Computer Communications 34 (2011) 2227–2235

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Practical real-time intrusion detection using machine learning approaches

Phurivit Sangkatsanee [a], Naruemon Wattanapongsakorn [a,*], Chalermpol Charnsripinyo [b]

[a] Department of Computer Engineering, Faculty of Engineering, King Mongkut's University of Technology Thonburi, 126 Pracha-utid Road, Toongkru, Bangkok 10140, Thailand
[b] National Electronics and Computer Technology Center, 112 Phahonyothin Road, Klong Luang, Pathumthani 12120, Thailand

ARTICLE INFO

Article history:
Received 6 October 2010
Received in revised form 11 March 2011
Accepted 2 July 2011
Available online 13 July 2011

Keywords:
Network intrusion detection
Machine learning
Denial of Service
Probe

ABSTRACT

The growing prevalence of network attacks is a well-known problem which can impact the availability, confidentiality, and integrity of critical information for both individuals and enterprises. In this paper, we propose a real-time intrusion detection approach using a supervised machine learning technique. Our approach is simple and efficient, and can be used with many machine learning techniques. We applied different well-known machine learning techniques to evaluate the performance of our IDS approach. Our experimental results show that the Decision Tree technique can outperform the other techniques. Therefore, we further developed a real-time intrusion detection system (RT-IDS) using the Decision Tree technique to classify on-line network data as normal or attack data. We also identified 12 essential features of network data which are relevant to detecting network attacks using the information gain as our feature selection criterions. Our RT-IDS can distinguish normal network activities from main attack types (Probe and Denial of Service (DoS)) with a detection rate higher than 98% within 2 s. We also developed a new post-processing procedure to reduce the false-alarm rate as well as increase the reliability and detection accuracy of the intrusion detection system.

1. Introduction

Internet services have become essential to business commerce as well as to individuals. With the increasing reliance on network services, the availability, confidentiality, and integrity of critical information have become increasingly compromised by remote intrusions. Enterprises are forced to fortify their networks against malicious activities and network threats. Therefore, a network system must use one or more security tools such as a firewall, antivirus software or an intrusion detection system to protect important data/services from hackers or intruders.

Relying on a firewall system alone is not sufficient to prevent a corporate network from all types of network attacks. This is because a firewall cannot defend the network against intrusion attempts on open ports required for network services. Hence, an intrusion detection system (IDS) is usually installed to complement the firewall. An IDS collects information from a network or computer system, and analyzes the information for symptoms of system breaches. As shown schematically in Fig. 1, a network IDS monitors network data and gives an alarm signal to the computer user or network administrator when it detects antagonistic activity on an open port. This signal allows the recipient to inspect the system for more symptoms of unauthorized network activities.

Network intrusion detection systems can be classified into two types which are host-based and network-based intrusion detection. Host-based detection captures and analyzes network data at the attacked system itself while the network-based detection captures and inspects online network data at the network gateway or server, before the attack reaches the end users. In addition, network intrusion detection systems can operate in two modes which are off-line detection and on-line detection. An off-line network intrusion detection system periodically analyzes or audits network information or log data to identify suspected activities or intrusions. In an on-line network intrusion detection system, the network traffic data has to be inspected as it arrives for detecting network attacks or malicious activities.

In this paper, we focus on real-time network-based intrusion detection where the incoming network data is captured on-line and the detection result is reported instantaneously or within a fraction of a minute, so that the network administrator is notified and can stop the on-going attack. Our approach also could be applied as host-based detection. We designed our intrusion detection system (IDS) using a misuse detection technique. The misuse detection approach can classify attacks into categories. In contrast, the anomaly detection approach can only differentiate between normal activity and abnormal/attack activity. Although, there are many possible features of network data that could serve as input to an IDS, we propose to consider only 12 features of network traffic data extracted from the headers of data packets. We show that these 12 features are effective in identifying normal network

* Corresponding author. Tel.: +66 2 470 9089; fax: +66 2 872 5050.
E-mail address: naruemon@cpe.kmutt.ac.th (N. Wattanapongsakorn).

0140-3664/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2011.07.001

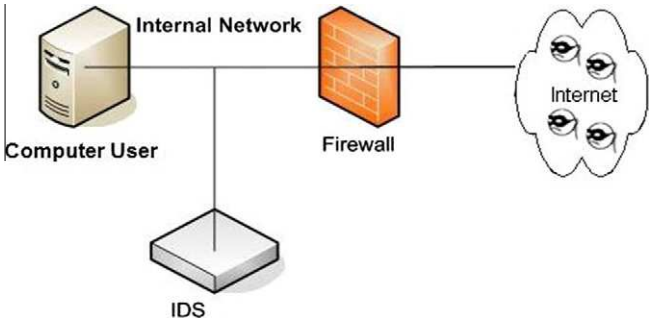

Fig. 1. Network intrusion detection system environment.

activity and classifying main attack activities into two attack types namely Port Scanning (PS or probing) and Denial of Service (DoS). Using a small number of features reduces the complexity of data analysis and thus can increase the detection speed and reduce computer resource (CPU and memory) consumption.

We are also interested in designing an IDS by applying well-known machine learning algorithms. This will give us a simple but efficient intrusion detection approach, so that anyone can easily adopt it. To do that, we consider and compare several existing machine learning algorithms which are Decision Tree, Ripper Rule, Back-Propagation Neural Network, Radial Basis Function Neural Network, Bayesian Network, and Naïve Bayesian for classifying incoming network data. Our experimental results showed that the Decision Tree was superior to the other approaches in terms of detection accuracy. Therefore, we further developed the Decision Tree approach for our Real-Time Intrusion Detection System (RT-IDS), where the input network data is captured on-line in a real environment. In addition, we designed an optional post-processing procedure for our real-time intrusion detection system to reduce the false alarm rate of the intrusion detection and thus increase its reliability. We evaluated our IDS in terms of the detection speed, CPU usage, memory consumption, and detection accuracy.

The remainder of this paper is organized as follows. In the next section, we describe and review related work in network intrusion detection systems. In Section 3, we present our research approaches, using information gain to select input features. We also provide a brief overview of the machine learning techniques that we evaluated. In Section 4, we present our real-time IDS (RT-IDS) process and algorithm. Each phase of the RT-IDS process is described in detail. We present the 12 essential features of network data relevant to classifying network attacks. A post-processing method for reducing false alarm rate of intrusion detection is also presented. In Section 5, we present the details of our experiments and performance evaluation. The experimental design and experimental data are described. The experimental results for different scenarios are presented. Lastly, in Section 6, we offer conclusions from our research work and summarize its contribution.

2. Literature survey

In previous research, most researchers have concentrated on off-line intrusion detection using a well-known KDD99 benchmark dataset to verify their IDS development. The KDD99 dataset is a statistically preprocessed dataset which has been available from DARPA since 1999 [13]. There also exist a few on-line intrusion detection approaches, which are discussed in Section 2.2.

2.1. Off-line network intrusion detection

The artificial neural network approach is one of the most popular techniques for the design of IDS. Jirapummin et al. [1] proposed a hybrid neural network using a combination of Self-Organizing Map (SOM) and Resilient Back-Propagation Neural Network (BPNN). To evaluate their approach, they used an available well-known preprocessed dataset which is KDD99 [2]. The KDD99 dataset is a network packet dataset consisting of normal network activity as well as many network attack types. The dataset is based on the DARPA98 dataset from MIT Lincoln laboratory, which provides answer class (labeled data) for evaluation of intrusion detection.

Pan et al. [3] designed a hybrid system by using a BPNN and a C4.5 Decision Tree considering the KDD99 dataset. The results showed that using only a BPNN without C4.5 Decision Tree, their system could not detect the network attack types such as User to Root (U2R) and Root to Local (R2L) at all. Moradi and Zulkernine [4] used a Multi- Layer Perceptron (MLP) artificial neural network in off-line mode to classify normal network activity, Satan (Probe) attacks and Neptune attacks using the KDD99 dataset. Ngamwitthayanon et al. [5] designed a multi-state IDS system to classify normal data and each attack type using the KDD99 dataset. Their results showed a higher detection rate in each classification category than when only a single state was used to classify all categories.

Fuzzy Sets is another technique often used for IDS. This technique generally falls into two categories, fuzzy misuse detection [6,7] and fuzzy anomaly detection [8–10]. Abraham and Jain [6] used three types of fuzzy rules to compare with linear generic programming (LPG), Decision Tree, and Support Vector Machines (SVM). Their result showed that one of their fuzzy rules gave the best detection rate using the 41 features of the DARPA 1998 dataset. Liao et al. [7] used fuzzy logic and an expert system with the DARPA 2000 dataset and achieved more than 91.5% detection rate over all attack types, while reducing complexity of traditional techniques for ranking fuzzy numbers. Ngamwitthayanon et al. [8] used Fuzzy Adaptive Resonance Theory (F-ART) to implement network anomaly intrusion detection with the KDD99 dataset. Similarly, Toosi and Kahani [9] and Tsang et al. [10] also used fuzzy rules as their intrusion detection approaches and evaluated with the KDD99 dataset.

Pukkawanna et al. [11] proposed BLINd classification or BLINC model which is a graphlet model of network packet patterns. BLINC could detect normal network activity and four network attack types using a signature-based detection approach.

Lee et al. [27] proposed a Decision Tree approach using ID3 learning algorithm for anomaly detection (unknown or new attacks). They considered four types of attacks which are DoS, Probe, U2R and R2L using the DARPA dataset that KDD99 dataset is based on. Most of their anomaly detection rates were lower than the detection rates with known/trained data.

Other off-line intrusion detection approaches have also been proposed using the KDD99 as input dataset such as Katos et al. [12] where analysis of data and clustering evaluation were investigated, and Chen et al. [13] where anomaly score of a packet was computed based on the deviation from the normal behavior.

2.2. On-line (real-time) network intrusion detection

A system that can detect network intrusion while an attack is occurring is called a real-time detection system. A real-time IDS captures the present network traffic data which is on-line data. In this paper, the terms "on-line detection" and "real-time detection" are used interchangeably. In our literature survey, we found very few on-line (real-time) network IDS approaches that have been proposed previously. We discuss these few studies in this section.

Labib and Vemuri [14] developed a real-time IDS using Self-Organizing Maps (SOM) to detect normal network activity and DoS attack. They preprocessed their dataset to have 10 features for each data record. Each record contained information of 50 packets. Their IDS was evaluated by human visualization for different

characteristics of normal data and DoS attack. No detection rate was reported.

Puttini et al. [15] used a Bayesian classification model for anomaly detection to classify normal network activity and attack using a 3-month training dataset and a 1-month test dataset. They evaluated their approach by adjusting a penalty value to see how it affected the classification results. They also needed human expert to visualize the normal and abnormal network behaviors. No detection rate was reported.

Amini et al. [16] designed a real-time IDS using two unsupervised neural network algorithms which are Adaptive Resonance Theory (ART) and Self-Organizing Map (SOM). They classified two attack types plus normal data during a 4-day experiment with a 27-feature dataset, where each feature captures number of occurrences of an event in each time interval. The detection results showed that the ART-2 gave higher detection speed and detection rate than the SOM. The detection rate was reported to be over 97%, separating normal traffic data from network attacks. However, the attacks were not classified into types or categories.

Su et al. [17] created a real-time network IDS using fuzzy association rules and conducted their experiments by using four computers with 30 DoS attack types in WIN32. They could separate the normal network activity from network attacks but they did not identify the attack type. They preprocessed the network data to have 16 features. After testing, the results showed that the 30 DoS attack types have a similarity ratio of less than 0.4 while normal network activity gave a similarity ratio more than 0.75. The similarity ratio represents how close or similar the data is to normal data, i.e. 1.0 means that they are perfectly matched.

Li et al. [18] developed a high-speed intrusion detection model using TCP/IP header information. However, their approach is limited to only one type of attack which is DoS.

A well-known intrusion detection tool called SNORT was studied in [19]. SNORT has become a commercial tool. Its attack signature rules are available only to their registered customers. The signature rules or patches have to be frequently updated and installed in order to detect current attack types.

In summary, most researchers proposed IDS classification algorithms based on machine learning techniques and used KDD 99 dataset to evaluate their IDS approaches in an off-line environment without considering real-time processing/detection. The KDD99 dataset, which was created about 10 years ago, is complex and lacks of many current attack types. While this approach is of theoretical interest, it provides only post hoc assistance to network administrators and thus is less useful for building practical tools. Although a few researchers have started investigating real-time intrusion detection systems using different techniques, most of the on-line IDS performance still needs further improvement. Some on-line IDSs require a human expert to visualize or identify the attacks, while the remaining systems have other limitations. For example, some IDS can separate normal network data from attack data, but cannot classify attack type. Some IDSs were designed to detect only one attack type.

In this paper, we are interested in developing a practical real-time IDS approach which can be applied with well-known machine learning algorithms. The approach should be simple but efficient in detecting network intrusion in an actual real-time environment. We describe the concept of information gain and machine learning methods which are applied to our real-time IDS approach in the following section.

3. Information gain and machine learning techniques

Intrusion detection is fundamentally a problem of classification. An IDS must classify a given set of network packets as normal or attack. Various characteristics or features of the network data stream provide input to the classification. The number of features used impacts the complexity of computation and the amount of computer resources required. Thus we would like to identify the smallest possible set of relevant or effective features.

In this section, we present the concept of using information gain to identify relevant features of network data packets that are captured and preprocessed in our IDS. The information gain parameter measures how relevant each network feature is when used for data classification. We also describe briefly the concepts of machine learning techniques which can be used to classify data into categories.

3.1. Information gain

Information Gain (InGain) is a criterion for feature selection. To use InGain, we compute an entropy value for each attribute or feature of data. The entropy value is used for ranking features that affect data classification. If a feature has very low information gain then it does not have much influence on data classification. Thus, a feature with low information gain value can sometimes be ignored without affecting the detection accuracy. Formal definition of information gain is as follows.

Let X and Y be discrete variables representing sample attributes $(x_1,x_2,\ldots,x_m)$, and class attributes $(y_1,y_2,\ldots,y_n)$, respectively. The information gain of a given attribute X with respect to the class attribute Y is the uncertainty reduction of the value of Y when we know the value of X. $Entropy(Y)$ is the uncertainty of the value of Y. The $Entropy(Y|X)$ is the uncertainty of the value of Y when we know the value of X. The information gain can be mathematically presented as

$$InGain(Y;X) = Entropy(Y) - Entropy(Y|X)$$

where $Entropy(Y) = -\sum_{i=1}^{n} P(Y=y_i)\log_2 P(Y=y_i)$

$P(Y = y_i)$ = probability that the class attribute y_i occurs

$$Entropy(Y|X) = -\sum_{j=1}^{m} P(X=x_j)Entropy(Y|X=x_j)$$

The information gain can be also represented as

$$InGain(Y;X) = Entropy(X) + Entropy(Y) - Entropy(X,Y)$$

The $Entropy(X,Y)$ is the joint entropy of X and Y, so

$$Entropy(X,Y) = -\sum_{i,j} P(X=x_j, Y=y_j)\log_2 P(X=x_j, Y=y_i)$$

In this paper, X defines individual feature of inputs in classification, and Y defines attack class (Normal data, Probe attack and DoS attack). In our research, we computed the InGain for a large number of candidate features. We selected a small number of features with high InGain to use in our network traffic classification.

3.2. Machine learning techniques

There are many machine learning techniques which can be used for data classification. We present the general concepts and a brief description of some popular supervised learning techniques that generally give high detection accuracy for our IDS approach [21] below.

3.2.1. Decision Tree

Decision Tree is a well-known classification algorithm which can efficiently classify data [3]. A Decision Tree consists of non-terminal nodes (a root and internal nodes) and terminal nodes (leaves). The root node is the first attribute with test conditions to split each input data record toward each internal node depend-

ing on characteristics of the data record. First, the Decision Tree is trained with known data before it can classify new or untrained data. The training process builds the Decision Tree by identifying attributes and values that would be used to test the input data at each internal node. After training, the tree can predict or classify new data by starting from a root node and traversing internal nodes based on attributes of test conditions until it arrives at a leaf (terminal) node consisting of an answer class. The C4.5 version is an efficient and popular algorithm for training and building a Decision Tree [22].

3.2.2. Ripper Rule

Ripper (Repeated Incremental Pruning to Produce Error Reduction) Rule is an efficient rule-based learning algorithm that can process various noisy datasets [22]. The Ripper Rule algorithm consists of two stages. The first stage is to initialize the rule conditions. The next stage uses a rule optimization technique. This step reprunes each rule of the rule set to minimize the errors. A training dataset consists of a growing set and a pruning set. When the Ripper Rules have already been trained by the training dataset, this algorithm checks the condition in each rule to classify the testing dataset. The rules are in if-then-else form. Each class is considered one by one until the condition in a rule is met, or the data is classified into a class.

3.2.3. Back-Propagation Neural Network

Back-Propagation Neural Network (BPNN) model is a well-known supervised learning model that can effectively classify many types of data. BPNN is a feed-forward multi-layer network. Its input vectors and the corresponding target vectors are used in training the network until it can approximate a function, and associate its input vectors with specific output vectors. BPNN has many possible training algorithms such as gradient descent, momentum and resilient algorithm.

3.2.4. Bayesian Network

A Bayesian Network involves both a graphical model and probabilistic model representing random variables and conditional independence through a directed acyclic graph. Nodes in the graph represent random variables, while edges represent conditional dependencies. Thus nodes that are not connected represent variables that are conditionally independent from each other. The Bayesian Network learns the casual relations between attributes and class labels from the training dataset before it can classify unknown data.

3.2.5. Naïve Bayesian classifier

Naïve Bayes is a simple technique for classification using a simple probabilistic model from Bayes's theorem with the assumptions of independent attributes. Naïve Bayes is a type of supervised learning algorithm that uses a maximum likelihood method for parameter estimation. It requires a set of training data to estimate means and variances of the attributes for classification. This technique works well with many complex real-world applications [20].

3.2.6. Radial Basis Function Neural Network

Radial Basis Function Neural Network (RBF-NN) has a feed-forward structure similar to the Back-Propagation Neural Network (BPNN). Both models have an input layer, a hidden layer and an output layer. RBF-NN uses radial basis functions as activation functions. They can be used for function approximation and time-series prediction. The RBF-NN is also similar to a K-Nearest Neighbor (k-NN) model, where data is classified by measuring Euclidean distances between input and the hidden layer centers. In dealing with large dataset, the RBF-NN usually uses much less computation time than a BPNN does [24].

4. Real-time IDS process and algorithm

In this section, we present our proposed real-time intrusion detection system and describe how it works. Our real-time IDS process, shown in Fig. 2, consists of three phases: the pre-processing phase, the classification phase, and the post-processing phase.

4.1. Preprocessing phase

In the pre-processing phase which is shown in the upper part of Fig. 2, we use a packet sniffer, which is built with Jpcap library [26], to extract network packet information including IP header, TCP header, UDP header, and ICMP header from each packet. After that, the packet information is partitioned by considering connections between every pair of IP addresses (source IP and destination IP) and formed into a record by aggregating information every 2 s. Each record consists of data features considered as the key signature features representing the main characteristics of network data and activities. We performed extensive experiments to find key features that define the signatures of normal vs. attack network traffic. We used information gain to select 12 essential features for our IDS approach. The features along with the data type and the information gain value are shown in Table 1a.

The information gain value of each feature represents the relevance of the feature to the output class. Parameters of information gain are X and Y, where X defines individual features such as number of TCP packets, number of TCP source ports, and Y defines class groups which are Normal data, Probe attack and DoS attack. Tables 1b and 1c present how each feature is relevant to the DoS attack, and the Probe attack, respectively. The results from information gain indicate that we have to consider all 12 features of the network data for the intrusion detection and classification.

Examples of the data records obtained from the pre-processing phase are shown in Table 2.

4.2. Classification phase

In the classification phase, we classify each of the preprocessed data records obtained from the preprocessing phase as normal data or attack data. This classification phase consists of two main processes which are training and testing network data. In the first phase, we train the selected machine learning algorithm as an IDS model by using a set of network records with known answer classes. Based on the trained IDS model, the classification technique can classify the data in each record into normal network activity or main attack types. Then, we test each model with new or untrained dataset where each record was captured in a real-time environment as shown in the classification phase in Fig. 2. Note that any of the different machine learning techniques as described in the previous section can be applied to the classification phase.

4.3. Post-processing phase

The post-processing phase is used to eliminate outliers or false-alarm detections from the result of classification. We propose to use a majority voting algorithm for every five consecutive detection results for each pair of IP Addresses (source and destination pair) to determine if the result is normal network activity or an attack type. To do this, we group the network data from the classification phase into groups of five records. In each group, if there are at least 3 out-of 5 records which are reported to be the same attack

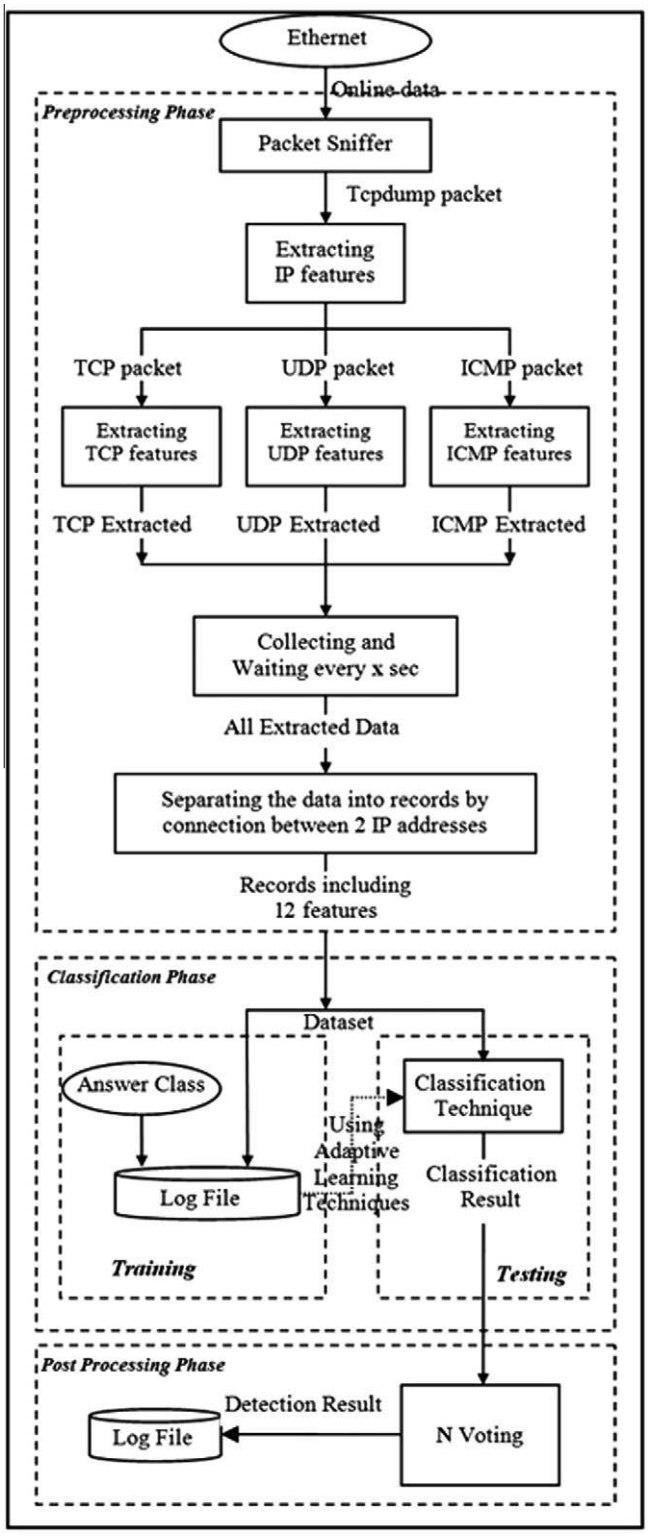

Fig. 2. RT-IDS process.

type, then this group of data is considered the attack type. Otherwise, the data is considered as normal network activity. This post-processing approach is performed for each pair of source–destination IP addresses. The post-processing phase can help eliminating false alarms from the system. Therefore, the IDS with the post-processing procedure can increase the detection accuracy and the user's confidence in the warning provided by the IDS.

Table 1a
Twelve essential features in the pre-processed data (normal, DoS, and probe).

No.	Feature description	Data type	Information gain
1	Number of TCP packets	Integer	0.626
2	Number of TCP source port	Integer	0.67
3	Number of TCP destination port	Integer	0.704
4	Number of TCP fin flag	Integer	0.177
5	Number of TCP syn flag	Integer	0.341
6	Number of TCP push flag	Integer	0.192
7	Number of TCP ack flag	Integer	0.407
8	Number of TCP urgent flag	Integer	0.155
9	Number of UDP packets	Integer	0.582
10	Number of UDP source port	Integer	0.202
11	Number of UDP destination port	Integer	0.247
12	Number of ICMP packets	Integer	0.159

Table 1b
Information gain values (normal vs. DoS).

No.	Feature description	Information gain
1	Number of TCP packets	0.355
2	Number of TCP source port	0.5071
3	Number of TCP destination port	0.3754
4	Number of TCP fin flag	0.0501
5	Number of TCP syn flag	0.21
6	Number of TCP push flag	0.2674
7	Number of TCP ack flag	0.3398
8	Number of TCP urgent flag	0
9	Number of UDP packets	0.5644
10	Number of UDP source port	0.2467
11	Number of UDP destination port	0.2458
12	Number of ICMP packets	0.1019

Table 1c
Information gain values (normal vs. probe).

No.	Feature description	Information gain
1	Number of TCP packets	0.48544
2	Number of TCP source port	0.35742
3	Number of TCP destination port	0.59803
4	Number of TCP fin flag	0.1076
5	Number of TCP syn flag	0.32282
6	Number of TCP push flag	0.02964
7	Number of TCP ack flag	0.26951
8	Number of TCP urgent flag	0.14922
9	Number of UDP packets	0.04131
10	Number of UDP source port	0.00393
11	Number of UDP destination port	0.08518
12	Number of ICMP packets	0.052

5. Experiments and performance evaluation

In this section, we present the experimental results and performance evaluation of our proposed real-time IDS. We first present the network data used in the experiment. We then describe our experimental design and performance metrics used for evaluating the real-time IDS. Finally, we present the experimental results.

5.1. Experimental data

Our experimental network data consists of four DoS attack types, 13 Probe attack types, and normal activity as presented in Table 3. All attack types were generated using many different tools as shown in the table, while the normal network data was captured from the actual network environment. We captured online network data from the Computer Engineering Department at King Mongkut's University of Technology Thonburi (KMUTT), Thailand. We call this dataset the RLD09 (Reliability Lab Data 2009) dataset.

Table 2
Data records from the pre-processing phase.

Source IP address	Destination IP address	Data of 12 features	Category
$a_1.a_2.a_3.a_4$	$a_5.a_6.a_7.a_8$	2138, 33, 33, 4, 4, 644, 2136, 0, 0, 0, 0, 0	Normal
$b_1.b_2.b_3.b_4$	$b_5.b_6.b_7.b_8$	12, 2, 2, 0, 0, 1, 12, 0, 0, 0, 0, 0	Normal
$c_1.c_2.c_3.c_4$	$c_5.c_6.c_7.c_8$	230, 2, 120, 0, 0, 0, 0, 0, 0, 0, 0, 0	Probe
$x_1.x_2.x_3.x_4$	$x_5.x_6.x_7.x_8$	6, 3, 1, 2, 2, 2, 2, 2, 153, 2, 78, 0	Probe
$y_1.y_2.y_3.y_4$	$y_5.y_6.y_7.y_8$	0, 0, 0, 0, 0, 0, 0, 0, 48810, 1, 1, 48810	DoS
$z_1.z_2.z_3.z_4$	$z_5.z_6.z_7.z_8$	145, 145, 1, 0, 145, 0, 0, 0, 0, 0, 0, 0	DoS

Table 3
Attack types and normal activity.

No.	Data	Tools (to Generate)	Category
1	Smurf	Smurf.c	DoS
2	UDP Flood	Net Tools 5	DoS
3	HTTP Flood	Net Tools 5	DoS
4	Jping	Jping.c	DoS
5	Port Scan	Net Tools 5	Probe
6	Advance Port Scan	Net Tools 5	Probe
7	Host Scan	Host Scan 1.6	Probe
8	Connect	NMapWin 1.3.1	Probe
9	SYN Stealth	NmapWin 1.3.1	Probe
10	FIN Stealth	NmapWin 1.3.1	Probe
11	UDP Scan	NmapWin 1.3.1	Probe
12	Null Scan	NmapWin 1.3.1	Probe
13	Xmas Tree	NmapWin 1.3.1	Probe
14	IP Scan	NmapWin 1.3.1	Probe
15	ACK Scan	NmapWin 1.3.1	Probe
16	Window Scan	NmapWin 1.3.1	Probe
17	RCP Scan	NmapWin 1.3.1	Probe
18	Normal	Actual environment	Normal

The dataset was captured during regular office hours for 4 days. The data consists of approximately 10 million preprocessed data packets and contains a total of 17 attack types plus normal network activity as shown in Table 3.

The attack classes can be divided into two main attack types which are Denial of Service (DoS) and Probe. A Denial-Of-Service (DoS) is an attack that attempts to interrupt network services by preventing the user from accessing the services. Moreover, this attack also causes the network to be unavailable for services by consuming the user's bandwidth. A DoS attack works by generating a stream of packets to overload the network. The DoS attacks that we generated include Smurf, UDP flood, HTTP flood, and Jping. User Datagram Protocol (UDP flood) is initiated by generating a large number of packets to random service ports so that the system is busy dealing with these packets that finally make the system unavailable for other services. HTTP flood targets at HTTP port by performing an attack that is difficult to distinguish from normal HTTP communication. This attack consumes the resources of the web server resulting in a degradation of response time. Smurf and Jping are used to flood on ICMP (Internet Control Message Protocol).

Probe attacks are techniques that are used to find available ports and services on the victim network for the hacker to break into. This technique works by sending a message to a port and waiting for a response message. Common probe techniques include the following.

- The Connect Scan is the simplest technique using the connect system call to a victim system. This method uses the TCP three-way hand shaking to connect with listening ports.
- The SYN Stealth works by sending the SYN packet to a victim network and waiting for a response. If the victim network sends the SYN/ACK back, it means this port is listening. Otherwise, the port is closed.
- The FIN Stealth sends TCP packets with the FIN flag set equal to one to the victim network. The attackers will wait for the RST response to determine which port is open.
- The Xmas Tree is used to determine the available ports by sending the URG, PSH flags and FIN in TCP header to a victim network.
- The Null Scan sends a TCP packet which has a sequence number instead of a flag to the targeted network and waits for a response from the open port.

5.2. Experimental design and performance metrics

We performed three experiments to evaluate our RT-IDS. In the first experiment, we considered off-line mode detection with known and unknown network data. We considered and compared various machine learning classification approaches. In the second experiment, we used on-line network data for real-time intrusion detection, with the best classification method obtained from the first experiment. In the last experiment, we performed a further detection analysis adding the post-processing procedure to improve the detection accuracy of our IDS.

In our experiments, we simultaneously generated attacks from many computers as shown in Fig. 3. We created four DoS attack types and 13 Probe attack types and sent the packets to a victim computer that hosted our RT-IDS. On the victim computer, many Internet services on Ethernet were engaged in order to generate normal network activity. Our RT-IDS was implemented on a 2.83 GHz Intel Pentium Core2 Quad 9550 processor with 4 GB RAM and 100Mbps LAN. The consumption of CPU and memory resource in our running RT-IDS system was captured by the Process Explorer tool [25].

We have adopted well-known machine learning techniques which have been developed as free-ware tools in our classification approaches. We use the Weka tool version 3.6.0 [23] for our IDS development.

We measured the detection performance of our RT-IDS as follows:

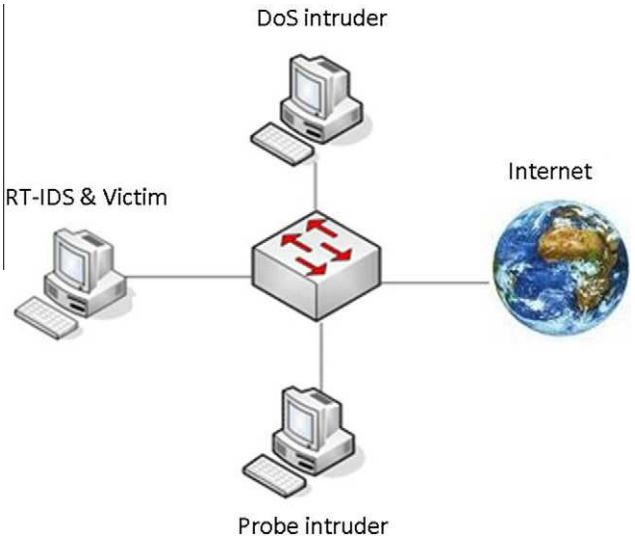

Fig. 3. Network testing environment.

- Total Detection Rate (TDR) is the percentage of DoS attacks, Probe attacks, and normal network data that the RT-IDS can correctly detect.
- Normal Detection Rate (NDR) is the percentage of the normal class that the RT-IDS can correctly detect.
- Attack Detection Rate (ADR) is the percentage of all attack classes that the RT-IDS can correctly detect.
- DoS Detection Rate (DDR) is the percentage of the DOS attacks that the RT-IDS can correctly detect.
- Probe Detection Rate (PDR) is the percentage of the Probe attacks that the RT-IDS can correctly detect.

5.3. Experimental results with off-line detection

We performed two experiments for off-line detection. The first experiment compared the well-known KDD99 dataset with our RLD09 dataset, which is captured in an actual network environment. Three well-known machine learning techniques were considered and detection results were evaluated. The second experiment investigated the detection performance with our RLD09 dataset using various machine learning techniques. These two experiments are intended to verify the validity of our dataset and to select the best classification approach for real-time detection which is conducted in the next step.

In the first experiment, we considered the 41-feature KDD99 dataset and 12-feature RLD09 dataset. Different datasets were used as training dataset and testing dataset. The KDD and the RLD09 training datasets each had 20,000 records consisting of 10,000 attack records and 10,000 normal records. The KDD and the RLD09 testing datasets each had 10,000 records consisting of 5000 attack records and 5000 normal records. The attack records in each dataset contain various types of DoS and Probe attacks.

We selected three techniques for classification which are Back-Propagation Neural Network, Decision Tree and Ripple Rule. Experimental results with the KDD99 dataset are presented in Table 4, showing all detection rates which are total detection, normal detection and attack detections. Most of the detection rates are over 99 percents meaning that the detection is highly accurate.

The experimental results with RLD09 dataset are presented in Table 5. All classification techniques gave total detection rates higher than 99% as well. The training time required for RLD09 is less than those of KDD99 dataset. This is expected since the RLD09 dataset uses far fewer features than the KDD99.The neural network approach uses more training time than the other two approaches. Among the three approaches, the C4.5 Decision Tree model for data classification required the least amount of training time in both experiments.

The results of our first experiment show that RLD09 provides comparable accuracy to KDD99, even though it uses much fewer features. In the second experiment, we sampled the RLD09 dataset into two groups consisting of a training dataset and a testing dataset.

We randomly chose 7200 records from 8 million records that were captured during 4-day experiment. The training dataset with 7200 records consisted of 2400 DoS records, 2400 Probe records and 2400 normal records. The testing dataset had 4800 records consisting of 1600 DoS records, 1600 Probe records and 1600 normal records. The results from this experiment are reported in Table 6.

Table 4
Results using the KDD99 input dataset.

Classification method	TDR (%)	NDR (%)	ADR (%)	Training time (s)
Back-Propagation NN	99.2	99.4	98.9	27.85
Decision Tree	99.6	99.6	99.6	0.95
Ripper Rule	99.5	99.5	99.5	4.44

Table 5
Results using the RLD09 input dataset.

Classification method	TDR(%)	NDR (%)	ADR (%)	Training time (s)
Back-Propagation NN	99.1	99.6	98.6	16.92
Decision Tree	99.4	99.7	99.0	0.28
Ripper Rule	99.3	99.7	98.9	1.5

Table 6
Off-line detection results from various classification methods.

Classification method	TDR (%)	NDR (%)	DDR (%)	PDR (%)	Training time (s)
Decision Tree	99.0	99.7	99.4	97.9	0.28
Ripper Rule	98.6	98.8	99.4	97.8	1.98
Back-Propagation NN	93.0	88.0	98.6	92.4	29.47
Bayesian Network	89.3	87.0	86.1	94.8	0.09
Naïve Bayes	78.7	98.7	78.2	59.1	0.05
Radial Basis Function NN	77.5	97.6	63.6	71.4	1.75

Various machine learning techniques were considered as classification techniques, and evaluated using the same input dataset. The detection rates that we considered were total detection rate (TDR), normal detection rate (NDR), DoS detection rate (DDR) and Probe detection rate (PDR). The table ranks the techniques in terms of these detection rates. The experimental results show that the Decision Tree technique is the best with outstanding detection rates, followed by the Ripper Rule. The detection results are not so good for Bayesian Network, Naïve Bayes and Radial Basis Function Neural Network. Therefore, we used the best model (the C4.5 Decision Tree) with highest detection rate for our real-time intrusion detection system (RT-IDS).

5.4. Experimental results with on-line detection (real-time IDS)

We implemented the C4.5 Decision Tree technique in our IDS for on-line detection. In the experiment, the training data had 55,000 records including 10,000 DoS records, 30,000 Probe records and 15,000 normal records. After being trained with the training data, the C4.5 Decision Tree model consists of 197 non-terminal nodes (a root and internal nodes) and 99 terminal nodes (leaves). The RT-IDS with the trained Decision Tree was used to test with on-line network data over a 24-h time interval, capturing about 109 million connections. After preprocessing the testing data, we obtained a total of 1,02,959 records of testing data, consisting of 19,454 DoS records, 8392 Probe records, and 75,113 Normal records, to feed into the classification part of the real-time system or the RT-IDS. The corresponding results are shown in Tables 7 and 8.

Table 7 shows that the total detection rates for DoS attack, Probe attack and normal network activity were 99.17%, 98.73% and 99.43%, respectively. The DoS attack was 0.76% misclassified as Probe, and 0.07% misclassified as normal data. The Probe attack was 1.27 % misclassified as DoS or else normal data. The normal network data was misclassified 0.08% as DoS and 0.47% as Probe, leading to a false-alarm rate of 0.53%. A false-alarm is a situation where normal data is classified as attack.

The detection rates are summarized in Table 8 for total detection rate (TDR), normal detection rate (NDR), DoS detection rate (DDR) and Probe detection rate (PDR), respectively.

We also measure the detection speed of our RT-IDS by using the operating system clock. The detection time was approximately 2 s per record. This amount of time is mainly used for data preprocessing, which takes about 2 s because we aggregate network packets

Table 7
On-line detection confusion matrix.

Actual	Prediction		
	DoS (%)	Probe (%)	Normal (%)
DoS	99.17	0.76	0.07
Probe	0.07	98.73	1.2
Normal	0.08	0.47	99.43

Table 8
On-line performance evaluation.

Our RT-IDS	TDR (%)	NDR (%)	DDR (%)	PDR (%)
Decision Tree	99.33	99.43	99.17	98.73

in two second interval. The data classification consumes only a few milliseconds.

While the RT-IDS was running, we used the Process Explorer tool [25] to capture the consumption of CPU capacity and memory resources. These data are shown in Figs. 4 and 5. When running with full load traffic (100 Mbps), our RT-IDS used less than 25% of CPU resources while consuming about 94.5 MB of memory.

5.5. Experimental results with post-processing procedure

In the post-processing procedure, we use a majority voting algorithm for each pair of IP addresses (source and destination pair) to determine if the result is normal network activity or an attack type. When the classification phase identifies an attack type in at least 3 out of 5 contiguous records, the attack is acknowledged and reported. Otherwise, the records are classified as normal network activity.

The results of our IDS with post-processing and without the post-processing procedure are compared in detail as shown in Table 9. With the post-processing procedure, five consecutive records are grouped into one record if they all have the same identification results from the IDS. Thus, the number of records is reduced when the post-processing is considered. The number of normal data records is reduced from 65,536 records down to 24,085 groups of records. The number of records with false detection is also grouped and reduced from 384 down to 141 for the normal data, as shown in Table 9. Since the data records are collected from many source–destination IP pairs, the last remaining records of each IP pair may have fewer than 5 records grouped together.

With the post-processing process, only 5 out of 24,084 groups of data are reported as false detection. On the other hand, without considering majority voting in the post process procedure, 141 out of 24,084 groups of data would be reported as fault detection. Therefore, this post processing process can greatly reduce the fault detection rate from (141 out of 24,084) down to (5 out of 24,084). As shown in Table 9, the normal detection rate is enhanced with the post-processing procedure, reaching a 99.979% normal detection rate.

The DoS and the Probe detection rates can also be greatly improved with the post processing procedure. These are shown in the table as the number of false positives which is reduced from 30 (with record grouping of 5) down to 22, and 21 down to 19, respectively. Consequently, the DoS and Probe detection rates are enhanced with the post-processing procedure, reaching to 99.434% and 98.868%, respectively.

In summary, our experimental results showed that our RT-IDS can offer a total detection rate (TDR) higher than 99%, with a false alarm rate that is very low (less than 0.5%). When capturing network traffic with full load (100 Mbps), our RT-IDS consumes less than 25% of CPU resources while using only 94.5 MB of memory. This does not greatly affect the PC performance when other applications are also running. Our RT-IDS can quickly detect malicious data packets within 2–3 s which is sufficient to alert the computer user or network administrator for network protection. Essentially the Decision Tree algorithm is suitable for the real-time intrusion detection. Finally, the results with our post-processing approach show that we can improve the detection rates of normal network activities, DoS attacks and the Probe attacks, while reducing the false detection rates for all record types. Therefore, our real-time detection is effective and more reliable with the post-processing approach.

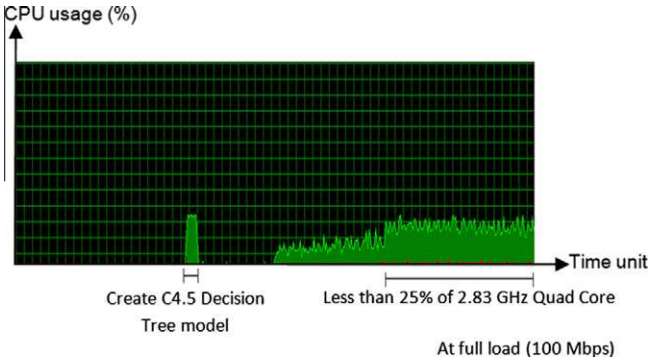

Fig. 4. CPU consumption of the RT-IDS.

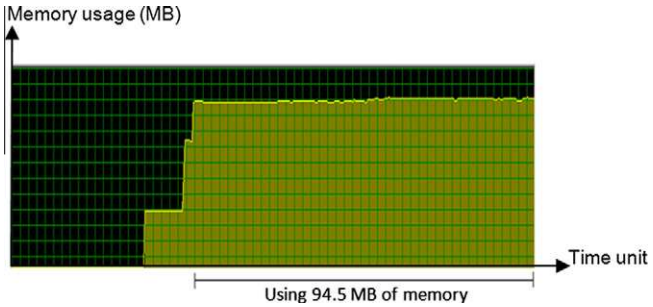

Fig. 5. Memory consumption of the RT-IDS.

Table 9
Results of detection with/without post-processing (pp) procedure.

	Normal		DoS		Probe	
	Without pp.	With pp.	Without pp.	With pp.	Without pp.	With pp.
Number of records	65536	24084	19446	3892	8392	1679
False detection	384	5 from 141	152	22 from 30	103	19 from 21
Detection rate	99.414	99.979	99.218	99.434	98.736	98.868

6. Conclusions

In this paper, we presented a practical and efficient real-time network intrusion detection system (RT-IDS) model which can be used with existing well-known machine learning algorithms. Our RT-IDS model consists of three phases: the pre-processing phase, the classification phase, and the post-processing phase. We also presented how we preprocess the network packet header data into records of 12 essential features. The essential features were identi-

fied with the information gain method, to see how each of them is relevant to the network attack types.

We considered and evaluated various machine learning algorithms which are Decision Tree, Ripper Rule, Back-Propagation Neural Network, Radial Basis Function Neural Network, Bayesian Network, and Naïve Bayesian for designing the intrusion detection system. The experimental results showed that the Decision Tree gave higher total detection rates than the other algorithms. Thus, we developed a new real-time IDS using the Decision Tree technique.

The performance evaluation results showed that our real-time IDS is efficient in terms of real-time detection speed and consumption of CPU and memory. It takes only 2 s to detect and classify the incoming network data. The classification results can be improved with an optional data post-processing procedure, which can additionally reduce the false alarm rate.

Our research work has several contributions as follows. (1) We present an uncomplicated IDS model which can be easily applied with existing machine learning techniques. No human expert is needed to visualize or identify the attack. (2) We propose 12 essential features which are relevant to DoS and Probe attacks. This small number of features can significantly improve the on-line (real-time) IDS detection speed and consumption of computer resources. (3) We present a practical real-time, network-based IDS that not only can efficiently detect but also can classify network data into three categories which are normal, Denial of Service (DoS) and Port Scanning (Probe). (4) We develop a post-processing procedure for reducing false alarm rate.

In future work, we plan to implement a combination technique for misuse detection and anomaly detection with the proposed 12 relevant features in order to better detect unknown attack types.

Acknowledgement

This work was supported by King Mongkut's University of Technology Thonburi, the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program (Grant No. PHD/0016/2551), the Higher Education Research Promotion and National Research University Project of Thailand, Office of the Higher Education Commission.

References

[1] C. Jirapummin, N. Wattanapongsakorn, J. Kanthamanon, Hybrid neural networks for intrusion detection system, in: The International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Thailand, 2002, pp. 928–931.
[2] W. Lee, S. Stolfo, K. Mok, Mining in a data-flow environment: experience in network intrusion detection, in: The 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '99), San Diego, 1999.
[3] Z. Pan, S. Chen, G. Hu, D. Zhang, Hybrid neural network and C4.5 for misuse detection, in: The 2nd International Conference on Machine Learning and Cybernetics, China, 2003, pp.2463–2467.
[4] M. Moradi, M. Zulkernine, A neural network based system for intrusion detection and classification of attacks, in: The IEEE International Conference on Advances in Intelligent Systems Theory and Applications, Luxembourg, 2004, pp. 148–153.
[5] N. Ngamwitthayanon, N. Wattanapongsakorn, C. Charnsripinyo, D.W. Coit, Multi-stage network-based intrusion detection system using back propagation neural networks, in: Asian International Workshop on Advanced Reliability Modeling (AIWARM), Taiwan, 2008, pp. 609–619.
[6] A. Abraham, R. Jain, Soft computing models for network intrusion detection systems, in: Knowledge Discovery, Computational Intelligence, vol. 4, Heidelberg, 2005, pp. 191–207.
[7] N. Liao, S. Tian, T. Wang, Network forensics based on fuzzy logic and expert system, Computer Communications 32 (2009) 1881–1892.
[8] N. Ngamwitthayanon, N. Wattanapongsakorn, D.W. Coit, Investigation of fuzzy adaptive resonance theory in network anomaly intrusion detection, in: The IEEE International Symposium on Neural Networks, China, 2009.
[9] A.N. Toosi, M. Kahani, A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers, Computer Communications 20 (2007) 2201–2212.
[10] C-H. Tsang, S. Kwong, H. Wang, Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection, Pattern Recognition 50 (2007) 2373–2391.
[11] S. Pukkawanna, V. Visoottiviseth, P. Pongpaibool, Lightweight detection of DoS attacks, in: The IEEE International Conference on Networks (ICON), Australia, 2007, pp 77–82.
[12] V. Katos, Network intrusion detection: evaluating cluster, discriminant, and logit analysis, Information Sciences: an International Journal 177 (15) (2007) 3060–3073.
[13] C-M. Chen, Y-L. Chen, H-C. Lin, An efficient network intrusion detection, Computer Communications 33 (2010) 477–484.
[14] K. Labib, R. Vemuri, NSOM: a real-time network-based intrusion detection system using self-organizing maps, Networks and Security (2002).
[15] R.S. Puttini, Z. Marrakchi, L. Me, A Bayesian classification model for real-time intrusion detection, in: The 22nd International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering. AIP Conference Proceedings, vol. 659, 2003, pp. 150–162.
[16] M. Amini, A. Jalili, H. Reza Shahriari, RT-UNNID: a practical solution to real-time network-based intrusion detection using unsupervised neural networks, Computer & Security 25 (2005) 459–468.
[17] M-Y. Su, G-J. Yu, C-Y. Lin, A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach, Computers and Security 28 (2009) 301–309.
[18] Z. Li, Y. Gao, Y. Chen, HiFIND: a high-speed flow-level intrusion detection approach with DoS resiliency, Computer Networks 54 (2010) 1282–1299.
[19] S. Chakrabarti, M. Chakraborty, I. Mukhopadhyay, Study of snort-based IDS, in: International Conference and Workshop on Emerging Trends in Technology (ICWET), Mumbai, India, 2010, pp. 43–47.
[20] M. Panda, M.R. Patra, Semi-Naïve Bayesian method for network intrusion detection system, Neural information processing, Lecture Notes in Computer Science (Springer Link) 5863 (2009) 614–621.
[21] P. Sangkatsanee, N. Wattanapongsakorn, C. Charnsripinyo, Network intrusion detection with artificial neural network, decision tree and rule based approaches, in: The International Joint Conference on Computer Science and Software Engineering, Thailand, 2009.
[22] P.N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Pearson Addison Wesley, 2005.
[23] Weka 3.6.0 tools [Online]. <http://www.cs.waikato.ac.nz/ml/weka/>.
[24] S.X. Wu, W. Banzhaf, The use of computational intelligence in intrusion detection system: a review, Applied Soft Computing 10 (2010) 1–35.
[25] Process Explorer tool [Online]. <http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx>.
[26] Jpcap library [Online]. <http://jpcap.sourceforge.net/>.
[27] J.H. Lee, J.H. Lee, S.G. Sohn, J.H. Ryu, T.H. Chung, Effective value of decision tree with KDD 99 intrusion detection datasets for intrusion detection system, in: International Conference on Advanced Communication Technology (ICACT), Korea, 2008.