note that we don't regard those transformed from discrete attributes in this step.

$$v_i^{(k)'} = \frac{v_i^{(k)}}{\sqrt{\sum_{k=1}^{\|v_i\|} (v_i^{2(k)})}} \tag{3}$$

2.3 Clustering and Profile Selection

At present, we use standard k-means algorithm [10] as our clustering approach. K-means is a centroid-based clustering with low time complexity and fast convergence, which is very important in intrusion detection due to the large size of the network traffic audit dataset.

Each cluster in profile can be simply expressed as a centroid and an effect influence radius. So a profile record can be represented as the following format:

$$\langle\ \boldsymbol{centroid}, \boldsymbol{radius}, \boldsymbol{type}\ \rangle$$

Centroid is a centric vector of the cluster, *radius* refers to influence range of a data point (represented as the Euclidean distance from the centroid), and *type* refers to the cluster's category, e.g. normal or attack. We can determine whether a vector is in the cluster or not only by computing the distance between the vector and the centroid and comparing the distance with the radius. If the distance is less than radius, we consider that the vector belongs to the cluster. And then we can label the vector as the cluster's type. Therefore, the whole search in the profile only includes several simple distance calculations, which means we can deal with the data rapidly.

Of course, not all clusters can serve as the profile. Some maybe include both normal and attack examples and not fit for the profile apparently. It is necessary to select some clusters according to a strategy. At present, we use the following conditions as our selection criterion.

Condition 1: *The number of examples in the cluster as the profile must surpasses a threshold.*

Condition 2: *The purity of the cluster as the profile must surpasses a threshold.*

Condition 3: *The density of the cluster as the profile must surpasses a threshold.*

Cond.1 emphasizes the confidence of the cluster as the profile. We think one cluster with more examples often means more stable and more representative. On the contrary, a little cluster, e.g. there are only 5 examples in it, is distinctly not fit for a profile.

In ***cond.2***, the purity of a cluster refers to percentage of majority examples in the cluster. Formally, it can be represented as follows:

$$Purity(X) = \frac{Number\ of\ Majority\ Examples}{Total\ Number\ of\ Examples\ in\ Cluster\ X}$$

A majority example is an example that belongs to the most frequent class in the cluster. The higher the purity is, the better the cluster is served as a profile. A cluster with small purity means that there are many attacks with different types in the cluster, so we don't select such cluster as our profile. Instead, we use them as the training set for classifier.

Cond.3 is less important than the first two conditions. Usually, most clusters meet this condition by nature. Here, we just use it to prevent those sparse clusters. For a cluster with low density, it is possible that some novel attacks will lie in it. So we think the sparse cluster is not fit for the profile.

After the clusters are selected for the profile, we put them into the profile repository. The basic contents include *centroid*, *radius* and *type*. Here, we use the type of majority examples in one cluster as the whole cluster's type regardless of the minority examples.

Parameters Determination

There are 4 parameters determining the profile selection: the number of clusters K, the *size*, *purity* and *density* of cluster. It is rather difficult to decide how to set these values to let the system be the best. But according to the experimental results, we found even if these parameters are set simply, the system can achieve a good performance. Intuitively, we hope the *size* is equal to the average size of cluster, i.e. the parameter *size* = the total number of samples in training set/the parameter K. In contrast to K, the parameter *size* is meaningful and set more easily. The larger size means the cluster is more stable and, unfortunately, the number of cluster satisfying the condition is less. Therefore, the value is set to 200 in our experiment. Accordingly, the parameter K is also determined. The parameter *purity* is very easy to set. This value decides the quality of cluster directly. If it is too little, many mixed clusters will be served as profile which will reduce the final detection. In the following experiment, we fixed it as 0.98. Finally, for simplicity, the parameter *density* is defined as the scale of the number of samples in the cluster to the radius of such cluster.

2.4 Influence-Based Classifier

There are many classification algorithms, such as Naive Bayes and decision tree, but they all don't support the "default" label in itself. Therefore, we present a new algorithm to address this problem, which is called influence-based classification algorithm in which we introduce the concept of data field and influence. We view the whole feature space as a large data field, in which every object interacts with each other. We use a function to quantify the influence of an object, which is called influence function. We adopt the Gaussian function to measure

it. Denote the N-dimension feature space by $\Re^n$. So, the influence function can be represented as follows:

$$f_y(x) = \phi(x, y) = e^{-\frac{d^2(x,y)}{2\sigma^2}} \tag{4}$$

where $x, y \in \Re^n$, $f_y(x)$ means the influence function of a data object y. $d^2(x, y)$ is the square of the distance between x and y, while σ is called influence factor determining the influence scope of y.

The influence function of a dataset $D \subset \Re^n$ is defined as the sum of the influence functions of all data objects in D.

$$f_D(x) = \sum_{y \in D} f_y(x) = \sum_{y \in D} e^{-\frac{d^2(x,y)}{2\sigma^2}} \tag{5}$$

As we know, for a Gaussian distribution, rough 99.7% of the values fall within 3σ margin, which is the famous "3σ criterion". That is, the influence scope of a data object is rough equal to 3σ. So, in our algorithm, we only focus on those objects inside this range and ignore others. The whole algorithm is illustrated in Fig. 1.

Input: a sample P to be labeled, the influence factor σ, and the training set D
Output: Label P as normal, known-attack or anomaly
Begin
1. normalize P;
2. $f_+ \leftarrow 0$, $f_- \leftarrow 0$;
3. **for each** sample Q in D
4. **if** $d(P, Q) > 3\sigma$ **continue**;
5. compute the influence at P generated by Q and add it to f_+ **if** Q is normal, otherwise add it to f_-;
endfor
6. **if** $f_+/(f_- + f_+) > T_N$ label P as normal;
7. **else if** $f_-/(f_- + f_+) > T_A$ label P as known-attack;
8. **else** label P as anomaly.
End.

Fig. 1. Influence-based Classification Algorithm

3 Experiment and Result

In the experimentg, we handled 10% of the whole KDD'99 dataset [9] corresponding to 494019 training connections and 311029 testing connections. Fig. 2 shows the results of our experiments, in which there are 5 ROC curves, 4 curves corresponding to 4 categories of attacks respectively, i.e. PROBE, DOS, U2R and R2L, and the left one corresponding to the overall attacks. "PROBE (4166)" denotes there are 4166 probing examples in the test set. Also, "OVERALL (250436/60593)" means there

are total 250436 attacks and 60593 normal examples in the test set, and the corresponding curve describes the overall detection performance of our system. Furthermore, we list the more detailed results including each attack name, category, total number in the testing set and corresponding detection rate at the false alarm rate of 0.7% (stated in Table 1).

Table 1. The detection performance of all attacks in the test set. "*" means the attack type is novel, i.e. it doesn't occur in the training set. Note that the false alarm rate is 0.7%, TOTAL means the total number of attacks with the same category in the test set and TDR denotes true detection rate.

ATTACK NAME	TOTAL (TDR)	ATTACK NAME	TOTAL (TDR)	ATTACK NAME	TOTAL (TDR)	ATTACK NAME	TOTAL (TDR)
portsweep (PROBE)	*354 (99.72%)*	smurf (DOS)	*164091 (100%)*	udpstorm* (DOS)	*2 (100%)*	processtable* (DOS)	*759 (94.20%)*
satan (PROBE)	*1633 (99.88%)*	pod (DOS)	*87 (98.85%)*	xterm* (U2R)	*13 (84.62%)*	snmpgetattack* (R2L)	*7741 (0%)*
nmap (PROBE)	*84 (100%)*	neptune (DOS)	*58001 (99.97%)*	apache2* (DOS)	*794 (58.94%)*	snmpguess* (R2L)	*2406 (0.04%)*
ipsweep (PROBE)	*306 (99.02%)*	land (DOS)	*9 (100%)*	mailbomb* (DOS)	*5000 (12.20%)*	guess_passwd (R2L)	*4367 (14.88%)*
saint* (PROBE)	*736 (99.05%)*	teardrop (DOS)	*12 (83.33%)*	Perl (U2R)	*2 (100%)*	buffer_overflow (U2R)	*22 (95.45%)*
mscan* (PROBE)	*1053 (99.24%)*	back (DOS)	*1098 (99.36%)*	phf (R2L)	*2 (50%)*	loadmodule (U2R)	*2 (100%)*
rootkit (U2R)	*13 (23.08%)*	ps* (U2R)	*16 (68.75%)*	xlock* (R2L)	*9 (44.44%)*	warezmaster (R2L)	*1602 (63.05%)*
sendmail* (R2L)	*17 (17.65%)*	ftp_write (R2L)	*3 (66.67%)*	multihop (R2L)	*18 (61.11%)*	httptunnel* (U2R)	*158 (84.18%)*
xsnoop* (R2L)	*4 (50%)*	named* (R2L)	*17 (35.29%)*	worm* (R2L)	*2 (0%)*	sqlattack* (U2R)	*2 (100%)*
imap (R2L)	*1 (100%)*	-	-	-	-	-	-

It is shown that the performance of detection of PROBE and DOS attacks of the system is superior to that of other attacks, especially detection of R2L attacks. We analyzed the results in detail and found the reason for the low detection rate for R2L attacks. Both PROBE and DOS attacks often have the distinct traffic characteristic while U2R and R2L are more similar to normal examples. Especially, two R2L attack types (*snmpgetattack* and *snmpguess*) are hardly detected, which account up rough 63% of all R2L attacks. In fact, they are almost identical with normal examples and hardly detected only by the connection information. This means the detection rate for R2L attacks would reach 37% at most no matter what the false alarm rate is. Therefore, in Fig. 2, the detection rate for R2L attacks keeps stable (about 36.6%) when false positive rate surpasses 2.8%. Excluding these two types, our system can detect other attacks with the interesting detection and false alarm rates.

Fig. 3 shows the discrimination of the test data graphically, in which X axis denotes the number of testing samples with different categories, while Y axis denotes the ratio of the influence at a testing point produced by the normal samples to those produced by all samples, i.e. $f_+/(f_+ + f_-)$. For simplicity, we call the ratio value as the positive influence ratio. If the influence at a point

in the data field is zero, we let the value be 0.5. Considering the mass of DOS attacks, we only use a little part of them, but keep all other attacks. Note that the value *cutoff 1* and *cutoff 2* are all thresholds, respectively corresponding to $(1-T_A)$ and T_N in Fig. 1. In the experiment, we found that they were insensitive, which means they are easy to set and don't affect the final results too much. Meanwhile, we found the obtained values mostly focused on 0, 1 and 0.5. That is, these samples could be discriminated easily. For example, there are rough 99.2% of total 60593 normal samples, of which the positive influence ratio are equal to 1. We, however, also can see that a few attacks are mislabeled, in which most are *snmpgetattack* and *snmpguess* (they are labeled in the figure too).

Fig. 4 shows the average positive influence ratio of all samples in this test set. Clearly, the average ratio of normal samples is distinct from that of intrusions excluding *snmp* attacks. Note that the values of novel attacks are mostly approximate to 0.5 according to our algorithm.

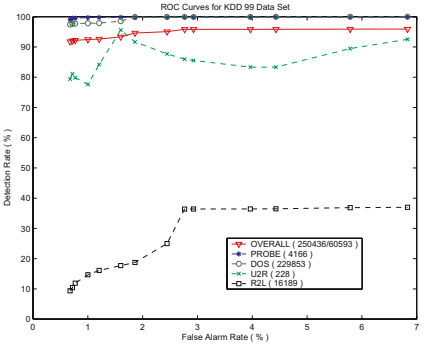

Fig. 2. The performance of proposed system. The curves are obtained by varying the influence factor σ.

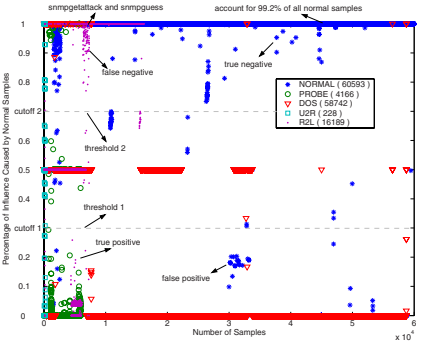

Fig. 3. The distribution of positive influence ratio of all samples in testing set. We omit a lot of DOS attacks. cutoff 1 and cutoff 2 are thresholds deciding the class of data.

Furthermore, we have compared our approach with other proposed methods, of which some participated in the task of KDD Cup. Since KDD Cup is concerned with multi-class classification but we are only interested in knowing whether the record is normal or anomalous, we have converted the results of those methods into our format. Specifically, the detection rate measures the percentage of intrusive connections in the test set that are labeled as known-attack or anomaly,

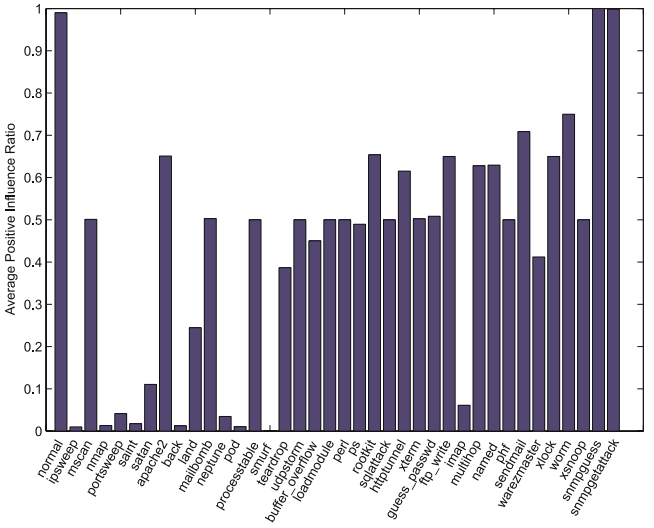

Fig. 4. The average positive influence ratio of all samples in test set in KDD Cup data

Table 2. Comparison of our system with other approaches

METHOD	FAR	PROBE	DOS	U2R	R2L
Our approach	0.7%	**99.5%**	**97.92%**	**81.14%**	10.44%
C5 Bagged Boosting	0.55%	87.73%	97.7%	26.32%	10.27%
Kernel Miner	0.55%	89%	97.57%	22.37%	7.38%
NN	**0.45%**	83.3%	97.3%	8.33%	2.5%
Decision Tree	0.5%	77.92%	97.24%	13.6%	0.52%
Naive Bayes	2.32%	88.33%	96.65%	11.84%	8.66%
PNrule	0.5%	78.67%	97%	14.47%	**10.8%**

without considering whether they are classified into the correct intrusion categories. The results are shown in Table 2, in which FAR means false alarm rate. The best results are highlighted by bold face. It can be seen that our system outperforms others significantly, especially in detection of PROBE and U2R attacks, while false alarm rate is comparable to the other approaches.

Table 3. The example distribution of 3 subsets in 3-fold cross validation experiments

Subsets	NORMAL	PROBE	DOS	U2R	R2L
A	52602	2940	204790	16	4755
B	52599	2987	207168	146	5213
C	52670	2346	209344	118	7347

Table 4. The grouping in 3-fold cross validation experiment

-	Training Set	Test Set
Group 1	A + B	C
Group 2	A + C	B
Group 3	B + C	A

Table 5. Results of 3-fold cross validation. We lists the detection performance at 5 different levels of false alarm rate, and P, D, U and R refers to detection rate of PROBE, DOS, U2R and R2L, respectively.

FAR	0.005				0.007				0.01				0.015				0.025			
TDR	P	D	U	R	P	D	U	R	P	D	U	R	P	D	U	R	P	D	U	R
Group 1	*.81*	*.99*	*.57*	*.51*	*.87*	*.99*	*.75*	*.52*	*.88*	*.99*	*.83*	*.53*	*.89*	*.99*	*.96*	*.53*	*.89*	*.99*	*.96*	*.53*
Group 2	*.84*	*.97*	*.72*	*.41*	*.89*	*.99*	*.77*	*.50*	*.95*	*.99*	*.90*	*.52*	*.97*	*.99*	*.98*	*.54*	*.97*	*.99*	*.98*	*.54*
Group 3	*.93*	*.99*	*.82*	*.45*	*.97*	*.99*	*1.0*	*.54*	*.98*	*.99*	*1.0*	*.55*	*.98*	*.99*	*1.0*	*.55*	*.98*	*.99*	*1.0*	*.55*

In addition to regular evaluations above, we have performed the 3-fold cross validation, i.e. we incorporated the original training and testing sets into one set, and randomly split it into 3 subsets of approximately equal size. Afterwards, we trained the model 3 times, each time leaving out one of the subsets from training, but using only the omitted subset to compute detection rate and false alarm rate. In these subsets, we let some attacks only occur in one subset intentionally in order that these attacks could be regarded as novel attacks when the subset was used as test set. The sample distribution of 3 subsets and experiment grouping are shown in Table 3 and Table 4, respectively. and the experimental results are shown in Table 5.

4 Conclusion

Indeed, the proposed framework is a supervised system including the benefit of clustering and classification. Compared with another famous supervised system ADAM which use frequent episodes to build the normal profile, we adopt clusters as system profile. We deem that this method characterizes the network behaviors better and more precise. In addition, we can get not only normal profile but also known-attack profile if the training data set includes the attack samples. As far as detection performance is concerned, our system can find many categories attacks while ADAM is devised to detect only PROBE and DOS attacks.

We adopt a influence-based classification algorithm to perform the final detection. Specifically, we view the whole feature space as data field, in which each point has a limited influence on others. So, we use the influence to discriminate the data. The experimental results show the approach is effective.

Acknowledgement

This work was supported in part by grants from the Major Project of High-Tech Research and Development Program of China (20060112A1037), Natural Science Foundation of Tianjin (06YFJMJ00700), the Research Foundation of CAUC (05YK12M) and the Open Foundation of Tianjin Key Lab for Advanced Signal Processing. We would like to thank those organizations and people for their supports.

References

1. Barbara, D., Couto, J., Jajodia, S., Wu, N.: ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. SIGMOD Record (2001)
2. Ye, N., Chen, Q.: An Anomaly Detection Technique Based on a Chi-Square Statistic for Detecting Intrusions into Information Systems. Quality and Reliability Engineering International 17(2), 105–112 (2001)
3. Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.J.: A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. In: Proc. Of Application of Data Mining in Computer Security, Kluwer, Dordrecht (2002)
4. Leung, K., Leckie, C.: Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. In: Proc. of 28th Australasian Computer Science Conference (ACSC), Newcastle, Australia, pp. 333–342 (2005)
5. Oldmeandow, J., Ravinutala, S., Leckie, C.: Adaptive Clustering for Network Intrusion Detection. In: Proc. of the 3th International Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) (2004)
6. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion Detection with Unlabeled Data Using Clustering. In: Proc. of ACM CSS Workshop on Data Mining Applied to Security (2001)
7. Ertoz, L., Eilertson, E., Lazarevic, A.: The MINDS - Minnesota Intrusion Detection System. In: Proc. Of Workshop on Next Generation Data Mining (2004)
8. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient Algorithms for Mining Outliers from Large Data Sets. In: Proc. Of the ACM SIGMOD Conference (2000)
9. KDD Cup 1999 Data (2006), http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
10. MacQueen: Some Methods for Classification and Analysis of Multivariate Observations. In: Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (2001)