A New Unsupervised Anomaly Detection Framework for Detecting Network Attacks in Real-Time

Wei Lu and Issa Traore

Department of Electrical and Computer Engineering, University of Victoria,
PO Box 3055 STN CSC, Victoria, B.C., Canada
{wlu, itraore}@ece.uvic.ca

Abstract. In this paper, we propose a new unsupervised anomaly detection framework for detecting network intrusions online. The framework consists of new anomalousness metrics named IP Weight and an outlier detection algorithm based on Gaussian mixture model (GMM). IP Weights convert the features of IP packets into a four-dimensional numerical feature space, in which the outlier detection takes place. Intrusion decisions are made based on the outcome of outlier detections. Two sets of experiments are conducted to evaluate our framework. In the first experiment, we conduct an offline evaluation based on the 1998 DARPA intrusion detection dataset, which detects 16 types of attacks out of a total of 19 network attack types. In the second experiment, an online evaluation is performed in a live networking environment. The evaluation result not only confirms the detection effectiveness with DARPA dataset, but also shows a strong runtime efficiency, with response times falling within seconds.

1 Introduction

Intrusion detection has been extensively studied since the seminal report written by Anderson [1]. Traditionally, intrusion detection techniques are classified into two categories: misuse detection and anomaly detection. Misuse detection is based on the assumption that most attacks leave a set of signatures in the stream of network packets or in audit trails, and thus attacks are detectable if these signatures can be identified by analyzing the audit trails or network traffic behaviors. However, misuse detection approaches are strictly limited to the latest known attacks. How to detect new attacks or variants of known attacks is one of the biggest challenges faced by misuse detection.

To address the weakness of misuse detection, the concept of anomaly detection was formalized in the seminal report of Denning [4]. Denning assumed that security violations could be detected by inspecting abnormal system usage patterns from the audit data. As a result, most anomaly detection techniques attempt to establish normal activity profiles by computing various metrics, and an intrusion is detected when the actual system behavior deviates from the normal profiles. According to Axelsson, "the early anomaly detection systems were

Y.G. Desmedt et al. (Eds.): CANS 2005, LNCS 3810, pp. 96–109, 2005.

self-learning, that is, they automatically formed an opinion of what the subject's normal behavior was" [2]. Self-learning techniques combine the early statistical model based anomaly detection approaches [10][12][17], and the AI based approaches [8] or the biological models based approaches [9], and thus they are still applied for current anomaly detection schemes. According to whether they are based on supervised or unsupervised learning techniques, anomaly detection schemes can be classified into two categories: unsupervised anomaly detection and supervised anomaly detection [14].

Supervised anomaly detection establishes the normal profiles of systems or networks through training based on labeled datasets. In contrast, unsupervised anomaly detection attempts to detect intrusions without using any prior knowledge of attacks or normal instances. The main drawback of supervised anomaly detection is the need of labeling the training data, which makes the process error-prone, costly and time consuming. Unsupervised anomaly detection addresses these issues by allowing training based on unlabelled datasets and thus facilitating online learning and improving detection accuracy.

Clustering algorithm is one of the most widely used unsupervised learning techniques. Some examples of using clustering algorithms for intrusion detection were suggested in literature [5], [6] and [14]. Although clustering techniques have showed their capability for intrusion detection, labeling clusters is still a difficult problem faced by this kind of approach. In order to label the clusters, the approach usually makes two assumptions: (1) data instances always belong to two categories: normal clusters and intrusive clusters; (2) the number of normal data instances largely outnumbers the number of intrusions. However, these assumptions are not always the case in practice. The number of clusters is not supposed to be determined in advance. When data instances include only normal behavioral data, the assumptions will lead a high false alert rate. In order to obtain an efficient and effective detection, we propose in this paper a new unsupervised anomaly detection framework based on outlier detection techniques. The proposed detection scheme consists of a feature extraction technique based on new anomalousness metrics, named IP Weight and an outlier detection algorithm based on Gaussian mixture model (GMM).

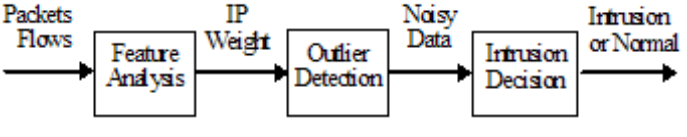


Fig. 1. General architecture

Fig. 1 illustrates the general architecture of our framework, which consists of three components, namely feature analysis, outlier detection and intrusion decision. During feature analysis, IP Weights are generated from standard IP packet flows. This allows extracting salient and useful domain knowledge and reducing significantly the dimensionality of the feature space. Then, noisy data

of IP Weights are detected in outlier detection phase. Intrusion decision is made based on the outcome of outlier detections.

We discuss each of these phases in the rest of the paper. Specifically, Section 2 outlines empirical observations based on network traffic, and derives the IP Weight metrics based on these observations. Section 3 presents the outlier detection algorithm and the corresponding intrusion decision strategy. Section 4 presents the experimental evaluation of our approach and discusses the obtained results. Section 5 makes some concluding remarks and discusses future work.

2 Feature Analysis

Feature analysis consists of feature selection and extraction. In this paper we primarily focus on the detection of network attacks, and thus the main data source for our approach consists of network packets. Through empirical observations to network traffic behaviors, we derive a collection of empirical utility functions. We name these utility functions IP Weight metrics. IP Weight metrics measure the degree of anomalousness of IP packet flows. In the remainder of this section, we define the feature space, summarize these empirical observations, which are the basis of our feature selection process, and then derive the IP Weight metrics.

2.1 Feature Selection

Feature Space. Based on the standard characteristics of IP packets on networks, we define a set of features to describe a single IP packet. Let us denote by P the set of IP packets. A packet $p \in P$ can be represented as a 13-dimensional feature vector $< t$, *dip*, *dp*, *sip*, *sp*, *ihl*, *pktl*, *ident*, *fragoff*, *pro*, *thl*, *seq*, *ack*$>$; t is the time stamp corresponding to the appearing time of the packet in a certain time window; *dip* is the destination IP address and it usually corresponds to the address of a host we want to protect; *dp* stands for the destination port; *sip* stands for the source IP address; *sp* means source port; *ihl* refers to the length of IP header for the IP packet; *pktl* is the packet length including header and data; *ident* is an integer that identifies the current data in a packet, which can be used to piece together data fragments; *fragoff* is the offset of the IP packet indicating the position of the fragment's data relative to the beginning of the fragment data in the original data; *pro* stands for the upper-layer program receiving the incoming IP packets after IP processing is complete; *thl* is the length of TCP header; *seq* is the data location of the TCP segment; *ack* is the number of data received by the destination host.

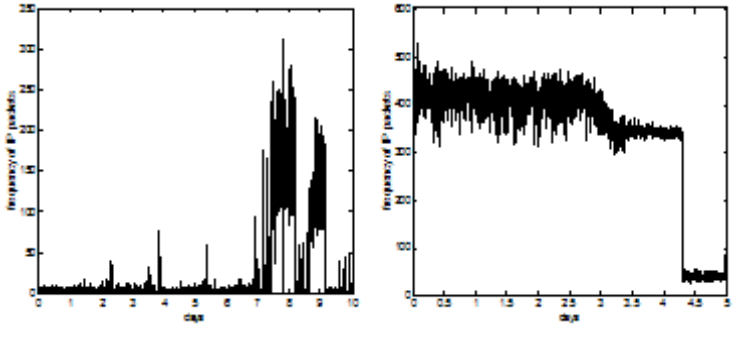
We define a packet flow as group of packets flowing to a specified destination during a specified observation period. Let us denoted by F the set of all packet flows. A packet flow $f \in F$ is defined as a 6-dimensional vector $f =< g$,t,δ_t,*dip*, *nop*,$nodp_{max} >$; where $g \in \wp(\mathrm{P})$ is the set of packets observed and $\wp(\mathrm{P})$ denotes the power set of P; t is the starting time of the observation; δ_t is the observation time window; *dip* refers to the destination IP address that we want to protect; *nop* stands for the total number of packets in the flow; $nodp_{max}$ means the maximum number of packets over all destination ports in the packet flow.

Empirical Observations. The goal for the feature selection is not to provide a full description about anomalous activities; instead, we are interested in identifying a limited number of facts that can allow achieving an effective and efficient detection. Specifically, the work is based on the following four intuitive observations:

1. Network traffic involving a high frequency of packets flowing to the same destination address within a very short time period, or network traffic involving a high frequency of packets flowing to the same destination address with same destination port during a very short time period, is likely anomalous.
2. During the normal network usage, for those network traffic flowing to the same destination over a given time period, the likelihood of their corresponding destination ports to be randomly distributed is low. The same observation applies for their corresponding source IP addresses and source ports.
3. Network traffic containing one or several packets that violate basic structural rules of packets is likely anomalous.
4. For a normal host, its incoming traffic and (matching) outgoing traffic are most likely similar.

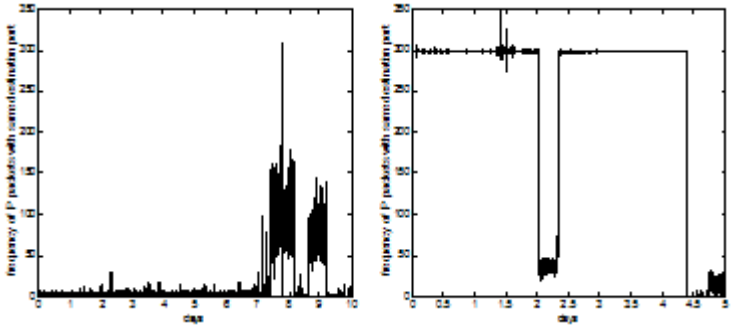
Observation 3 simply derives from the TCP/IP protocol specification. To confirm other three observations, we conducted a pilot study, in which network data are collected over three weeks: two weeks for normal network usage and one week for anomalous network usage including some known attacks. The destination server was deployed behind the firewall. We ensured that the traffic over two weeks' normal usage was normal by auditing the after-event logs of the firewall. During the anomalous network usage over one week, the server operated as a honey pot. Several utilities including known vulnerabilities were purposely installed on the server and exposed to the public.

Fig. 2-a to Fig. 7-b illustrate the analysis made from the collected data. In these figures, we denote the IP address of the server by *dip* and the size of the



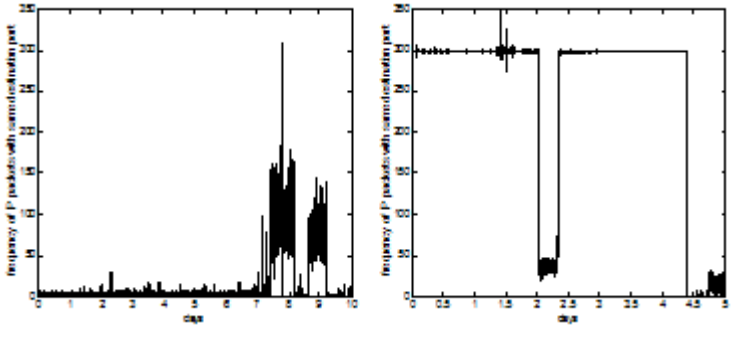
(a) (b)

Fig. 2. (a) Frequency of normal packets flowing to same *dip* over δ_t. (b) Frequency of anomalous packets flowing to same *dip* over δ_t.



(a) (b)

Fig. 3. (a) Maximum frequency of normal packets flowing to same *dip* with same *dp* over δ_t. (b) Maximum frequency of anomalous packets flowing to same *dip* with same *dp* over δ_t.

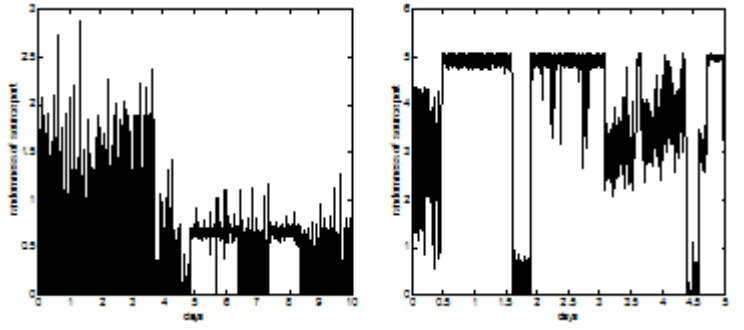


(a) (b)

Fig. 4. (a) Randomness of *dp* for normal packets flowing to same *dip* over δ_t. (b) Randomness of *dp* for anomalous packets flowing to same *dip* over δ_t.

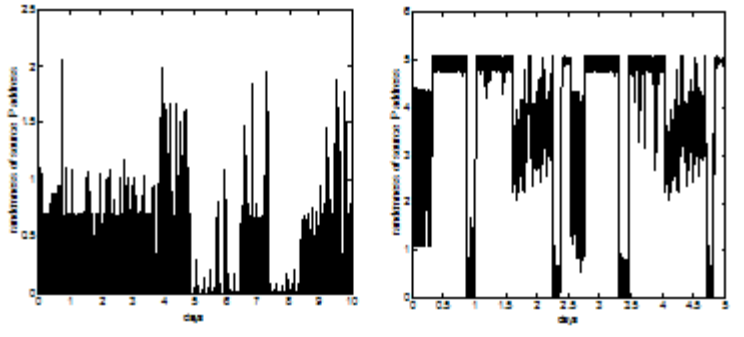
observation time window by δ_t. The frequency of IP packets flowing to the same *dip* during δ_t in two weeks normal network usage and one week anomalous network usage are plotted in Fig. 2-a and 2-b respectively. Fig. 3-a and Fig. 3-b plot the maximum frequency of packets flowing to same *dip* with same destination port during δ_t, respectively. In these figures, the frequency of normal packet flows follow a regular pattern, while the frequency of anomalous packet flows is persistently high. These confirm observation 1.

The randomness of destination ports in packet flows with same destination port during δ_t over two weeks' normal network usage and one week's anomalous network usage are plotted in Fig. 4-a and Fig. 4-b, respectively. Similarly, Fig. 5-a and Fig. 5-b plot the randomness of source ports in corresponding packet flows. Fig. 6-a and Fig. 6-b plot the randomness of source IP addresses. In these



(a) (b)

Fig. 5. (a) Randomness of *sp* for normal packets flowing to same *dip* over δ_t. (b) Randomness of *sp* for anomalous packets flowing to same *dip* over δ_t.



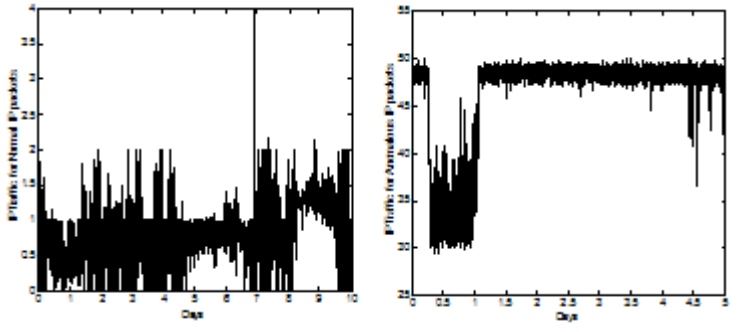
(a) (b)

Fig. 6. (a) Randomness of *sip* for normal packets flowing to same *dip* over δ_t. (b) Randomness of *sip* for anomalous packets flowing to same *dip* over δ_t.

graphs, the randomness of destination ports, source IP addresses and source ports of anomalous packet flows is more often higher than those of normal packet flows, which supports observation 2.

For the server we protect, Fig. 7-a and Fig. 7-b plot the load ratio of its corresponding incoming traffic to outgoing traffic over two weeks normal network usage and one week anomalous network usage. Both the incoming traffic and outgoing traffic correspond to the same destination IP address. The load ratio for the specified destination IP address of anomalous traffic is much higher than those of normal traffic, which confirms the observation 4.

Exceptions for these observations may occur in some special cases. For instance, a normal web sever working on high traffic load will violate the first observation. However, this kind of violations has a slight impact on the final intrusion decisions according to later experimental evaluations. This is because the outlier detection technique eliminates empirical observation errors. We con-



(a) (b)

Fig. 7. (a) Load ratio for normal packets with same *dip* over δ_t. (b) Load ratio for anomalous packets with same *dip* over δ_t.

sider only the most generic cases when we derive empirical utility functions of network traffic.

2.2 Feature Extraction

In order to achieve efficient and effective detection, we extract a limited feature set consisting of four dimensions by applying some transformations on the feature set denoted by F. Specifically, four ordinal utility functions are defined to characterize the degree of anomalousness of network activities, and each of them maps several features in F into a single numerical feature [16]. We name the four utility functions IP Weight. Each of the functions measures empirically the anomalousness along one of four dimensions, namely frequency, randomness, structure and load. We denote by ipw_{freq}:$F \rightarrow R$, ipw_{ran}:$F \rightarrow R$, ipw_{str}:$F \rightarrow R$, and ipw_{load}:$F \rightarrow R$ respectively the frequency, randomness, structure and load component of IP Weight, where R is the set of real numbers. An empirical assumption behind IP Weight metrics is that *the greater the value of IP Weight, the more anomalous a packet flow f∈F is.* All four components of IP weight metrics must satisfy this empirical assumption, and as a result, the underlying logic to derive the corresponding four utility functions is that *the greater the value of utility functions, the more anomalous a packet flow f∈F is.*

Frequency-Based Feature Extraction. Given a packet flow $f \in F$ with same *dip* during δ_t, we denote by x_1 and x_2 the appearing frequency of IP packets during δ_t and the maximum appearing frequency of IP packets over all destination ports during δ_t, respectively. Thus, we have:

$$x_1 = \frac{nop}{\delta_t} \text{ and } x_2 = \frac{nodp_{\max}}{\delta_t}$$

Empirical observations show that the greater the value of x_1 and x_2, the more likely the corresponding network packet flow is anomalous. Both x_1 and x_2 contribute to some extent the anomalousness of network traffic. As a result, we define ipw_{freq} by adopting a polynomial representation, which is expressed as follows:

$$ipw_{freq}(f) = (x_1 f_1(x_1, x_2) + x_2 f_2(x_1, x_2))g(x_1, x_2). \quad (1)$$

Where f_1: $R \times R \rightarrow [0,1]$ and f_2: $R \times R \rightarrow [0,1]$ are two numerical functions that represent the contributions of x_1 and x_2 respectively; $g : R \times R \rightarrow R$ is a numerical function adjusting the value of ipw_{freq}, which is selected as x_2/x_1. Given x_1 and x_2, the constraints $x_2 \leq x_1$ and $f_1(x_1,x_2) + f_2(x_1,x_2) = 1$ are always satisfied, and thus by selecting f_2 as x_2/x_1, we can derive the expression of f_1. By substituting f_1, f_2 and g in equation (1), we can express the empirical utility function ipw_{freq} as follows:

$$ipw_{freq}(f) = \left(x_1 - x_2 + x_1^{-1} x_2^2\right) \frac{x_2}{x_1}. \quad (2)$$

Randomness-Based Feature Extraction. The entropy is selected to measure the randomness of variables according to information theory. Given a packet flow$f \in F$ with same *dip* during δ_t, the randomness of their corresponding source IP addresses, ports, and destination ports is denoted by H_{sip}, H_{sp} and H_{dp}, respectively; $p(sip)$ refers to the appearing probability associated with source IP addresses *sip*, which is computed by taking the ratio of number of packets with specified source IP address *sip* by the total number of packets observed in the flow f. Using the same approach, we can compute the $p(sp)$ and $p(dp)$, which refer to the probabilities associated with source port *sp* and destination port *dp*, respectively.

Since each of these features has the same contribution to the anomalousness of network traffic, we combine them into a single feature by selecting their maximum value in order to satisfy the empirical assumption of IP weight metrics. Consequently, the utility function ipw_{ran} is defined as follows:

$$ipw_{ran}(f) = \max(H_{sip}(f), H_{sp}(f), H_{dp}(f)) \quad (3)$$

Load-Based Feature Extraction. For a normal target host *dip*, the magnitudes of its incoming and (matching) outgoing traffic are most likely similar. However, when denial of service (DoS) attacks are used to compromise this host, its outgoing traffic is usually low compared to its incoming traffic. Empirical observations made earlier confirm this.

Given a packet flow$f \in F$, we extract two new features $traffic_{in}$ and $traffic_{out}$ to represent the appearing frequency of packets flowing to *dip* over δ_t and the appearing frequency of packets outgoing from *dip* over δ_t. The ratio between $traffic_{in}$ and $traffic_{out}$ describes the load balance of the host we want to protect. Thus the utility function ipw_{load} is defined as follows:

$$ipw_{load}(f) = \frac{traffic_{in}}{traffic_{out}} \quad (4)$$

Load-Based Feature Extraction. Normal IP packets must satisfy some basic structural rules. In most cases, TCP/IP implementation will help to check the structures of packets. However, in some cases, the structure violation is difficult

to be found by TCP/IP stack implementation. According to our domain knowledge on the structure of IP packets, we define a limited number of rules, which are usually satisfied by any pair of packets belonging to the same TCP/UDP connection. Fig. 8 describes the rule base for these rules. Given a packet flow$f \in F$ with same *dip* during δ_t and a pair of packets p_1,$p_2 \in g$ observed during δ_t, the utility function ipw_{str} is defined as follows:

$$ipw_{str}(f) = \frac{1}{nop(nop-1)} \sum_{p \in g} \sum_{\substack{q \in g, \\ p \neq q}} \varepsilon_{pq} e^{-|ident(p) - ident(q)|} \tag{5}$$

rule 1 - if $ident_1$=$ident_2$, then $fragoff_1 \neq fragoff_2$
rule 2 - if $ident_1$>$ident_2$+1 then $seq_1 \geq seq_2$ and $ack_1 \geq ack_2$
rule 3 - if $ident_2$>$ident_1$+1 then $seq_2 \geq seq_1$ and $ack_2 \geq ack_1$
rule 4 - if $ident_1$=$ident_2$+1 then seq_1=seq_2+$pktl_2$−ihl_2−thl_2 and $ack_1 \geq ack_2$
rule 5 - if $ident_2$=$ident_1$+1 then seq_2=seq_1+$pktl_1$−ihl_1−thl_1 and $ack_2 \geq ack_1$

Fig. 8. Rule-base for packets in the same connection

Where |*ident*(*p*)-*ident*(*q*)| stands for the absolute value of the difference between identification fields of packets *p* and *q*; Coefficient *nop*(*nop*-1) is a normalizing factor, and *nop* stands for the total number of packets in the flow *f*. ε_{pq} is a positive integer, which takes the following values: ε_{pq}=1 if packets *p* and *q* belong to the same TCP/UDP connection and violate the rule base simultaneously; ε_{pq}=0 otherwise.

|*ident*(*p*)-*ident*(*q*)| is used to measure the uncertainty of two packets belonging to the same connection. It is difficult to establish that two packets belong to the same connection with full certainty although the notion of connection-similarity has defined a precondition that IP packets belonging to the same TCP/UDP connection must have *the same source IP addresses and ports, the same destination IP addresses and ports, and the same protocol types.* The standard specification and empirical observation show that given two packets *p* and *q* which satisfy the connection-similarity precondition, the smaller the difference value |*ident*(*p*)-*ident*(*q*)|, the higher the probability of two packets *p* and *q* belonging to the same connection.

3 Outlier Detection and Intrusion Decision

We use Gaussian mixture model (GMM) to detect the outlier from a given dataset. In pattern recognition, it was established that Gaussian mixture distribution could approximate any distribution up to arbitrary accuracy, as long as a sufficient number of components are used [15], and thus the unknown probability density function can be expressed as a weighted finite sum of Gaussian

with different parameters and mixing proportions [18]. Given a random variable x, its probability density function $p(x)$ can be represented as a weighted sum of components:

$$p(x) = \sum_{i=1}^{k} a_i f_i(x; u_i, v_i)$$

Where k is the number of mixture components; a_i ($1 \leq i \leq k$) stand for the mixing proportions, whose sum is always equal to 1. $f_i(x;u_i,v_i)$ refers to the component density function, in which u_i stands for the mean of variable x and v_i is the variance of x,. The density function can be a multivariate Gaussian or a univariate Gaussian.

Table 1. Proposed outlier detection algorithm

Function: GMM_Outlier_Detection (dataset and k) **returns** outlier data set
Inputs: *dataset* $X \sim \{x_n | n = 1, 2, ..., N\}$, *and the estimated number of components* k
Output: Outlier Data Set
Initialization:
Outlier Data Set = ϕ; $j \leftarrow 0$;
Initial parameters $\{\alpha_i^j, \mu_i^j, \nu_i^j\}$, $1 \leq i \leq k$, are randomly generated;
Calculate the initial log-likelihood L_j;
Repeat:
For $1 \leq i \leq k$, $1 \leq n \leq N$
If ($\alpha_i^j \geq outlier_{thres}$**) then** compute posterior probability $p_j(i|x_n)$;
Else $p_j(i|x_n) = 0$;
$j \leftarrow j + 1$;
Re-estimate $\{\alpha_i^j, \mu_i^j, \nu_i^j\}$ by using $p_{j-1}(i|x_n)$, $1 \leq i \leq k$, $1 \leq n \leq N$;
Calculate the current log-likelihood L_j;
Until: $|L_j - L_{j-1}| < th_1 \text{or} j > th_2$
For $1 \leq i \leq k$, $1 \leq n \leq N$
If $(p_{j-1}(i|x_n) = 0)$, assign x_n to Outlier Data Set
Return Outlier Data Set;

Expectation-Maximization (EM) algorithm has been suggested as an effective algorithm to estimate the parameters of GMM [3]. In the E-step, the posterior probability $p(i|x_n)$ is calculated for each data $X \sim \{x_n | n=1,2,\ldots,N\}$ and each mixture component $i(1 \leq i \leq k)$. In M-step, the set of parameters $\{a_i, u_i, v_i\}$ are re-estimated based on posterior probabilities $p(i|x_n)$, which maximize the likelihood function. The EM algorithm starts with some initial random parameters and then repeatedly applies the E-step and M-step to generate better parameter estimates until the algorithm converges to a local maximum.

Our outlier detection algorithm is based on the posterior probability generated by EM algorithm. The posterior probability describes the likelihood that the data pattern approximates to a specified Gaussian component. The greater the posterior probability for a data pattern belonging to a specified Gaussian component, the higher the approximation is. As a result, data are assigned to the

corresponding Gaussian components according to their posterior probabilities. However, in some cases there are some data patterns whose posterior probability of belonging to any component of GMM is very low or close to zero. These data are naturally seen as the outliers or noisy data. We illustrate the detailed outlier detection algorithm in Table 1.

Thresholds th_1 and th_2 correspond to the termination conditions associated with the outlier detection algorithm: th_1 measures of the absolute precision required by the algorithm and th_2 is the maximum number of iterations of our algorithm. Threshold $outlier_{thres}$ refers to the minimum mixing proportion. Once the mixing proportion corresponding to one specified Gaussian component is below $outlier_{thres}$, the posterior probability of the data pattern belonging to this Gaussian component will be set to 0.

The intrusion decision strategy is based on the outcome of outlier detection: *if no outlier data are detected, the network packet flows is normal; otherwise, the packet flows represented by this outlier is reported as the intrusion.* This strategy is reasonable based on the empirical assumption that *the greater the value of IP Weight metrics, the more anomalous the network packet flows.*

4 Evaluation

4.1 Offline Evaluation

The 1998 DARPA intrusion detection dataset is the first standard corpus used for evaluating intrusion detection approaches offline [11]. Over 300 attacks are simulated in nine weeks. Training data are generated in the first seven weeks and testing data are derived in the rest two weeks. The attacks consisting of a total of 33 different attack types are divided into four different attack categories, namely DoS, R2L, U2S and Probing. In this paper, the proposed IP Weight metrics characterize the anomalousness of packet flows and hence we are interested in those multiple-connection based network intrusions, in which attacks are involved into multiple network connections and IP Weight metrics can be calculated from these multiple connections (i.e. packet flows). During the experiment, we extracted these attacks from the DARPA dataset and established a new multiple-connection based network intrusion dataset, which contains a combination of DoS, R2L, and Probing attacks. Specifically, we select 17 days' data from the total 45 days' (nine weeks) dataset. The corresponding attack types include *synflood*, *smurf*, *pod*, *teardrop*, *portsweep*, *ipsweep*, *land*, *back*, *saint*, *udpstorm*, *guest*, *nmap*, *satan*, *imap*, *dict*, *apache2*, *processtable*, *mscan* and *mailbomb*.

One data record for the 1998 DARPA intrusion detection dataset mainly includes nine fields, namely index, traffic start date, traffic start time, duration, service type, source port, destination port, source IP address and destination IP address. Based on these data, we calculate three components of IP Weight metrics, namely ipw_{freq}, ipw_{ran} and ipw_{str}. The fourth component ipw_{load} cannot be derived in the offline evaluation since the DARPA dataset didn't provide the exact incoming or outgoing traffic information for the protected target host. As

a result, the specified network traffic in DARPA dataset are represented by total 3600 instances along with three dimensions, namely ipw_{freq}, ipw_{ran} and ipw_{str}. 3493 instances represent the normal network traffic and 107 instances represent the intrusive traffic, which include 19 types of multiple-connection based network intrusions.

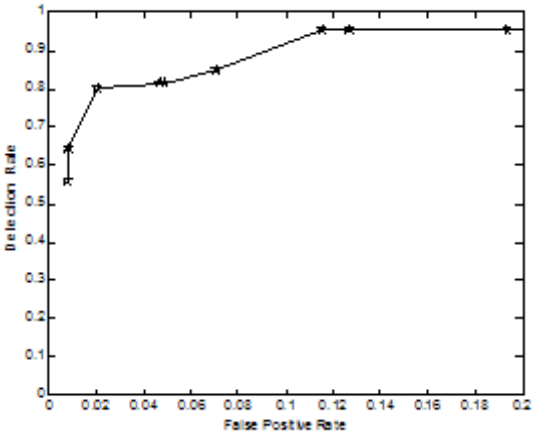


Fig. 9. ROC curves for our detection system

Two performance metrics are used to evaluate the detection effectiveness, namely detection rate (DR) and false positive rate (FPR). DR is the ratio of the number of attack instances detected to the total number of attack instances and FPR is the ratio of the number of normal instances detected as alerts to the total number of normal instances. We calculate the DR and FPR by varying the threshold $outlier_{thres}$ and then plot the receiving operator characteristic (ROC) curve in Fig. 9. The best detection result is obtained at a threshold of 10^{-8}, which corresponds to <DR = 80.37%, FPR = 2.03%> and 16 types of multiple-connection based attacks out of a total of 19 attack types are detected at this point. Attacks *processtable*, *dictionary*, and *guest* (variant of *dictionary*) are missed by our detection system.

4.2 Online Evaluation

Online evaluation in a real networking environment is conducted to assess the efficiency and effectiveness of our detection system. The implementation of our system includes two modules, namely grader and detector. In the grader, IP packets are collected and then IP Weights are calculated according to the corresponding metrics. The IP Weights are then stored on a database. Synchronously, the detector reads IP Weights from database, detects the outlier data for IP Weights and then makes intrusion decisions.

The hardware topology of the live networking environment includes two LANs: LAN1 and LAN2. The internal attack traffic is generated from LAN1;

LAN2 is the victim network. Our detection system is deployed on a specified sever in LAN2. During the evaluation, we executed six types of network attacks from LAN 1, four of which, categorized as distributed denial of service (DDoS) attacks, include *synflood*, *smurf*, *udpflood* and a mixture of *synflood* and *udpflood*. The other two attacks fall into the category of probing attacks, namely *xscan* [19] and *fluxay* [7].

Table 2. Statistical information about response time for real attacks

Attack	udpflood	synflood	smurf	mixture flood	fluxay	xscan
Avg. res. time (s)	2.2	1.9	2.8	2.5	5.7	7.6

During the evaluation, each attack was repeated ten times and corresponding attack *starting time* and intrusion *detection time* were recorded. The intrusion *response time* is calculated as follows:

$$response\ time = detection\ time - starting\ time$$

The online evaluation shows that our detection system detected all the six types of attacks and the corresponding response time is on the second level. Table 2 illustrates the average response time for each attack over 10 times.

5 Conclusions

In this paper, we propose a real-time online intrusion detection framework, in which network packet flows are first characterized and quantified by using some anomalousness metrics named IP Weight and then an online outlier detection algorithm is used to detect the outlier (noisy) data of IP Weight metrics. Intrusion decisions are made according to the outcome of outlier detections. The offline evaluation with the 1998 DARPA intrusion detection dataset and the online evaluation in a live networking environment show the efficiency and effectiveness of our detection system.

The future works include improving the detection rate and decreasing false alarm rate for our system. The best result in the offline evaluation yields a 80.37% detection rate with a 2.03% false alarm rate. Although it was suggested that the DARPA dataset itself has some flaws [13], the offline experimental results still provide a strong criteria for the performance of our detection framework. In order to address the above issues, two substantial works are necessary in the future. One is the characterization of the 'normal' network packet flows by deriving new metrics under the constraint of effectiveness and efficiency. The other is related to the detection techniques discriminating anomalous behaviors from normal profiles and the decision strategies verifying intrusive or normal.

References

1. Anderson, J.P.: Computer Security Threat Monitoring and Surveillance. Technical Report, James P. Anderson Co., Fort Washington, Pennsylvania (1980)
2. Axelsson, S.: The Base-Rate Fallacy and the Difficulty of Intrusion Detection. ACM Transactions on Information and System Security (TISSEC), Vol. 3. (2000) 186-201
3. Dempster, A. P., Laird, N. M. and Rubin, D. B.: Maximum Likelihood from Incomplete Data via the EM Algorithm (with discussion). Journal of the Royal Statistical Society B, Vol. 39. (1977) 1-38
4. Denning, D. E.: An Intrusion Detection Model. IEEE Transactions on Software Engineering, No. 2. (1987) 222-232
5. Eskin, E.: Anomaly Detection over Noisy Data using Learned Probability Distributions. Proceedings of 17th International Conference on Machine Learning. (2000) 255-262
6. Eskin, E., Arnold, A., Prerau, M., Portnoy, L. and Stolfo, S.: A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. On Application of Data Mining in Computer Security, Kluwer Academic Publisher (2002)
7. Fluxay, http://www.netxeyes.com
8. Frank, J.: Artificial Intelligence and Intrusion Detection: Current and Future Directions. Proceedings of the 17th National Computer Security Conference, (1994) 11-21
9. Forrest, S., Hofmeyr, S.A. and Longstaff, T.A.: A Sense of Self for Unix Processes. Proceedings of 1996 IEEE Symposium on Security and Privacy. (1996) 120-128
10. Hochberg, J., Jackson, K., Stallings, C., McClary, J. F., DuBois, D. and Ford, J.: NADIR: An Automated System for Detecting Network Intrusion and Misuse. Computers & Security, 12 (3). (1993) 235-248
11. Kendall, K.: A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Master's Thesis, Massachusetts Institute of Technology (1998)
12. Lunt, T., Jagannathan, R., Lee, R., Listgarten, S., Edwards, D., Neumann, P., Javitz, H. and Valdes, A.: IDES: The Enhanced Prototype, A Real-time Intrusion Detection System. Technical Report, SRI Project 4185-010, Computer Science Laboratory, CA. (1988)
13. McHugh, J.: The 1998 Lincoln Lab IDS Evaluation - A Critique. Proceedings of Recent Advances in Intrusion Detection, No. 1907 in LNCS. (2000) 145-161
14. Portnoy, L., Eskin, E. and Stolfo, S.: Intrusion Detection with Unlabeled Data using Clustering. Proceedings of ACM CSS Workshop on Data Mining Applied to Security (2001)
15. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge, U.K., Cambridge University Press (1996)
16. Roberts, F. S.: Measurement Theory. Addison-Wesley Publishing Company (1979)
17. Smaha, S. E.: Haystack: An Intrusion Detection System. Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference. (1988) 37-44
18. Titterington, D., Smith, A. and Makov, U.: Statistical Analysis of Finite Mixture Distributions. John Wiley & Sons, New York (1985)
19. X-scan, http://www.xfocus.org