Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 647216, 24 pages
http://dx.doi.org/10.1155/2014/647216

Research Article

Signal Waveform Detection with Statistical Automaton for Internet and Web Service Streaming

Kuo-Kun Tseng,[1] Yuzhu Ji,[1] Yiming Liu,[1] Nai-Lun Huang,[2] Fufu Zeng,[1] and Fang-Ying Lin[3]

[1] *Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, China*
[2] *Chunghwa Telecom Laboratories, Yangmei, Taoyuan 32601, Taiwan*
[3] *Bartlett School, University College London, London WC1E 6BT, UK*

Correspondence should be addressed to Fang-Ying Lin; regina.lin.09@ucl.ac.uk

Received 6 December 2013; Accepted 19 February 2014; Published 16 June 2014

Academic Editors: Z. Chen and F. Yu

Copyright © 2014 Kuo-Kun Tseng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In recent years, many approaches have been suggested for Internet and web streaming detection. In this paper, we propose an approach to signal waveform detection for Internet and web streaming, with novel statistical automatons. The system records network connections over a period of time to form a signal waveform and compute suspicious characteristics of the waveform. Network streaming according to these selected waveform features by our newly designed Aho-Corasick (AC) automatons can be classified. We developed two versions, that is, basic AC and advanced AC-histogram waveform automata, and conducted comprehensive experimentation. The results confirm that our approach is feasible and suitable for deployment.

1. Introduction

With the development of the Internet and web service computing and as more transactions are carried out online [1, 2], there is greater vulnerability to network attacks. Thus, Internet and web service content analyses have become important pieces of technology to protect our network security. Such systems collect information through a number of key points in the computer network or system and analyze it to determine whether there are the violations of the security policy or intrusion phenomenon.

Conventional packet analysis systems used packet classification and deep packet inspection, which may not be sufficient to detect all kinds of network service. Therefore, we have tried a new approach to analyze the network packets. We propose a new signal waveform based on deep packet classification. In this paper, we review some previous researches and discuss the existed approaches of packet analysis and then propose new analysis architecture based on a mathematical statistical approach. Compared with conventional methods, the network data within a time period is recorded to obtain the training waveform. The attention is on the statistical data of the waveform features, and we use their feature extractions as input to the classifiers.

Packet classification can be divided into two parts, training and testing. The training part is used to establish patterns and behavioural rules, and the testing part is for the classification and determination of the samples. In the training stage, data frame length information is extracted, and the integer of the length information of the data frame converted to a string. Then a certain number of characters are taken as a unit of the recording pattern and the pattern mode and its number of occurrences will be counted (the pattern retreats one packet each time). Through the training process, the system determines the signal waveform and its number of occurrences. Then, the same transform is carried out on other data, by loading the test string data into the Aho-Corasick (AC) automaton for matching, obtaining a match situation, and then we make a comparison with standard data. We use their differences as the judgment basis.

On the other hand, recent intrusion detection systems commonly use two detection technologies, namely, misuse detection and anomaly detection. Anomaly intrusion detection is a behaviour-based detection regime. Its basic

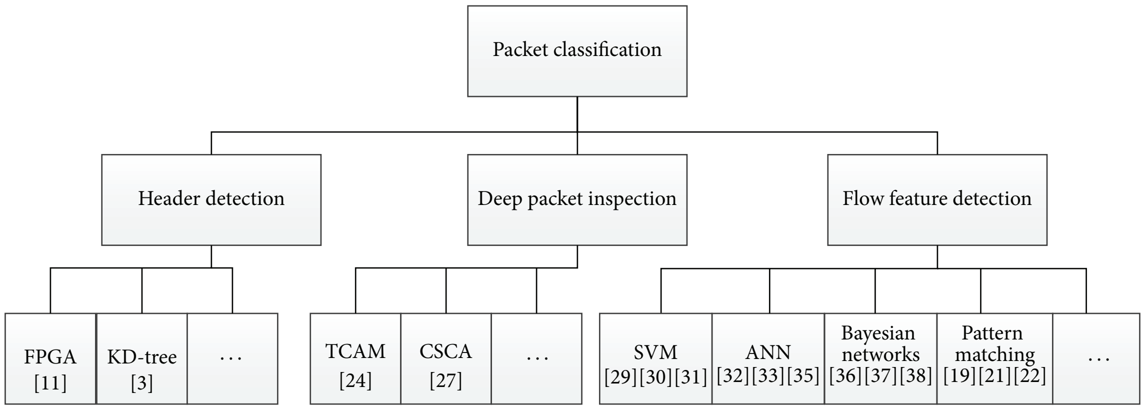


Figure 1: Categories of packet classification algorithms.

principle is to determine whether a deviation from the so-called normal behaviour contour exists by comparing the current system and user's behaviour with the established system or user's normal behaviour contours, which can determine whether it is abnormal (intrusion) behavior; this is an indirect detection method. Misuse intrusion detection has also been referred to as knowledge-based detection, the principle of which is conducting attack feature extraction with known attack methods to constitute a signature database then processing pattern matching between the audit data and the feature library to determine the intrusion behaviour. This approach, which is based on the principles of the above two intrusion detection methods, consolidates the user's behaviour and methods of attacks and will put forward new intrusion detection principles for the whitelist and blacklist of network's behavior. That will enhance the efficiency of traditional anomaly detection and misuse detection methods.

The method applied in this paper is similar to misuse detection, in that the characteristics and rules of the packets are obtained for different kinds of applications, which will form the basis of the classification. This paper puts forward a new idea for packet detection by calculating the features of the network packet, which means that it draws on the allied technology of packet classification and uses these technologies as its foundation. However, this topic uses a certain flow of data packets as the object of the analysis and processing and extracts and calculates the signal waveform information of the characteristics to classify the data packets, which is different from the majority of packet classification algorithms based on a single or data connection packet. Furthermore, it implements a network-based real-time packet detection system, to carry out the detection and analysis of the network behaviour of the user. This paper with calculation and matching of the overall data packets flowing through the network can get more accurate grasp and analysis of user behaviour, as well as achieving the purpose of detection of user behaviour of network applications. At the same time, the algorithm can also be used for the detection of some forms of intrusion. In addition, through appropriate training, it can also be implemented in user identification and system safety rating.

In this paper, some related knowledge is introduced in Section 2, and our system architecture and proposed approaches are presented in Sections 3 and 4. In Section 5, we introduce some advanced analysis of this issue. And we have conducted some preliminary assessment of the approach, which will be described in Section 6. In Section 7, we conclude the paper and offer a look towards planned future work.

2. Background and Related Work

As effective means of network monitoring and traffic control, network packet classification technology has developed by leaps and bounds since its birth. At the same time, various classification methods and means have continually emerged, which have been used in a wide range of network packet inspection methods. In this section, an introduction and analysis of several classic packet classifications are given, as well as a brief summary of the research situation regarding the packet classification algorithms.

According to the different division standards, there are several different classifications of intrusion detection systems (see Figure 1).

Recent packet classification technology mainly includes Deep Packet Inspection method (DPI), Stateful Packet Inspection Method (SPI) (header packet classification), and inspections based on the flow characteristics of known data sets, such as support vector machines (SVMs) for classification and the artificial neural network classification algorithm, as well as the pattern matching classification algorithm.

DPI technology generally utilizes the packet payload to identify applications. DPI checks not only the header information, but also the deep packet payload content to determine the application layer protocol. DPI technology utilizes exact string on behalf of the network application mode or regular expressions for matching with the data packet payload, so that it can be determined whether the detected pattern is present in the data packet payload.

SPI is designed to prevent harmful or unsolicited packets entering the computer. SPI was originally developed for firewalls, but so far it has become an important factor in network-based intrusion detection systems (NIDS) and many other applications [3].

The so-called "state" denotes "things before recording." Essentially, SPI is a dynamic packet filter working in a network layer. It checks each IP packet to obtain the necessary information and then the tracking network connection state, thus allowing or denying the subsequent traffic through the firewall. SPI gradually replaced the static packet filter and became an industry standard for network firewall solutions.

SPI tracks network connection states (such as TCP streams or UDP communications) and saves the important attributes of each connection into its memory. These attributes are collectively referred to as the state of the connection, which may include the IP address, port number, and serial number. State inspection monitors packet data and the connection status of incoming and outgoing packets and stores them into a dynamic state table. The accumulated data will be evaluated, and filtering decisions are based not only on predefined rules set by the administrator, but also on the configuration established by previous connections and packets belonging to similar connections.

On the other hand, many scholars have designed packet classifications and algorithms according to the known data set, such as the classification algorithm based on support vector machines (SVM-based) and artificial neural networks (ANN-based), as well as classification algorithm based on Bayesian network, calculation of pattern matching, and so forth.

The SVM is a supervised learning method that can be used for classification and regression analysis. It will map the training vectors to a high dimensional feature space, and each vector will be marked by its category. The SVM treats the classification as a quadratic optimization problem. The combination of SVM and generalization control technology, which can avoid the "curse of dimensionality" by setting the maximum interval between the different categories, makes it a practical tool in processing large number of dynamic collected data. By defining a series of SVMs, SVM constructs separating hyperplanes in the feature space to conduct data classification [4].

In the past, several research groups have proposed a variety of intrusion detection methods based on SVM technology. Fugate and Gattiker [5] proposed an anomaly detection method using SVM. Kim and Park [6] proposed a method of applying SVMs to network-based intrusion detection systems. Xie et al. [7] proposed a feature extraction algorithm using a factor analysis (FA) and support vector decision function ranking method (SVDFRM). Mukkamala et al. [8] described approaches to intrusion detection using SVMs. They built efficient and accurate classifiers to detect intrusions. Hu et al. [9] have proposed a host-based anomaly detection method using RSVM. Kim et al. [10] proposed a fusion of a genetic algorithm (GA) and SVM for anomaly detection in an intrusion detection system (IDS).

Artificial neural networks (ANN), also called neural networks, are a mathematical or computational model that imitates the structure and function of a biological neural network. The neural network is constitutive of interconnected artificial neurons, which uses the link method to calculate and process information. In most instances, the neural network is an adaptive system, which means that it can change its structure in the learning stage. So far, ANNs have been widely applied in the field of intrusion detection. Cannady [11] presented an approach to the process of misuse detection that utilizes the analytical strengths of neural network. Lippmann and Cunningham [12] improved high false-alarm rates by using a combination of discriminative training and generic keywords. Bivens et al. [13] used a time-window method. The method allows us to generalize input further than Cannady's method, enabling us to recognize longer multipacket attacks. They collect and monitor network traffic trends. Lorenzo-Fonseca et al. [14] proposed a sort of compromise between both ends of the scale: a model based on principal component analysis (PCA) as the chosen algorithm for reducing characteristics in order to maintain efficiency without hindering the capacity of detection. Moradi and Zulkernine [15] aimed to solve a multiclass problem in which the type of attack is also detected by the neural network. Wang et al. [16] proposed a novel approach for ANN-based IDS, the FC-ANN, to enhance detection precision for low-frequent attacks and detection stability.

The Bayesian network, also called evidential network, is an expansion of the Bayesian method and one of the most efficient models in the uncertain knowledge and reasoning field. It can be used to express and analyze uncertainty and probability events and conditionally relies on decision making with a variety of control factors, conducting reasoning from incomplete, imprecise, or uncertain knowledge or information [17, 18].

Similarly, there are many studies on Bayesian-based IDS. Bulatovic and Velasevic [19] suggested a Bayesian alarm network to work as an independent network intrusion detection agent. Kruegel et al. [20] utilized a Bayesian decision process to classify input events. Burroughs et al. [21] used Bayesian multiple hypothesis tracking as a basis for identifying the activities of individual attackers as they move across many networks. They improved the understanding of the attackers' behaviour by using existing data being gathered from distributed IDS. Cemerlic et al. [22] proposed an adaptive network intrusion detection system using a Bayesian network, trained with a mixed dataset containing real-world and DARPA dataset traffic.

At present, the use of a pattern matching method to detect intrusion is gradually becoming the mainstream of intrusion detection systems. As the core of pattern matching methods, string matching algorithms have been researched and applied widely. With the present development of pattern matching algorithm, there are mainly two research directions, single mode and multimode. Classic pattern matching algorithms contain the KMP algorithm, BM algorithm, AC algorithm, and so on.

The KMP algorithm [23] is proposed by Knuth et al.. The algorithm is based on prefix matching. The KMP algorithm matching principle is an approach run within the matching window from the testing text left to right to match. Inspired by KMP algorithm, Boyer and Moore [24] proposed a new fast string matching algorithm: the BM algorithm. The

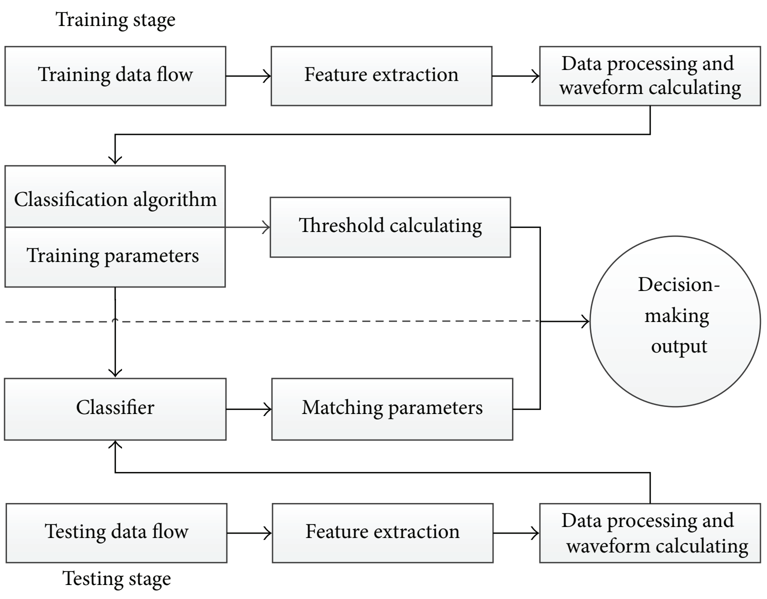


FIGURE 2: System framework.

BM algorithm is a suffix-based matching algorithm, which considers the position of the characters that may appear in the string pattern and can glide the body dramatically.

The main idea of the BM algorithm is to conduct the matching comparison from the left position i in the text data with patterns from the right to left. At the beginning, the pattern is aligned with the text in the leftmost side. When mismatching occurs, the next match position should be $i + \text{dist}[\rho_i]$ in the text, which is equivalent to right sliding of the pattern by a distance of $\text{dist}[\rho_i]$ and a skip of $\text{dist}[\rho_i]$ characters without comparison. If the character does not appear in the pattern, or appears at the end of the pattern, then the slide to the right would be with the greatest distance of m.

The Aho-Corasick (AC) algorithm [25] can be regarded as an extension of the KMP algorithm for the problem of multipattern matching. The AC algorithm uses a TRIE data structure to organize all the pattern strings. It reduces the cycle the strings require in all modes within each matching window match within the window of the TRIE data structure. The AC algorithm constructs a dendriform finite state machine, of which the different internal nodes are interconnected. These connections enable the failed match to quickly jump between the other branches of the tree which share a common prefix. Therefore this also means that the automata do not need to backtrack during the transition of pattern matching.

3. System Architecture and Detection Algorithm Process

This paper proposes a signal waveform packet classification algorithm and on this basis designs a real-time network packet detection system. The necessary raw data can be obtained by real-time capturing network packets and then the final purpose of matching and classification accomplished after system processing and computation. Through proper training the system can detect the behaviour of the network applications of users. The design of the framework of this packet inspection system is shown in Figure 2.

As with the conventional detection system, the system can be divided into two parts: training part and testing part.

The training part is used to establish the classification model of the system for detection. After data processing, the statistical waveform features will be extracted and classifier rules will be established, and then the decision threshold should be designed.

The testing part will monitor the real-time data traffic and extract feature information of the message, which will be fed to the classifier for matching. According to the matching extent of the testing data and the default data of the classifier, as well as the training threshold value, the system will conduct the final decision making, to achieve the purpose of monitoring the network behaviour.

The system implemented by this paper is constructed in an environment established by ourselves. The procedure of the acquisition and analysis of the data at the gateway can be divided into five steps.

(1) Capture data packets. Use *libpcap* to capture network packets on a Linux platform, and record the content of the data packets.

(2) Feature extraction and preprocessing. Extract the feature information of the system from the network packet header which is required for the matching judgment, and then conduct operation and quantization processing according to the different characteristics and obtain the required feature values of this issue.

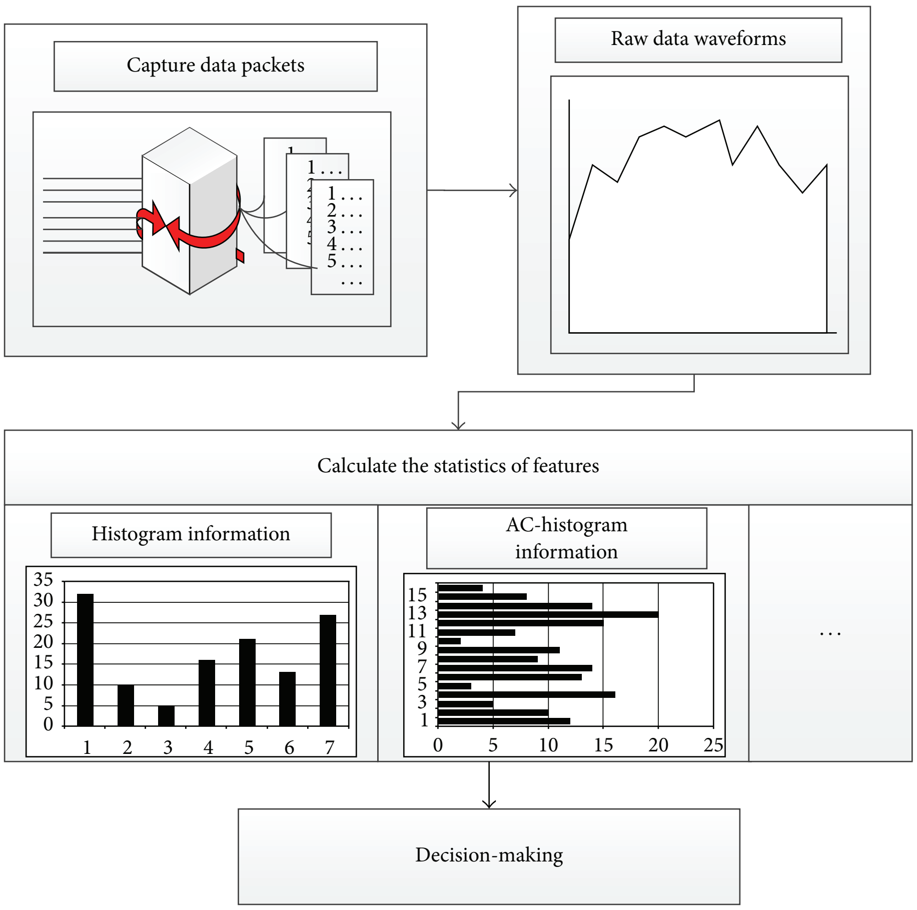


Figure 3: Implementation of matching algorithm.

(3) Calculation of data statistics. After obtaining a certain amount of network traffic data frames, record the waveform of the feature information. Then calculate the appropriate amount of statistical data according to the digital characteristics of the feature information, which will be used as input to the classifier.

(4) Classifier determination. Formulate classification strategies according to the various features and statistics collected, and design different classifiers for the matching of various inputs. Decision-making is to determine the testing data by training and experimentation, so that the analysis of network packets can be implemented.

(5) Results and output. Generate users' network behaviour logs, summarize the behaviour of the gateway network application, and then output the results.

Through the several steps above, the system can implement the real-time monitoring of network traffic flowing through the established environment and enable the identification of the user network application behaviour more effectively.

The core detection contents of the system are the classification and matching of packet patterns. The specific process of the packet classification algorithms designed in this paper and related algorithms are shown in Figure 3.

From the beginning of the collection of the data to the final output of the decision-making, in this paper, a variety of algorithms are implemented for data conversion, calculation, and matching. The following section will give a detailed description and analysis of the algorithms from several aspects, such as the original data processing and the process of pattern matching.

4. Proposed Algorithm

4.1. Original AC Algorithm. In the analysis and matching of specific network behaviour, AC automaton is utilized. The algorithm was developed in Bell Labs in 1975 and is one of the most famous multimode matching algorithms.

The basic idea of the algorithm is as follows. In the preprocessing stage, the AC automaton algorithm will establish three functions: a steering function which is defined as *goto*, a failure function named *fail*, and an output function *output*, which constructs a tree-type finite automaton. In the searching stage, it firstly locates all the occurrence positions of the keyword in the text by scanning the text through an intersecting utilization of the three functions.

This algorithm has two features: one is that during the process of scanning the text completely it does not need to backtrack, and the other is that the time complexity will be $O(n)$, and the time complexity has nothing to do with the number or length of the keyword.

The implementation of the AC multipattern matching algorithm can be divided into two stages: preprocessing and matching. In the preprocessing stage, a finite state machine is generated according to the string pattern to be matched; in the matching stage the state machine will conduct a state transition according to the input text string. When a certain state is reached, if the status has a matching pattern string, then the match succeeds. The AC state machine includes the three functions of *goto*, *fail*, and *output*.

(*A*) *goto Function*. The AC-matching algorithm, which takes character as its processing unit, hypothesizes that the string group P contains n patterns, and state machine will be generated by the use of pattern string groups consisting of five steps (shown below).

(1) If $n \leq 0$, return error; $i = 0$;

(2) if $i \leq 0$, take out the string pattern P_i and set the current state $S = 0$; otherwise, end the generation stage;

(3) take the next character C of the P_i, if C exists, then $S = goto(S,C)$; otherwise, jump to step (2);

(4) if $S! = -1$ (-1 denote null state), jump to step (3);

(5) $goto(S,C) = NewState$ (NewState: generate a new state and jump to step (3)).

(*B*) *fail Function*

(1) Define the shortest length of the path from state 0 to the state S as the depth of state S;

(2) set all states whose depth is 1 to $fail(s) = 0$; calculate the value of fail from low to high according to depth;

(3) the fail value of the depth d state is calculated by state r whose depth is $d - 1$;

(4) set $state = fail(r, a)$;

(5) execute $state = fail(state)$ zero to several times until $goto(state, a)! = Fail$;

(6) set $fail(s) = goto(state, a)$.

(*C*) *output Function*

(1) During the calculation of the *goto* function, after the completion of a mode string added into the state tree, add the current mode string as the output for the last state of that mode string;

(2) during the calculation of the *fail* function, when the calculated r equals $fail(s)$, add the output mode string of r to the state of s.

4.2. Proposed Basic Approach: AC Waveform. The flow of the AC waveform method is as follows.

The system processing of the AC Waveform is described by the following notations: $\text{App}^{(i)}$ denotes the ith application and $S_T^{(i)}$ is ith training stream, label T means the stream is for training, and $\text{App}_{S_T}^{(i)}$ is used to represent the true application of the training stream, where $i = 1, 2, 3, \ldots$ is the serial number of the training applications. Thus, consider $(S_T^{(i)}, \text{App}_{S_T}^{(i)})$ as a training sample labeled by $\text{App}_{S_T}^{(i)}$, which will be the input of the system in the training stage. (Note: in the following description, without special instruction, the symbol := means the assignment operation and the notation = represents truth assertion, that is, the comparison operation between the left and right sides of the value, and $Value\{condition(s)\}$ means that the value can only be obtained provided the condition(s) inside the brackets is(are) met.)

Training stage. First, a preprocessing procedure is conducted on the data stream which will be denoted by the following equation:

$$\left(L_T^{(i)}, \text{App}_{S_T}^{(i)}\right) := \Gamma_e\left(S_T^{(i)}, \text{App}_{S_T}^{(i)}\right), \tag{1}$$

where $L_T^{(i)}$ represents the extracted frame length information.

The next step will be the waveform generation process, which can be demonstrated by the following equation:

$$\omega_T^{(i)} := \Lambda_\omega\left(L_T^{(i)}, \text{App}_{S_T}^{(i)}\right). \tag{2}$$

Consider $\omega_T^{(i)}$ as the generated waveform by conducting the process of Λ_ω function with the frame length information $L_T^{(i)}$ extracted in the last step. Then it converts the waveform information into short patterns, and we use the Δ_c function to represent this process; the string segmentation strategy involved will be described in detail in the following sections. Here, the following equation is simply used to illustrate this procedure, regardless of the implementation details of the process:

$$\delta\left[P_m\right]_T^{(i)} := \Delta_c\left(\omega_T^{(i)}\right), \tag{3}$$

where $\delta[P_m]_T^{(i)}$ denotes a string array of a group of short patterns with label $\text{App}_{S_T}^{(i)}$ generated in the Δ_c process and P_m is the mth short pattern with $m = 1, 2, 3, \ldots$.

Then, we feed $\delta[P_m]_T^{(i)}$ into the system to establish AC automaton $\phi_T^{(i)}$, which can be expressed by

$$\phi_T^{(i)} := \Phi_e\left(\delta\left[P_m\right]_T^{(i)}\right). \tag{4}$$

Similarly, in the test stage, $S_D^{(j)}$ is used to denote the testing data stream, where the subscript of label D signifies that the stream is for detection. Then, the testing data stream will be fed into the testing system. The same data preprocessing steps will be conducted, including frame length information extraction and waveform establishment (see (1) and (2)); we will conduct a data conversion procedure, which will generate the short testing patterns. Similarly, the process can be denoted by the following equation:

$$\delta\left[P_n\right]_D^{(j)} := \Delta_c\left(\omega_D^{(j)}\right). \tag{5}$$

Then, the data is input into the established AC automaton in the training stage, and the total number of matches will be counted:

$$\mathscr{C}_{S_D^{(j)}}^{(i)} := \sum_n \mathbf{1}\left\{M_c\left(\delta\left[P_n\right]_D^{(j)}, \phi_T^{(i)}\right)\right\}, \tag{6}$$

where the *Matching* function refers to the matching process after the data is fed to the AC automaton. $MR_{(j)}^{(i)}$ is used to represent the decision-making result by comparing the counted result with the set threshold, which can be denoted by

$$\mathbb{M}_{(j)}^{(i)} := \begin{cases} \mathbf{1}, & \mathscr{C}_{S_D^{(j)}}^{(i)} > \text{threshold} \\ \mathbf{0}, & \mathscr{C}_{S_D^{(j)}}^{(i)} \le \text{threshold} \end{cases} \tag{7}$$
$$(i = 1, 2, 3, \ldots; j = 1, 2, 3, \ldots),$$

where the value of $\{\mathbf{1}, \mathbf{0}\}$ refers to a testing data stream $S_D^{(j)}$ match or mismatch with $\text{App}^{(i)}$ according to its following condition holds.

Then, it uses the following equation:

$$\mathbb{R}_{S_D^{(j)}} := \text{App}^{(i)}\left\{\mathbf{1} = \mathbb{M}_{(j)}^{(i)}\right\} \tag{8}$$

to represent the output of the testing result, which means that the testing stream $S_D^{(j)}$ belongs to application $\text{App}^{(i)}$, labelled by i when the value of decision-making result $\mathbb{M}_{(j)}^{(i)}$ equals $\mathbf{1}$.

In short, after the standard file is loaded, the system extracts the length information of the data frame and converts it to a string, with a certain number of characters as the unit record and short waveform, and counts the number of occurrences of the short waveform (back-off packet length every time). So the standard waveform and its number of occurrences would be obtained through the training and AC automata established on this basis. In the testing stage, the system conducts string conversion with other data and carries out the AC match, as well as recording pattern matching occurrence. The matching number and the standard number are calculated, in order to obtain the square of the difference, which will be considered as the basis of the judgment. Both the differences between similar applications and different applications will be recorded. Likewise, the best judgment threshold value will be found to distinguish the different types of application through repeated experiments. On one side of the threshold value, they are similar applications, and on the other side they do not fall into this category (see Figure 4).

4.3. Advanced Approach: AC-Histogram Waveform. As with the description of the AC waveform, similarly, Symbol $\text{App}^{(i)}$ is also used to represent applications and $S_T^{(i)}$ is the ith training stream, whereby label T means that the stream is for training and $\text{App}_{S_T}^{(i)}$ is used to represent the application of the training stream with $i = 1, 2, 3, \ldots$ which is the serial number of the training application. Thus, $(S_T^{(i)}, \text{App}_{S_T}^{(i)})$ is regarded as a training sample labelled by symbol $\text{App}_{S_T}^{(i)}$, which will be the input of the system in the training stage.

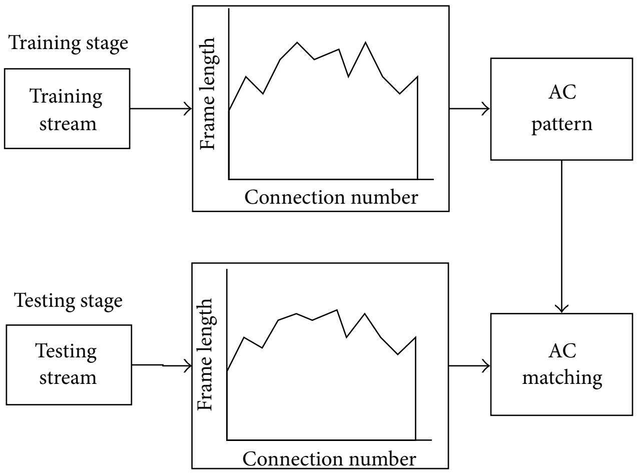


Figure 4: AC waveform flow.

In the training stage, after the same data preprocessing, which includes frame length information extraction, as well as waveform establishment (see (1) and (2)), however, we will conduct a different AC automaton establishment process, the $\Psi_e(\,)$ function, and the new automaton will be denoted by $\psi_T^{(i)}$ in the following equation:

$$\begin{gathered}\psi_T^{(i)} := \Psi_e\left(\delta\left[P_m\right]_T^{(i)}\right), \\ \text{for each } P_m : \mathscr{C}\left[P_m\right]_T^{(i)} := \sum_m \mathbf{1}\left\{O_c\left(\delta\left[P_m\right]_T^{(i)}\right)\right\},\end{gathered} \tag{9}$$

during which O_c function is added to counting the number of occurrences of each patterns, and an array symbol $\mathscr{C}[P_m]_T^{(i)}$ is used to store the statistical results for each pattern, which will provide the basis of the data for the establishment of the histogram in the following step.

The next step will be the histogram establishment process, which can be illustrated by

$$\mathscr{H}_T^{(i)} := H_e\left(\mathscr{C}\left[P_m\right]_T^{(i)}, \delta\left[P_m\right]_T^{(i)}\right). \tag{10}$$

In the test stage, $S_D^{(j)}$ is used to represent data stream for detection and feed it into the test system. Through the same data preprocessing, which includes frame length information extraction, as well as waveform establishment (see (1) and (2)), the quantized data of short patterns can be obtained by carrying out data conversion on the $\omega_D^{(j)}$ established:

$$\delta\left[P_n\right]_D^{(j)} := \Delta_e\left(\omega_D^{(j)}\right). \tag{11}$$

Then, input $\delta[P_n]_D^{(j)}$ into each of the AC-histogram automata, for which the symbol is $\psi_T^{(i)}$, constructed in the training stage, and simultaneously count the number of matches for each short pattern, which can be illustrated by the following equation:

$$\text{for each } P_n : \mathscr{C}[P_n]_{S_D^{(j)}}^{(i)} := \sum_n \mathbf{1}\left\{M_c\left(\delta[P_n]_D^{(j)}, \psi_T^{(i)}\right)\right\}, \quad (12)$$

where the M_c function refers to the matching process after the testing short patterns for testing is fed into AC-histogram automaton.

Then, a histogram of the counted results $\mathscr{C}[P_n]_{S_D^{(j)}}^{(i)}$ is established by

$$\mathscr{H}_D^{(j)} := H_e\left(\mathscr{C}[P_n]_{S_D^{(j)}}^{(i)}, \delta[P_n]_D^{(j)}\right). \quad (13)$$

After obtaining $\mathscr{H}_D^{(j)}$, the Euclidean distance between the $\mathscr{H}_T^{(i)}$ constructed in training stage and $\mathscr{H}_D^{(j)}$ will be calculated and the process can be denoted by the d function. Then, we also use $\mathbb{M}_{(j)}^{(i)}$ to represent the decision-making result by comparing the computation result with the set threshold, which can be denoted by

$$\mathbb{M}_{(j)}^{(i)} := \begin{cases} \mathbf{1}, & d\left(\mathscr{H}_D^{(j)}, \mathscr{H}_T^{(i)}\right) > \text{threshold} \\ \mathbf{0}, & d\left(\mathscr{H}_D^{(j)}, \mathscr{H}_T^{(i)}\right) \le \text{threshold} \end{cases} \quad (14)$$
$$(i = 1, 2, 3, \ldots; j = 1, 2, 3, \ldots).$$

Finally, use the following equation:

$$\mathbb{R}_{S_D^{(j)}} := \text{App}^{(i)}\left\{\mathbf{1} = \mathbb{M}_{(j)}^{(i)}\right\} \quad (15)$$

to represent the output of the testing result, which means that testing stream $S_D^{(j)}$ belongs to application $\text{App}^{(i)}$, labelled by i when the value of the decision-making result $\mathbb{M}_{(j)}^{(i)}$ equals $\mathbf{1}$.

In brief, after processing the packet frame length information, establish the automaton according to the training set. In the testing stage, feed the data into the automaton for matching to achieve classification and the determination of decision. This proposed algorithm accomplishes fuzzy matching by dividing string into shorter substrings and counting the matching extent of the substrings. The specific implementation of the algorithm is shown in Figure 5.

The algorithm execution process is also divided into two parts, training and testing. The training part mainly includes the following steps: the conversion of waveform information, partitioning of the pattern string, substring processing, establishing the AC-histogram automaton, and recording the original AC-histogram statistics.

The detection stage (testing stage) includes several steps: conversion of waveform information, AC automaton matching, and record matching of histogram statistics. In order to achieve the purpose of the final classification, the system will match the histogram statistics of the waveform with the original mode AC-histogram statistics of the waveform and make a comparison to measure the extent of the subsequent matching. The process will be specifically performed as in Figure 5.

4.4. Example for Data Preprocessing and Short Pattern Conversion. The data distribution of the frame length is wide range, which approaches continuous numeric values, and this feature can describe much of the information about the data packet; thus, we can take it as a key part in the detection process. After the mathematical processing described in the previous section, the system will effectively quantify the data again and reduce the range of the data to 1–35, which it subjects to the reservation about most of the information of features. Due to the continuous characteristics of the values, as well as for keeping the chronological correlation of the network traffic, the system uses multimode matching algorithm for data classification [26].

In the preprocessing stage, after the system reads the raw standard data, firstly, the waveform of the frame length is quantified, and for ease of matching the system converts the data from a decimal number into a character, which means that 1 to 35 can be mapped to a–z and A–I, a total of 35 characters. Specifically, according to the data distribution used to conduct the conversion, the conversion rules are as follows.

51 bytes–200 bytes (of which the interval is 10 bytes) are converted to "a" to "o"; that is, if the frame length of the data is between 51 bytes and 60 bytes, the frame length information will be recorded as the letter "a," 61 bytes to 70 bytes will be denoted by b, and so on; 201 bytes–2000 bytes (of which the interval is 100 bytes) are converted to "p" to "z" and "A" to "G"; that is to say, if the frame length of the data is between 201 bytes and 300 bytes, the frame length information will be recorded as "A," the 301 bytes to 400 bytes will be denoted by "B," and so on.

Those less than or equal to 50 bytes will be denoted "H"; and those greater than 2000 bytes will be recorded as "I."

So the data frame length information will be effectively quantified, and due to the presence of a divided interval, the fuzzy matching can be effectively conducted in order to reduce interference to the system caused by the noisy frame data.

Thus, we can obtain the string of the flow of the frame length. For this example, see Figure 6 for more details.

Thus, in this procedure of data processing, the algorithm firstly converts the integer of the length information of the data frame to a string, then takes a certain number of characters as the unit of the recording pattern, and counts the pattern mode and its number of occurrences. Then, it regards it as the standard and uses these patterns to establish the AC automata. At the testing stage, it conducts string conversion with other data, uses the AC algorithms to count matching number of each pattern, and then counts the histogram of the pattern's number of occurrences in chronological order. Furthermore, the algorithm calculates the sum of the square of the difference of the remaining label histogram and according to this makes the decision. Thus, the best judgment threshold value is found through repeated experiments.

4.5. Selection Strategy of the Decision Threshold. The algorithm selects the optimal threshold of the experiments for judgment through an exhaustive method, which is described as follows.

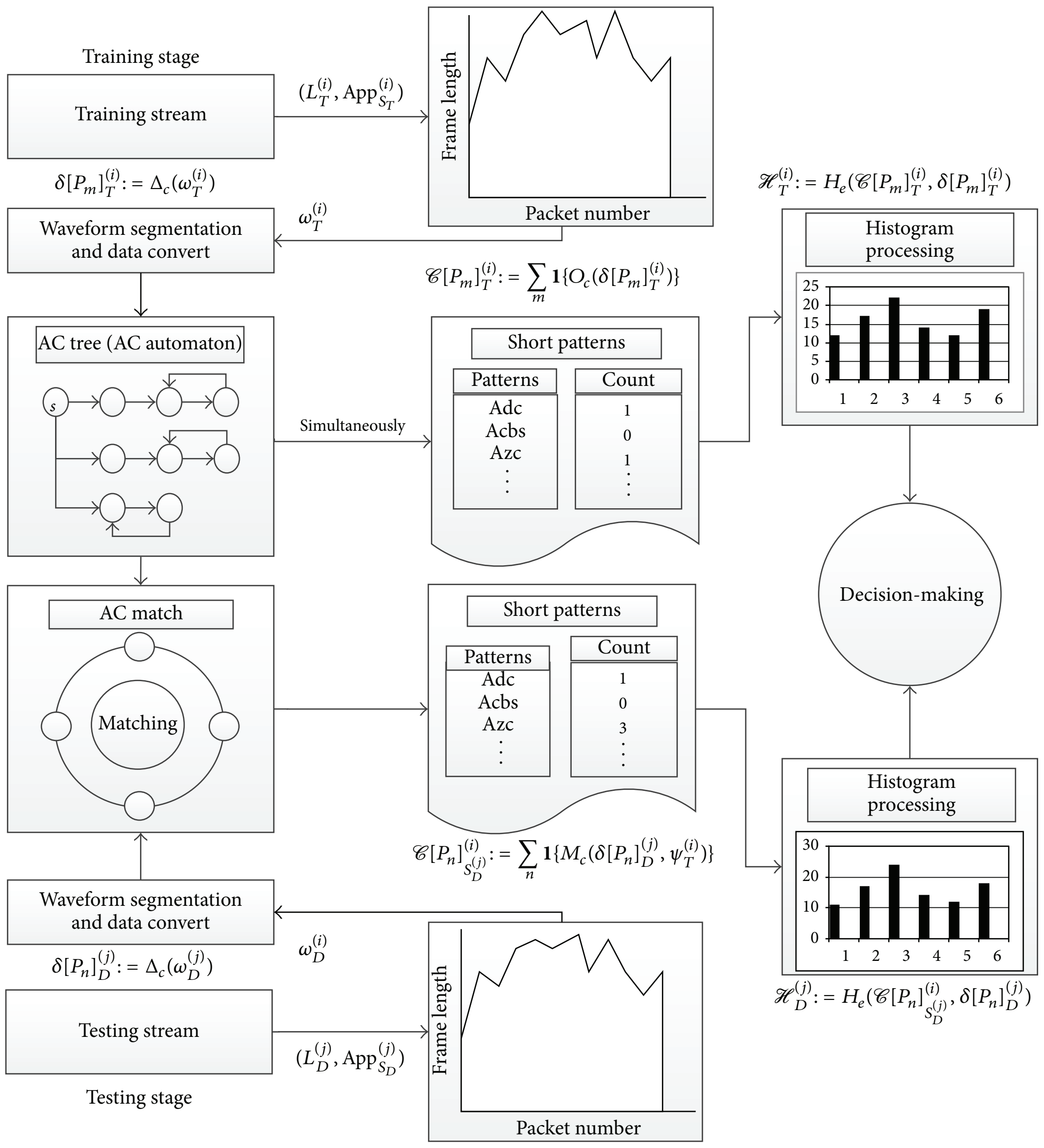

Training stage
Training stream
$(L_T^{(i)}, \text{App}_{S_T}^{(i)})$
Frame length
Packet number
$\delta[P_m]_T^{(i)} := \Delta_c(\omega_T^{(i)})$
$\omega_T^{(i)}$
Waveform segmentation and data convert
AC tree (AC automaton)
s
Simultaneously
$\mathscr{C}[P_m]_T^{(i)} := \sum_m \mathbf{1}\{O_c(\delta[P_m]_T^{(i)})\}$
Short patterns
Patterns
Count
Adc
Acbs
Azc
1
0
1
$\mathscr{H}_T^{(i)} := H_e(\mathscr{C}[P_m]_T^{(i)}, \delta[P_m]_T^{(i)})$
Histogram processing
Decision-making
AC match
Matching
Short patterns
Patterns
Count
Adc
Acbs
Azc
1
0
3
$\mathscr{C}[P_n]_{S_D^{(j)}}^{(i)} := \sum_n \mathbf{1}\{M_c(\delta[P_n]_D^{(j)}, \psi_T^{(i)})\}$
Histogram processing
Waveform segmentation and data convert
$\delta[P_n]_D^{(j)} := \Delta_c(\omega_D^{(j)})$
$\omega_D^{(i)}$
Testing stream
$(L_D^{(j)}, \text{App}_{S_D}^{(j)})$
Testing stage
Frame length
Packet number
$\mathscr{H}_D^{(j)} := H_e(\mathscr{C}[P_n]_{S_D^{(j)}}^{(i)}, \delta[P_n]_D^{(j)})$

Figure 5: AC waveform histogram flow.

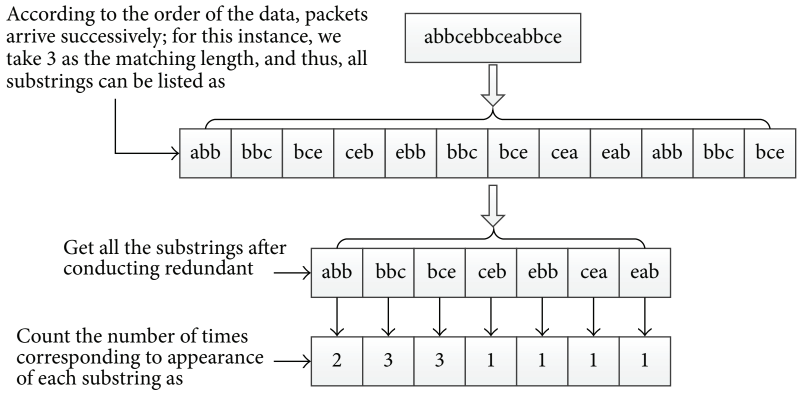

According to the order of the data, packets arrive successively; for this instance, we take 3 as the matching length, and thus, all substrings can be listed as
abbcebbceabbce
abb
bbc
bce
ceb
ebb
bbc
bce
cea
eab
abb
bbc
bce
Get all the substrings after conducting redundant
abb
bbc
bce
ceb
ebb
cea
eab
Count the number of times corresponding to appearance of each substring as
2
3
3
1
1
1
1

Figure 6: Example of quantization process of frame length information.

Suppose there are two types of data named as A and B, and each one of them is one-dimensional. The range of data of class A is $A_{\max} \sim A_{\min}$, while the range of the class B data is $B_{\max} \sim B_{\min}$.

If there is no crossover between the two types of data, we may wish to assume that $A_{\max}$ is less than $B_{\min}$, then taking a certain value between them as the judgmental threshold, and the accuracy of distinguishing it is 100% otherwise there should be two cases $B_{\max} \geq A_{\max} \geq B_{\min}$ or $B_{\max} \geq A_{\min} \geq B_{\min}$. We will enumerate all the values between $A_{\max}$ and $B_{\min}$ or $A_{\min}$ and $B_{\max}$, which will be regarded as the judgmental threshold sequentially in following steps, and then record the number of samples with an error, noting the smallest number which is favoured as the threshold.

4.6. Correctness of Proposed Algorithm. In this section, we prove the correctness of our proposed algorithms, which include AC waveform algorithm and AC-histogram algorithm.

4.6.1. Proof of Correctness of AC Waveform Algorithm. The changes of the data frame length information $L_T^{(i)}$ within a period of detection will be regarded as the determination basis for the network behaviour of an application. After the quantization process, the Γ_e function, the waveform information $\omega_T^{(i)}$ of the data frame length will be obtained, that is, the behavioural statistics information. Then, data conversion (Δ_c function) is conducted for the waveform to obtain the behavioural string data. Through fixed-length segmentation, the substring patterns of behaviour $\delta[P_m]_T^{(i)}$ will be obtained. On this basis, establish the AC automaton $\phi_T^{(i)}$ by the utilization of the redundant pattern strings. In the testing stage, by carrying on the same data preprocessing for detecting data stream, the pattern strings of the testing data $\delta[P_n]_D^{(j)}$ will be obtained. Then, input the patterns into the established AC automaton $\phi_T^{(i)}$ to conduct the pattern match, and count the number of matching occurrences $\mathscr{C}_{S_D^{(j)}}^{(i)} := \sum_n \mathbf{1}\{M_c(\delta[P_n]_D^{(j)}, \phi_T^{(i)})\}$. When the number of matching is larger than a certain threshold, we have reason to believe that the behaviour of the test data waveform has reached a certain level of matching with the waveform of the training samples, on the basis of which the AC automaton is established. And the degree of matching allows us to accept the assumption that the detected data belong to a certain application.

To a certain extent, the AC waveform detection method observes more of the overall trends in behavioural waveforms of the data frame length information within a certain period of time, rather than focusing on the detection of the detailed information of matching within certain short time intervals.

4.6.2. Proof of Correctness of AC Waveform Histogram Algorithm. The data preprocessing of AC waveform and histogram method is the same as the AC waveform method described above. Similarly, it treats the changes in the data frame length information $L_T^{(i)}$ within a period of detection as the basis of determination for the network behaviour of an application. After the process of quantization Γ_e, the waveform information on the data frame length, that is, the behavioural statistics information $\omega_T^{(i)}$, can be obtained. Then, data conversion (Δ_c function) for the waveform is conducted to obtain the behavioural string data. Through fixed-length segmentation, the pattern substrings of behaviour $\delta[P_m]_T^{(i)}$ will be obtained. On this basis, the establishment of the AC-histogram automaton $\psi_T^{(i)}$ will be conducted by the utilization of the redundancy pattern strings. However, the difference is that during the process of establishment of AC-histogram automaton $\Psi_e()$, more detailed information is simultaneously recorded, that is, the number of occurrences of each string pattern $\mathscr{C}[P_m]_T^{(i)} := \sum_m \mathbf{1}\{O_c(\delta[P_m]_T^{(i)})\}$, and establish histogram (H_e function) for them, which will be the basis of decision-making for matching. In the testing stage, we will carry out the same preprocessing as for the detecting data stream, and the pattern strings $\delta[P_n]_D^{(j)}$ of the testing data will be obtained. Then, the patterns are input into the established AC-histogram automaton $\psi_T^{(i)}$ to conduct pattern matching, similar to the training stage, the number of matching occurrences for each pattern is counted, $\mathscr{C}[P_n]_{S_D^{(j)}}^{(i)} := \sum_n \mathbf{1}\{M_c(\delta[P_n]_D^{(j)}, \psi_T^{(i)})\}$, and a histogram is constructed. Then, it makes comparison between the histograms built in the two stages, and the Euclidean distance is taken as the measure of the degree of matching. When the computed value of the distance is less than a certain threshold, it thus has reason to believe that the behaviour of the test data waveform achieves a certain level of match with the waveform of the training samples based on which AC-histogram automaton is established. And the degree of matching allows us to accept the assumption that the detected data belongs to the certain application.

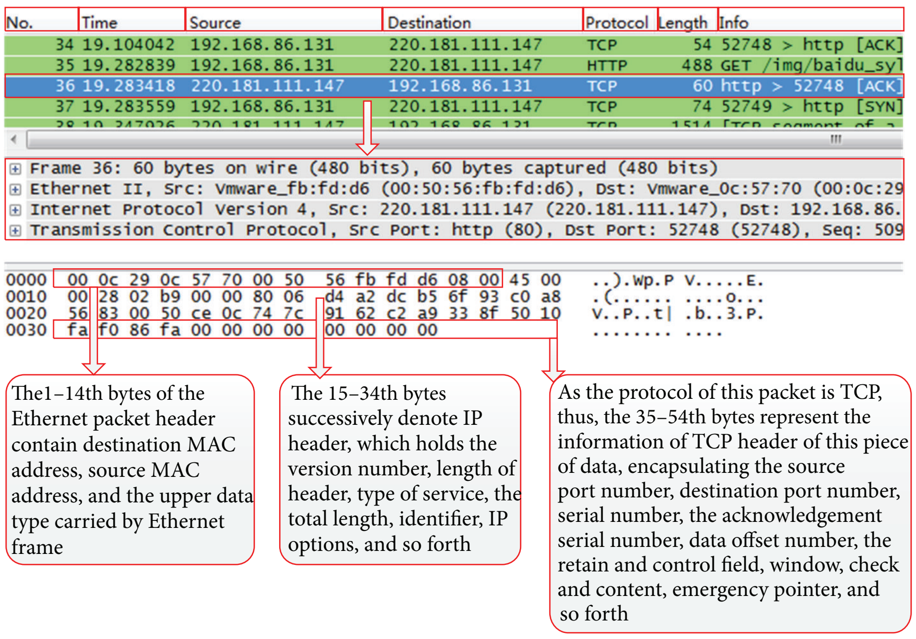
In contrast, to a certain extent the AC waveform histogram method requires more demanding conditions for matching determination. Not only the overall trend of behavior waveform within a certain period of time reaching a certain matching degree is required, but also the condition that small zones of the behaviour waveform need to achieve a certain level of matching. Depending on these conditions, the conclusion of decision-making for determination of matching can be drawn. That is to say, we conduct the measure of matching in smaller intervals, in order to achieve a higher accuracy.

5. Advanced Analysis

5.1. Extraction and Processing of the Packet Feature Information

5.1.1. Original Packet. The original data can be obtained by capturing packets at the gateway. For ease of interpretation, we use *Wireshark*, which is network software used for packet analysis, shown in Figure 7.

The hexadecimal data behind the comprehensive information is the contents of the data frame, such as data packet number, frame length, and arrival time.



```
No.  Time       Source           Destination      Protocol Length Info
  34 19.104042  192.168.86.131   220.181.111.147  TCP          54 52748 > http [ACK]
  35 19.282839  192.168.86.131   220.181.111.147  HTTP        488 GET /img/baidu_syl
  36 19.283418  220.181.111.147  192.168.86.131   TCP          60 http > 52748 [ACK]
  37 19.283559  192.168.86.131   220.181.111.147  TCP          74 52749 > http [SYN]

⊞ Frame 36: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
⊞ Ethernet II, Src: Vmware_fb:fd:d6 (00:50:56:fb:fd:d6), Dst: Vmware_0c:57:70 (00:0c:29
⊞ Internet Protocol Version 4, Src: 220.181.111.147 (220.181.111.147), Dst: 192.168.86.
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: 52748 (52748), Seq: 509

0000  00 0c 29 0c 57 70 00 50  56 fb fd d6 08 00 45 00   ..).wp.P V.....E.
0010  00 28 02 b9 00 00 80 06  d4 a2 dc b5 6f 93 c0 a8   .(...... ....o...
0020  56 83 00 50 ce 0c 74 7c  91 62 c2 a9 33 8f 50 10   V..P..t| .b..3.P.
0030  fa f0 86 fa 00 00 00 00  00 00 00 00               ........ ....
```

The1–14th bytes of the Ethernet packet header contain destination MAC address, source MAC address, and the upper data type carried by Ethernet frame

The 15–34th bytes successively denote IP header, which holds the version number, length of header, type of service, the total length, identifier, IP options, and so forth

As the protocol of this packet is TCP, thus, the 35–54th bytes represent the information of TCP header of this piece of data, encapsulating the source port number, destination port number, serial number, the acknowledgement serial number, data offset number, the retain and control field, window, check and content, emergency pointer, and so forth

Figure 7: Parse packets result from Wireshark.

Table 1: Specific features selected for this paper.

No.	Feature Type	Remark
1	Frame length	Data packet frame length, in bytes
2	Ethernet protocol type	Ethernet type (IP = 0x0800, ARP = 0x0806, RARP = 0x8035, and others recorded as −1)
3	IP protocol type	Such as TCP = 6, UDP = 17, ICMP = 1, and others recorded as −1
4	ARP/RARP opcode	ARP/RARP type of operation (ARP request is 1, the ARP response is 2; RARP request is 3, and RARP response is 4)
5	ICMP type	ICMP message type
6	ICMP code	ICMP message code
7	TCP/UDP source port number	TCP/UDP packet source port
8	TCP/UDP destination port numbers	TCP/UDP packet destination port
9	TCP control field	TCP packet flag field

Feature information needed by this system will be extracted and calculated from data packet similar to the above, and the following sections will provide specific description.

5.1.2. Feature Extraction and Quantification. Feature extraction is an important part of the input for the classifier, which has a direct impact on the quality of the final determination. This paper referenced some of the methods, Cannady selected nine data features; not only because they are normally present in the network packets but also because they provide a complete description of the packet transmission information [27], which includes protocol type, source port, destination port, source IP address, destination IP address, ICMP type, ICMP code, the length of the original data, and the data portion [28]. Today, however, due to the widely used characteristics of distributed technology, features such as source and destination IP address become nonrepresentative. In addition, KDD-CUP-99 processed the data of the 1998 DARFA intrusion detection evaluation program and selected 41 dimensional feature data with which we conduct detection analysis [29]. However, due to the cumbersome processing of the experimental data and offline detection, it is usually too much for academic research. According to the topic of statistical-based information and the real-time requirements, this paper extracted 9 data features from the data packets as the object for study. These features are a good representative of the information contained in the data transmissions, and the specific features are listed as in Table 1.

In order to facilitate the subsequent operation of the algorithm, the system will carry out the decimal conversion processing for each feature, wherein the hexadecimal data of the Ethernet protocol type is converted into decimal numbers; that is, IP = 2048, ARP = 2054, and RARP = 32821. In addition, the binary data of the TCP control field is converted into decimal numbers. The remaining feature information is denoted directly by its original decimal value. Through the above quantization method, the system can obtain 9-dimensional decimal data from each data frame based on the selected features, and the statistical waveform of the various characteristics of the segment data packets can be obtained after a certain flow of data packets.

5.2. Packets Matching Algorithm. After obtaining waveform information for the original features of the network segments, the core part of the system will be conducted, the pattern matching calculation section. To this end, the different computation and statistical programs were designed for various features of the original waveform in order to obtain the statistical feature values which carry significant characteristics of the segment of the data flow. Then, these data are used to set the classification rules of the classifier. Through a great deal of training and experimentations, the means of the measure of the matching degree are set in addition to the threshold determined, as well as the implementation of network packet matching and outputting of the results.

With regard to statistical feature extraction, the system adopted the histogram method to determine part of the information. However, for the data frame length information, this study designed a unique method, that is, the AC match-histogram method, and utilized the above program to complete the calculation process of the feature statistics. In addition, multiple strategies for measuring the degree of matching have been designed, such as the direct calculation of the Euclidean distance between the statistical waveforms. See below for a detailed description of the algorithm.

5.2.1. Design and Calculation of the Feature Information of the Statistical Waveform. After a certain time span of the data packets is obtained, the system will extract and quantify the feature information of each data packet, so that the statistical waveform information of each feature in the network traffic segments will be obtained. According to the distribution of the facts and characteristics of the figures for each feature, the system classified the features into three types, in order to obtain different statistics.

(1) Features of Data Protocol Type. The system will conduct a statistical investigation and analysis of the Ethernet protocol type, the IP protocol type, and ARP/RARP opcode of the data packet. Since the data of these features is discrete, which also keep the mutually exclusive relationship, the data thus can be divided into different 19 categories. To this end, the system regards IP protocol type, ARP/RARP operation as determination conditions, to classify and analyze the data packets.

When each packet is obtained, the system will carry out the first division according to its Ethernet type, and thus the data packet will be classified into three branches, which include IP packets (Ethernet type is 2048), ARP packets (Ethernet type: 2054), and RARP packets (Ethernet type 32821), and does not count the rest. For the first branch of the IP packet node, according to its specific IP protocol, the data is once again divided into TCP packets (IP protocol type 6), UDP data packet (IP protocol type 17), and ICMP data packets (IP protocol 1), and the rest will not be counted. For the second branch of the ARP packet node, according to its specific protocol, the data packet can be classified into specific types with regard to the operation requested: operation package (opcode = 1) and response operate packet (opcode = 2). Similarly, the third branch of the RARP data packet can also be divided into the requested operation packet (operation code is 3) and response operation packet (operation code is 4).

Thus the system will use the several data features mentioned above and the designed classification rules to ultimately summarize and allocate each data packet into one of seven categories. However, when the original packets of data traffic set by the system are obtained, the system will determine the number of data frames classified into each category for each segment, and after organization a piece of statistical information with a length of 7 can be obtained. See Figure 8(a) for details of the histogram diagram.

(2) Port Number and TCP, ICMP Specific Features of Information, including the Message Source and Destination TCP/UDP Port Number, TCP Control Field, ICMP Code, and ICMP Type. This part can extract seven sets of data from the packets, and all of them are discrete. The respective description of each of feature is listed as follows.

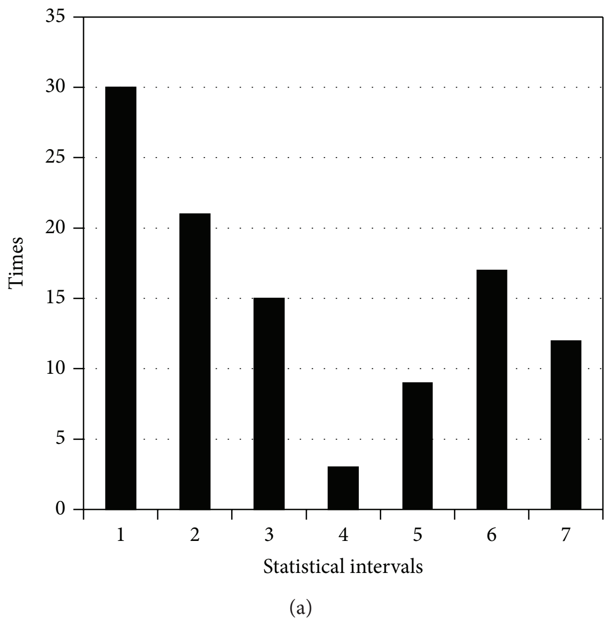
(a) TCP/UDP source and destination port number ranges from 0 to 65535, according to the type and distribution of the port. The port number can be divided into seven intervals. After the equalization process, the interval ranges are 0–21, 22–50, 51–100, 101–500, 501–1023, 1024–49151, and 49152–65535. Thus the system can obtain histogram statistics of the data port information of this data flow, as shown in the diagram in Figure 8(b).

(b) ICMP type features, divided according to the value of the data distribution, by equalizing the height, and the intervals are divided as follows: 0–7, 8, 9–18, and 19–255, that is, four intervals.

(c) ICMP code, which is very simple and does not require much processing, could be classified into five categories, which can be denoted by five intervals including 0, 1, 2, 3, and >3.

(d) TCP control field. According to the different senses of the 6-bit flag segment characters conduct statistical computation according to whether each bit is effective, which is divided into six intervals, URG = 1, ACK = 1, PSH = 1, RST = 1, SYN = 1, and FIN = 1, so that the system can obtain three statistical histogram waveforms similar to the port number of the data packets and ICMP and TCP information features.

(3) Frame Length Information. Frame length information in bytes, its value on the basis of the experiments in this



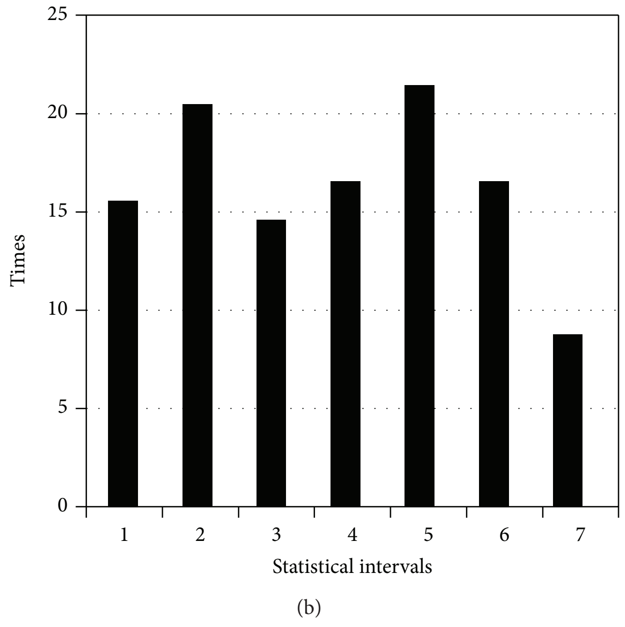


FIGURE 8: (a) Protocol type feature: port number feature. (b) Statistical waveform.

paper, is almost continuous, ranging from 50 to 2000 bytes. Similarly, according to a large number of experimental data, the optimal division of intervals is as in Table 2.

Thus the length information of each frame can be reduced to a range of 35 discrete data.

5.3. System Output. With the various statistical features of information matched, the system will obtain the corresponding result of the matching. Through the generalization of these results, the final decision about whether the data packet matches the corresponding behaviour will be made. The consolidated results of the matches of various statistical features can be denoted by the following equation:

$$\mu = \mu_1 * w_1 + \mu_2 * w_2 + \cdots + \mu_n * w_n = \sum_{i=1}^{n} \mu_i * w_i, \quad (16)$$

where μ_i is the match situation of each of the statistical features; if there is a matching case, its value will be 1; otherwise, it will be 0. w_i is the weight of each item; in this study, the features are similar; that is, no major core features exist, so the values will be set as

$$w_i = \frac{1}{n} = \frac{1}{9} \quad (17)$$

which means that the weights of the features are equal to each other. This trial extracted 9 features of information of the statistical waveform, therefore, basically set each of the weights to 1/9.

After the training process we can determine the threshold of μ, which is the consolidated matching value, and make the final decision with regard to the match results, so that we can accomplish the objective of the detection of network data packets.

TABLE 2: Interval setting strategy for frame length.

	Intervals
1	Less than or equal to 50 bytes
2	51–200 bytes (interval of 10 bytes), that is, 51–60 bytes, 61–70 bytes, and so forth
3	201–2000 bytes (interval of 100 bytes), that is, 201–300 bytes, 301–400 bytes, and so forth
4	greater than 2000 bytes

Note: a total of 35 intervals.

6. Evaluation

According to the description of the system framework and the detection processes in the previous sections, in this study, we built a corresponding network environment and implemented the network packet detection system on the gateway. The system utilizes related algorithms to conduct data preprocessing, feature statistics calculations, matching determination, and the output of the final test results. We also illustrate the performance of the algorithms by a comparison experiment with an approximation algorithm. We built the network environment in our lab and implemented a real-time packet inspection system based on the Linux operating system on the gateway and used PHP to build the front end operation webpage of the system in order to make the interface user-friendly.

6.1. Environment Setup. Experiments were carried out in an environment containing a PC, which will be the gateway through which other PCs can connect with the external network and be used. See Figure 9 for details of the environment setup.

Table 3: Quantization process for the 9 selected packet features.

Number	Feature type	Values
1	Frame length	The consecutive integer value
2	Ethernet protocol type	2048, 2054, 32832, −1
3	IP protocol type	6, 17, −1
4	TCP/UDP source port number	An integer of 0–65535, −1
5	TCP/UDP destination port numbers	An integer of 0–65535, −1
6	ICMP type	An integer of 0–255, −1
7	ICMP code	An integer of 0–255, −1
8	TCP control field	The consecutive integer value, −1
9	ARP/RARP opcode	1, 2, 3, 4, −1

The gateway PC uses double network cards to implement subnet data transmission. The intrusion detection system designed in this paper is also set up on this PC.

6.2. Packet Capture. The system utilizes *libpcap* to obtain the data packets. *Libpcap* is a network packet capture and filtering package which was originally developed by a research team at the Lawrence Berkeley Lab.

This system will mainly use the *pcap_findalldevs* function to search the network devices and the *pcap_loop* function to perform packet capture. The data packets flowing through the network card will be printed out, as shown in Figure 10.

The packet number, length, and capture time information will be first and then the packet specific hexadecimal content. The feature information required by the system on the packets will be read out and calculated from the data packet content. We can extract the various characteristics required by the algorithm which enables information of the packet to be described.

6.3. Data Preprocessing. According to the features extraction algorithm designed in the previous section, we will quantify 9-dimensional values of the data packets features, which will be converted into decimal integers after processing (see Table 3 for detailed information).

We can thus obtain data on the characteristics of the packet in a segment of network traffic, and the system will record it as shown in Table 4.

According to the feature value information of each data frame in the traffic, we can obtain the value of the various features of the characteristics of the waveform in the flow, thereby performing the computation of the amount of feature statistics.

6.4. Calculation of Packet Features. The system uses the features statistics algorithm designed in the previous section in order to calculate the statistical information of each of the values of features needed to be classified. For different features, different statistics calculations are applied. The conversion outcomes of the frame length information of the data flow generated by the log MSN application are following, and the system will consider 50 data frames as a process unit for evaluation. String patterns generated by the frame length information are listed as follows (length 3); see Table 5.

The numbers of occurrences of each pattern are shown as waveforms in Figure 11(a).

As shown in the above waveform, according to the chronological order of the first occurrence of the string pattern, a uniform interval histogram is constructed (interval length is 5); that is, Figure 11(b) shows the AC matching-histogram information on the length information of the data flow.

Figure 12(a) shows the statistical information of this network traffic according to the partition method described in the previous section.

This indicates that the network traffic mostly consists of TCP packets, as well as a small amount of UDP packets. With respect to the ICMP type and ICMP code information, since the flow does not contain ICMP packets, its statistical information is not recorded. Figure 12(b) shows the TCP control field information.

Figures 13(a) and 13(b) show the TCP source and destination port number information according to the previous section. Figures 13(c) and 13(d) show the UDP source and destination port number information.

6.5. Matching Classification. After obtaining the statistics on the various characteristics of the data segments, the system will obtain the parameters matching the algorithm required for each network behaviour through repeated training. The investigation established the environment and system and conducted real-time data monitoring and training on three types of instant messaging software, namely, QQ, Fetion, and Renren desktop. In order to enhance the efficiency of the experiment, when the frame length characteristic was extracted, we chose the TCP packet for training and the other protocol type packets were not included in the statistics. The parameters are listed as in Table 6.

Because the system does not capture TCP packets in the training process of the QQ application, the string pattern is empty, so there is no need to conduct matching. And the pattern strings from the Fetion and Renren desktop applications can be seen in Table 7.

Obviously the data packet flows of the Fetion and Renren desktop applications are approximate. However, in comparison with the QQ application, the case is quite different.

The procedures for matching and classification in the following sections are all based on these training data.

6.6. System Operation Interface. The webpage of the real-time network behaviour detection system designed in this article is shown in Figure 14.

In the training stage, the two edit boxes correspond to the number of training packets and name of the training application which will be processed in the following training steps. The TRAINING_PROCESS_1 button will conduct the

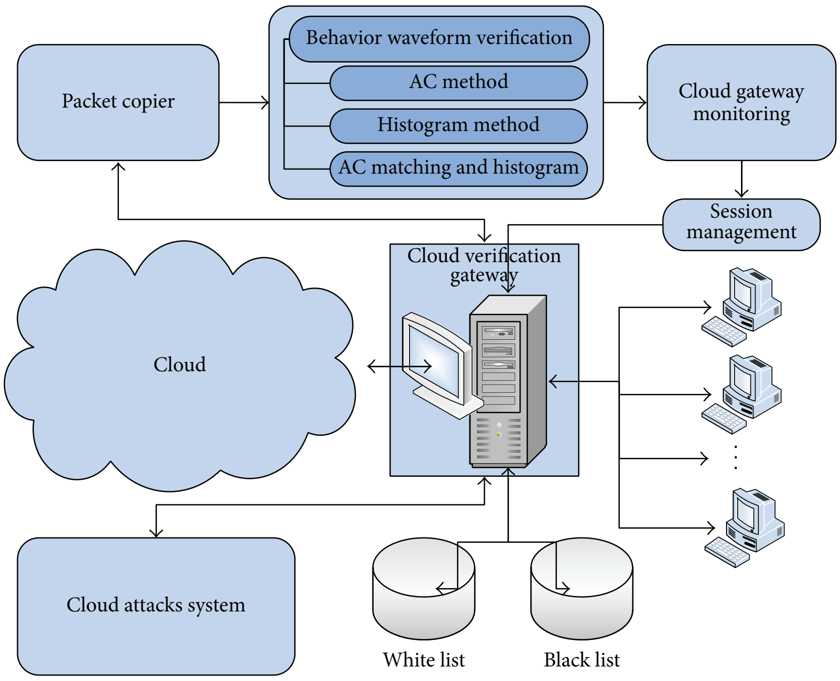
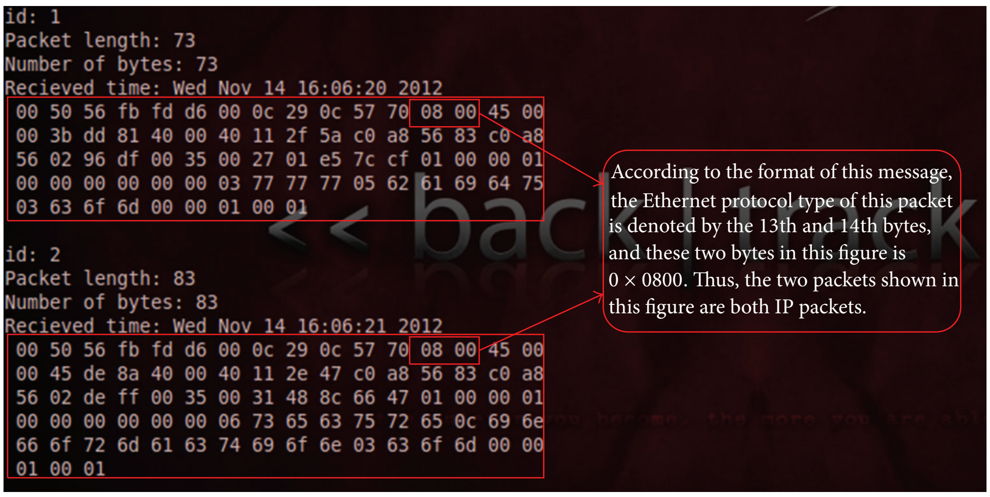


Figure 9: Diagram of evaluation environment.



```
id: 1
Packet length: 73
Number of bytes: 73
Recieved time: Wed Nov 14 16:06:20 2012
 00 50 56 fb fd d6 00 0c 29 0c 57 70 08 00 45 00
 00 3b dd 81 40 00 40 11 2f 5a c0 a8 56 83 c0 a8
 56 02 96 df 00 35 00 27 01 e5 7c cf 01 00 00 01
 00 00 00 00 00 00 03 77 77 77 05 62 61 69 64 75
 03 63 6f 6d 00 00 01 00 01

id: 2
Packet length: 83
Number of bytes: 83
Recieved time: Wed Nov 14 16:06:21 2012
 00 50 56 fb fd d6 00 0c 29 0c 57 70 08 00 45 00
 00 45 de 8a 40 00 40 11 2e 47 c0 a8 56 83 c0 a8
 56 02 de ff 00 35 00 31 48 8c 66 47 01 00 00 01
 00 00 00 00 00 00 06 73 65 63 75 72 65 0c 69 6e
 66 6f 72 6d 61 63 74 69 6f 6e 03 63 6f 6d 00 00
 01 00 01
```

According to the format of this message, the Ethernet protocol type of this packet is denoted by the 13th and 14th bytes, and these two bytes in this figure is 0 × 0800. Thus, the two packets shown in this figure are both IP packets.

Figure 10: Data packet information.

extraction process on the features of the training samples, as well as the operation of the establishment of the AC automaton; the TRAINING_PROCESS_2 button will carry out the comparison between the extraction statistics and training data and update the matching thresholds of the various training parameters. Users can conduct training by themselves, as well as increasing the number of training data and patterns according to their needs which indicate the improved scalability of the system.

In the testing stage, the edit box corresponds to the setting of the number of the behavior which will be detected. The DETECTAING button will conduct the real-time detection, and the STOP-DETECTING button will end the detection operation.

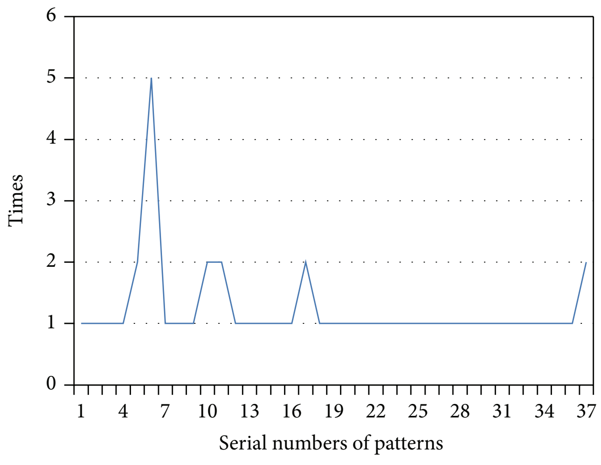
Figure 14 shows how to use the system. As can be seen from the figure, when the system is running and after the training stream window outputs each training session, the update of each parameter will be shown and the detecting stream window will display the real-time testing results.

During detection, the system will generate a detection log, to record and prompt the detection of network behavior, as well as the labelled classes and specific data frame number of the network behavior as shown in Figure 14.

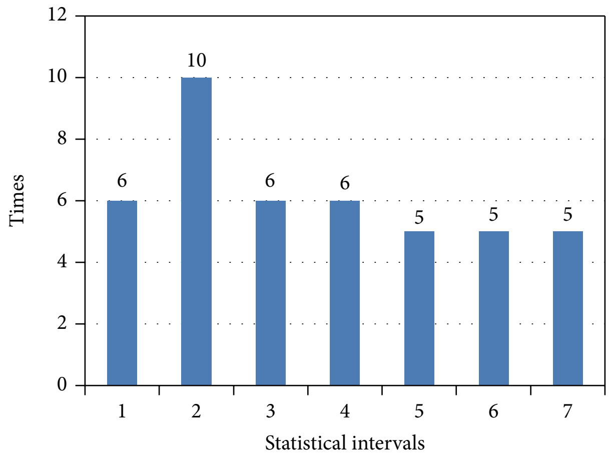
Table 4: Statistics for a piece of network traffic after quantization.

Number	Features number								
	1	2	3	4	5	6	7	8	9
1	60	2048	6	80	55652	−1	−1	18	−1
2	54	2048	6	55652	80	−1	−1	16	−1
3	441	2048	6	55652	55652	−1	−1	24	−1
4	60	2048	6	80	55652	−1	−1	16	−1
5	397	2048	6	80	55652	−1	−1	24	−1
6	54	2048	6	55652	80	−1	−1	16	−1
7	1514	2048	6	80	55652	−1	−1	16	−1
8	54	2048	6	55652	80	−1	−1	16	−1
9	1514	2048	6	80	55652	−1	−1	16	−1
10	54	2048	6	55652	80	−1	−1	16	−1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Comment: The feature number in this table corresponds to features in Table 4; for example, frame length = 1.

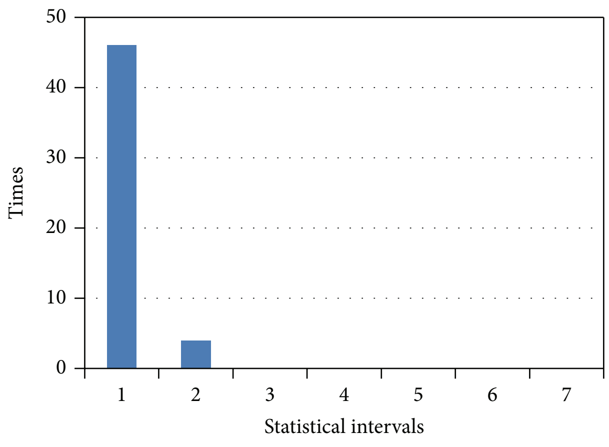


(a)

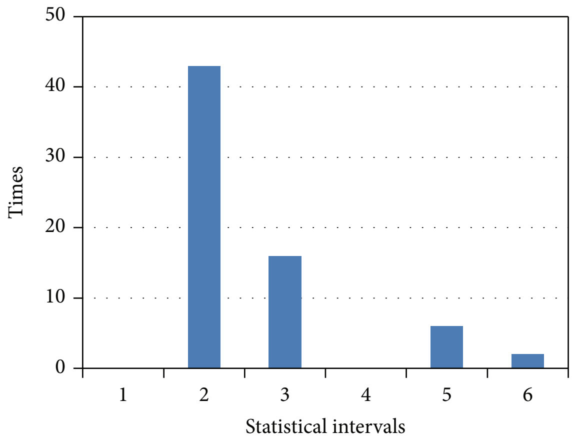


(b)

Figure 11: (a) String pattern occurrences waveform; (b) AC matching-histogram information of frame length.



(a)



(b)

Figure 12: (a) Packet protocol type statistical histogram; (b) TCP control the field statistical histogram information.

Table 5: String pattern.

Short patterns (length 3)								
dqb	qbc	bcq	cqb	qbb	bba	bak	akb	kbb
bad	adb	dbj	bjb	jbe	beb	ebC	bCC	CCC
CCh	Cha	haq	aqb	bbb	baf	afs	fsb	sbC
CCb	Cbb	bbx	bxb	xbp	bpd	pda	daa	aab
abb	bab	dbn						

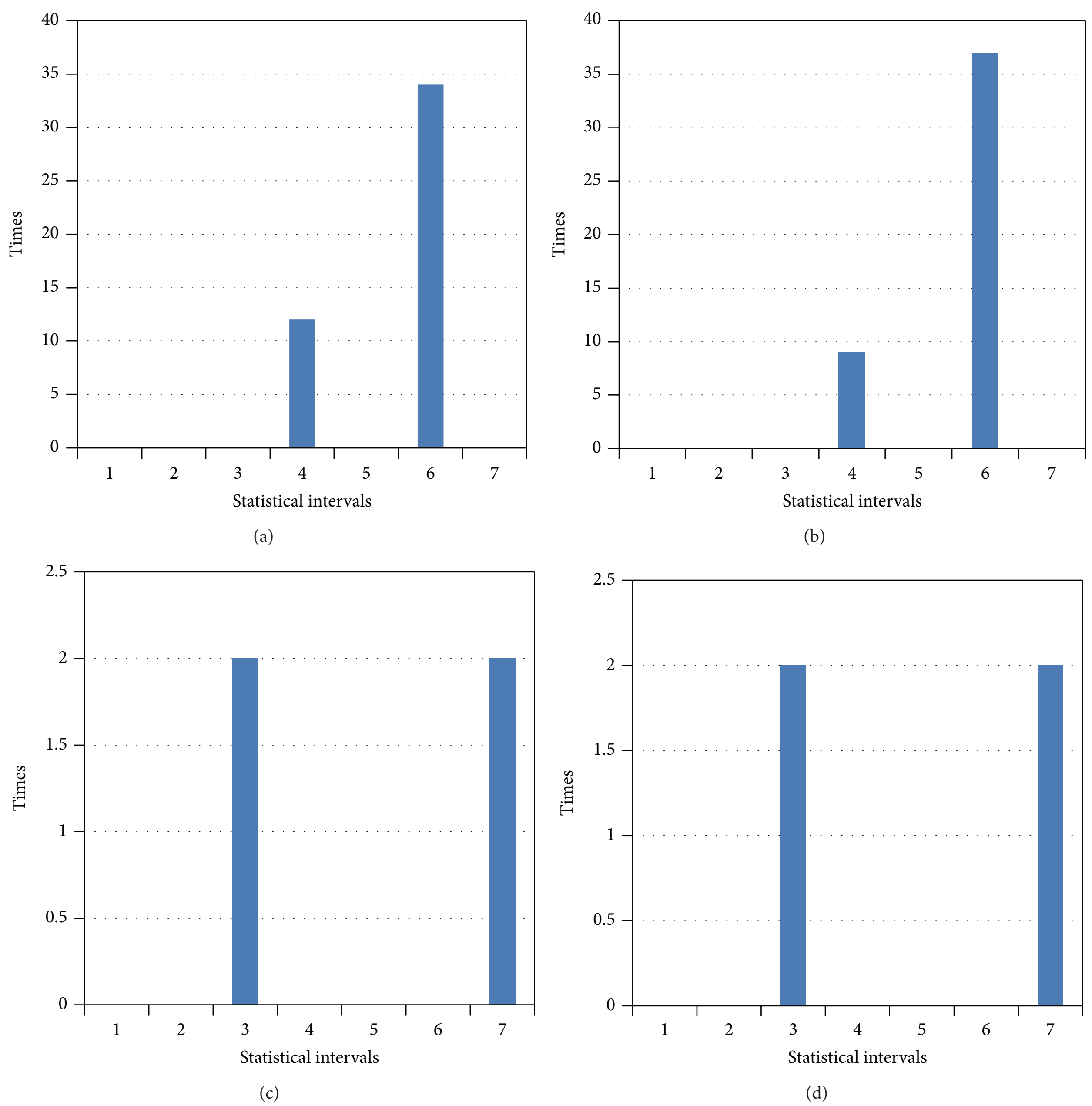


Figure 13: Statistical histogram information of TCP and UDP port number: (a) TCP source port number, (b) TCP destination port number, (c) UDP source port number, and (d) UDP destination port number.

Table 6: Results for three applications after training.

Features	Items					
	QQ		Fetion		Renren desktop	
	Match centre coordinates	Match threshold	Match centre coordinates	Match threshold	Match centre coordinates	Match threshold
Protocol type	(0, 48, 0, 0, 0, 0, 0)	2	(36, 13, 0, 1, 1, 0, 0)	18.44	(32, 15, 0, 1, 0, 0, 0)	22.69
TCP source port number	(0, 0, 0, 0, 0, 0, 0)	0	(0, 0, 17, 0, 0, 6, 19)	11	(0, 0, 8, 0, 0, 0, 33)	19.69
TCP destination port number	(0, 0, 0, 0, 0, 0, 0)	0	(0, 0, 14, 0, 0, 5, 23)	13.93	(0, 0, 9, 0, 0, 0, 32)	19.23
UDP source port number	(0, 0, 0, 0, 0, 48, 0)	4	(0, 0, 0, 0, 0, 3, 0)	3	(0, 0, 0, 0, 0, 3, 0)	3
UDP destination port number	(0, 0, 0, 0, 0, 48, 0)	4	(0, 0, 0, 0, 0, 3, 0)	3	(0, 0, 0, 0, 0, 3, 0)	3
ICMP type	(0, 0, 0, 0)	0	(0, 0, 0, 0)	0	(0, 0, 0, 0)	0
ICMP code	(0, 0, 0, 0, 0)	0	(0, 0, 0, 0, 0)	0	(0, 0, 0, 0, 0)	0
TCP control field	(0, 0, 0, 0, 0)	0	(0, 40, 8, 0, 4, 1)	21.33	(0, 40, 20, 0, 3, 5)	33.31
TCP frame length	—	—	(5, 5, 5, 12, 9, 5)	184	(6, 5, 5, 9, 5, 5, 5)	185

Table 7: String patterns of frame length information for three applications.

APP	String patterns for frame length information								
QQ	EMPTY								
Fetion	asB	sBb	BbB	bBa	Bay	ayb	ybb	bba	ban
	ana	nau	auB	uBm	Bmq	mqb	qbb	bbB	bBB
	BBa	BaB	aBB	BBB	Baa	aab	abB	bBy	Bya
	yaw	awr	wrp	Rpb	pbB				
Renren	hbb	bba	ban	anh	nho	hoo	ooa	oap	apf
	pff	ffm	fmw	mwp	wpg	pgp	gpp	ppa	pap
	app	ppn	pnp	npp	ppq	pqm	qmm	mmm	mmp
	mpp	paa	aab	abb	bbp	bps	psr	srb	rbb
	baa	aaq							

Table 8: Experimental results from network application detection.

	Fetion	MSN	Nmap	ooVoo	QQ	Renren	SkyDrive	Skype	Weibo	WoW	**miss**
Fetion	100										
MSN		87		7		6					
Nmap			100								
ooVoo		2		93							**5**
QQ					100						
Renren		2				92					**6**
SkyDrive							100				
Skype								100			
Weibo		2		2					91		**5**
WoW										100	

In addition we also use two scan tools: *Nmap* and *Netenum*, which are commonly used by hackers, to simulate scan attacks against the gateway. After training, the system can also effectively detect the attacks, which is shown in Figure 14.

In order to check the performance of the system we use the detection system in the laboratory to test 10 network applications. After training, we carried out packet detection on each application and captured 100 groups for each application, a total of 1,000 sets of data. The 10 network applications consisted of Fetion, MSN, Nmap, ooVoo, QQ, the Renren desktop, SkyDrive, Skype, Sina microblogging (Weibo), and World of Warcraft (WoW), and the experimental results are shown in Table 8.

The rows indicate the determination results from the applications, numerical value denotes the situation of accuracy measured, and the columns regard the actual corresponding class of each detected data flow. In case of Renren application, the detection of wrong msn rate is $20/1000 = 2\%$, miss detection rate is $60/1000 = 6\%$, and the final detection accuracy rate is $920/1000 = 92\%$.

Through the above experiment, it can be verified that our system can use the established network behavioural patterns collection to monitor and record the behavior of the application on the user's network, as well as the system's undetected and false detection rates being ideal.

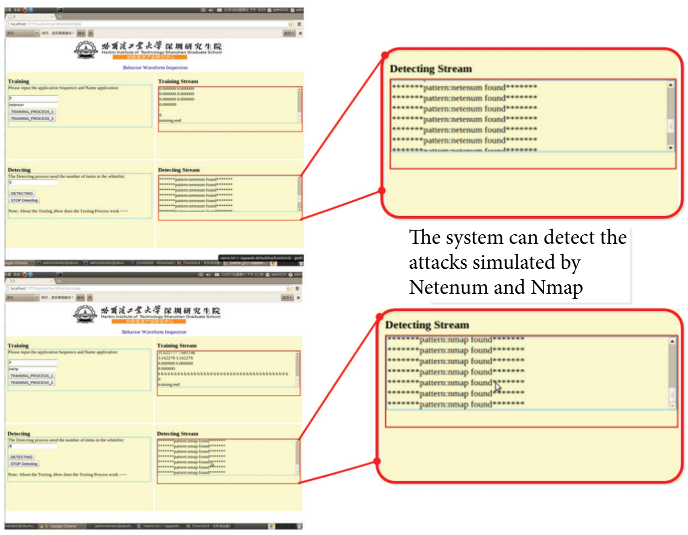
6.7. Compared Algorithm-Histogram Statistical Method. The histogram statistical methods will read the frame length information for the data packet and intercept the data frame within a certain time which may represent the characteristics of an attack or an application.

The statistical histogram is counted and generated according to the specific situation. After processing a large amount of data, the statistical interval division will be obtained according to the interval setting strategy in Section 5.

Due to the overbroad numerical range, the histogram distribution is very irregular, which is not favourable for the operator. In order to balance the histogram height, this study has conducted certain interval division of the histogram to obtain more appropriate nonuniform interval segments for calculation. By computing the original histogram of the data again to conduct the second-order histogram statistics, we can obtain the corresponding second order height balanced histogram.

First, obtain the total number of $\mathbb{m}$ data for the raw histogram information and the ideal histogram interval number $\mathbb{n}$ after processing. Then, increment the data from the initial value, and accumulate the occurrences of the value $\mathbb{c}_i$; that is,

$$\mathbb{m}_i := \sum_{j=0}^{i} \mathbb{c}_j, \tag{18}$$


Detecting Stream
The system can detect the attacks simulated by Netenum and Nmap
Detecting Stream

```
*******pattern:netenum found*******
*******pattern:netenum found*******
*******pattern:netenum found*******
*******pattern:netenum found*******
*******pattern:netenum found*******
*******pattern:netenum found*******
```

```
*******pattern:nmap found*******
*******pattern:nmap found*******
*******pattern:nmap found*******
*******pattern:nmap found*******
*******pattern:nmap found*******
*******pattern:nmap found*******
*******pattern:nmap found*******
```

FIGURE 14: Detection of simulated attacks.

where $\mathbb{m}_i$ represents the number of accumulated data which are less than or equal to the value i, and then record the value i that tends to meet the following conditions as the interval bounds:

$$\frac{\mathbb{m}_i}{\mathbb{m}} := \frac{1}{\mathbb{n}}, \frac{2}{\mathbb{n}}, \ldots, 1. \tag{19}$$

Thus the statistics of each interval can also be obtained by

$$\mathbb{p}_1 := \sum_{i=1}^{\mathbb{n}_1} \mathbb{c}_i, \qquad \mathbb{p}_2 := \sum_{i=1}^{\mathbb{n}_2} \mathbb{c}_i, \ldots. \tag{20}$$

So, in this way, it is possible to achieve effective histogram height equalization, for the utilization of the various kinds of arithmetics in the subsequent steps of the system.

After acquiring the histogram statistics, conduct pairwise comparisons and calculate the square of the difference of the two histograms as the basis of classification judgment. Euclidean distance, given by the Pythagorean formula, is used to measure the original distance between two points regulated by some kind of certain rules, and thus the formation of the Euclidean space has also become a metric space [30]. As the most common method of distance calculation, Euclidean distance can be directly and effectively used to reflect differences between objects. In this regard, this method puts forward a measuring algorithm by marking the waveform as the point in the multidimensional space and obtains the degree of matching by measuring the Euclidean distance between the points. This paper treats the points on the waveform as the coordinates of the characteristics in different dimensions, and the length of the waveform is the number of dimensions. In this way, each waveform can be denoted by a point in multidimensional space.

Thus the measurement of the difference between the waveforms is as follows:

$$t = \sum \left(x_i^2 - s_i^2\right), \tag{21}$$

where t is the difference, x_i denotes the value of each sample's interval, and s_i means the standard value of each interval. Then repeat the experiments to find the best judgment threshold.

6.8. Comparison of Results

6.8.1. Comparison of Three Methods. The histogram method when the matching effect is rough cannot reflect the impact of the time factor on its determination. The AC method is also not sensitive to the time factor, and both of them will generate a large error. However, the AC-histogram method greatly enhances the classification effect.

We selected several instant messaging applications, whose features of the data frames are very close to each other. They are MSN, ooVoo, Paltalk, Skype, and Yahoo messenger. We captured 10 groups of packets for each application and take one of them as the standard data. Then we use different methods of detection on them.

Figures 15, 16, and 17 show the experimental results. The first five rows show the differences between the 10 sets of data for the same category applications. There are 45 sets of data. The remaining rows identify the differences between different applications. There are 200 sets of data. (Simply, m represents MSN, o represents ooVoo, s represents Skype, p represents Paltalk, and y represents Yahoo messenger. Thus, m/s denotes the comparison of the experimental results between MSN and Skype, etc.)

6.8.2. Histogram Method. As described in the previous section on the intervals setting strategy, directly conduct statistical processing of the information on data frame length, as well as obtaining statistical histogram information. Then, directly calculate the Euclidean distance between the waveforms which will be regarded as the basis for classification determination.

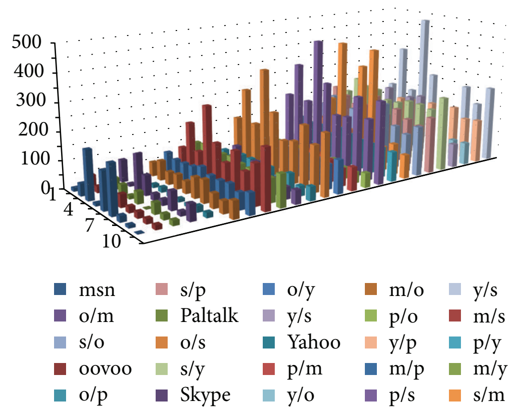


Figure 15: Experimental results of histogram method.

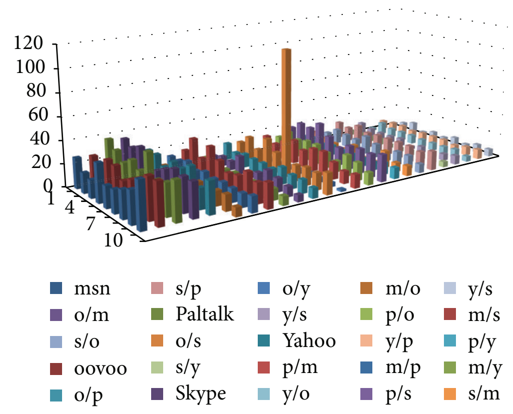


Figure 16: Experimental results of AC method.

In order to facilitate the comparison, the calculated distance did not conduct root operations. The experimental results are shown in Figure 15.

The histogram plotted in Figure 16 indicates the accuracy of the matching waveform: when the detecting waveform and the training waveform belong to one class, in fact the same application and the former five cylinders are generally low, which shows the extent of matching accuracy. Indeed, when matching with a different application's waveform, the result will be at a higher value of the distance between the waveforms. However, if the Euclidean distance between the two different application's waveform is also generally low, it is possible to make an incorrect classification.

In order to distinguish between similar applications with heterogeneous applications, we selected the appropriate threshold for the experimental design by a brute-force method to select the optimal threshold for the experiment.

Using the previously mentioned threshold selection method, we selected 28 as the threshold, and the error rate is 16/245 = 6.5%.

6.8.3. AC Waveform Method. Use the proposed algorithm to convert the packet frame length information into a string, and establish the AC automaton for the patterns by the fixed-length substring; then match and record the number of matches for each pattern. Finally, consider the total number of matching pattern strings for statistical test data as the basis for judgment. The experimental results are shown in Figure 16.

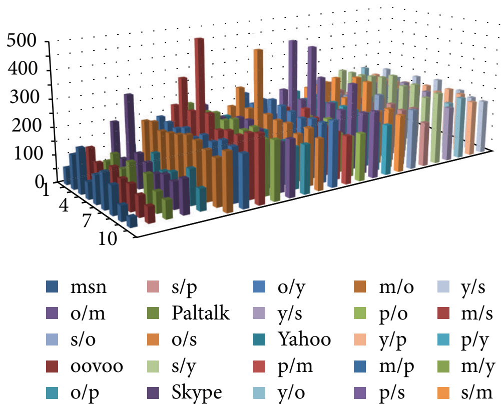


Figure 17: Experimental results of AC-matching and histogram method.

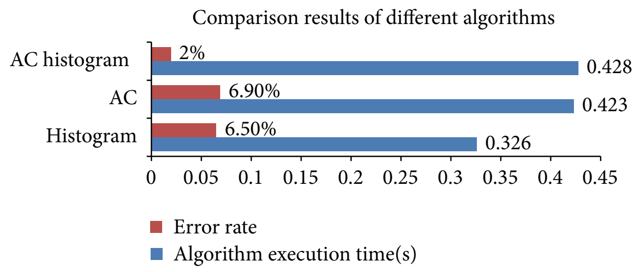


Figure 18: Comparison of different algorithms.

As can be seen in Figure 16, when matching with a similar application, the histogram cylinders plotted in the figure are generally high; but when matching with different applications, the number of matching patterns generally shows a low trend.

With previously mentioned threshold selection method, we selected 24 as the threshold, and the error rate is 17/245 = 6.9%.

6.8.4. AC-Matching and Histogram Method. TheAC-matching and histogram method, the advanced algorithm proposed in this paper, similarly uses the Euclidean distance between the waveforms as the basis for classification determination, and in order to facilitate the comparison, the calculated distance also did not conduct root operations. The experimental results are shown below.

First, the system will carry out frame length information quantification and use the AC automaton established in the training stage for pattern matching. Then, the histogram method is conducted to count the number of matches, and the Euclidean distance between the training and testing results obtained from the histogram method procedure is calculated. Thus the similar application shows a lower distance, and different applications show a larger distance. The utilization of the AC-matching and histogram method can get better

Table 9: Average execution times of different algorithms.

Methods	Histogram	AC	AC histogram
Algorithm execution time(s)	0.326	0.423	0.428
Error rate	6.5%	6.9%	2%

results in identifying the same application and can also show a high degree of distinction when matching different applications, as can be seen in Figure 17.

By the previously mentioned threshold selection method, we selected 129 as threshold, and the error rate is 5/245 = 2%.

Furthermore, the average execution times and error rates of each method are listed in Table 9. As can be seen from the experimental results, due to the consideration of the prioritization factors of the data packets, as well as the higher matching accuracy (Figure 18), the AC match-histogram method presents the best classification performance. The performances of the histogram method and AC method are close, but the degrees of differentiation are significantly lower than the AC match-histogram method. Furthermore, since the execution time of the AC-histogram method approximates to the two other methods, there is no need to concern the time-consuming problem of the algorithm.

7. Conclusion

With Internet and web attacks becoming more frequent and diverse, a new packet detection system with more diverse approaches is needed. In this paper we have constructed a new detection architecture using statistical-based waveforms. In the experiment we appropriately deal with the raw data to obtain the required waveform and then use the classification machine for training and testing.

By using the detection system raised by this study, on the one hand the efficiency of detection can be enhanced, and real-time requirements will be achieved. On the other hand, the concept of a detection whitelist and blacklist is proposed, which overcomes the defects in traditional anomaly detection and misuse detection to some extent. Meanwhile, this study uses statistics within a certain time window as the judgment basis, which increases the accuracy of the determination compared with conventional intrusion detection algorithms based on a single data frame. Compared with complete connection-based intrusion detection, it greatly enhances the convenience of the system and also provides satisfactory judgment accuracy.

Symbols and Descriptions

AC:	Aho-Corasick algorithm
i, j, k:	Serial number
ρ_i:	ith character in the text
$\mathrm{dist}[\rho_i]$:	Distance of the ith character needs to jump when match failure occurs in the matching algorithm
T:	Training
D:	Detecting
$\mathrm{App}^{(i)}$:	ith application
$S_T^{(i)}$:	ith training stream
$S_D^{(j)}$:	jth testing stream
$\mathrm{App}_{S_T}^{(i)}$:	Label of training stream $S_T^{(i)}$, which means the training stream truly belongs to $\mathrm{App}^{(i)}$
$\mathrm{App}_{S_D}^{(j)}$:	Label of testing stream $S_D^{(j)}$, which is unknown at the beginning
$(S_T^{(i)}, \mathrm{App}_{S_T}^{(i)})$:	Input of training stage
$(S_D^{(j)}, \mathrm{App}_{S_D}^{(j)})$:	Input of testing stage
$L_T^{(i)}$:	Extracted packet frame length information of the ith training stream
$L_D^{(j)}$:	Extracted packet frame length information of the jth test stream
$(L_T^{(i)}, \mathrm{App}_{S_T}^{(i)})$:	Input sample of data conversion in training stage
$(L_D^{(j)}, \mathrm{App}_{S_T}^{(j)})$:	Input sample of data conversion in testing/detecting stage
$\omega_T^{(i)}$:	Generated waveform of training stage for the ith application
$\omega_D^{(j)}$:	Generated waveform of testing stage for the jth application
$\delta[P_m]_T^{(i)}$:	A string array of group of short patterns generated in the data conversion process of the training stage for the ith application
$\delta[P_n]_D^{(j)}$:	A string array of group of short patterns generated in the data conversion process of the detecting stage for the jth application
$\phi_T^{(i)}$:	Established AC automaton for the ith application in the training stage of AC waveform method
$\psi_T^{(i)}$:	Established AC-histogram automaton for the ith application in the training stage of the AC waveform and histogram method
$\mathscr{H}_T^{(i)}$:	Established histogram for the patterns of the ith application in the training stage of the AC Waveform and histogram method
$\mathscr{H}_D^{(j)}$:	Established histogram for the matching patterns of the jth application in the detecting stage of the AC waveform and histogram method
$\mathscr{C}_{S_D^{(j)}}^{(i)}$:	Total number of matches of the jth detecting stream without application label when inputting it into the ith AC automaton, which is only used in the AC waveform method
P_m:	mth short pattern of the training stream with $m = 1, 2, 3, \ldots$

P_n:	nth short pattern of the testing stream with $n = 1, 2, 3, \ldots$
$\mathscr{C}[P_m]_T^{(i)}$:	An array to store the number of occurrences of each pattern for the training stream of the ith application
$\mathscr{C}[P_n]_{S_D^{(j)}}^{(i)}$:	An array to store the number of matches of each pattern for the detecting stream $S_D^{(j)}$ when inputting it into the ith AC-histogram automaton
$\mathbb{M}_{(j)}^{(i)}$:	Decision-making result of whether the detecting stream $S_D^{(j)}$ belongs to the ith application
$\mathbb{R}_{S_D^{(j)}}$:	Output of the detecting result of the detecting stream $S_D^{(j)}$
$Value\{condition(s)\}$:	Value that can be obtained if and only if the condition(s) inside the brackets is(are) met
$goto(\,)$:	goto function of AC algorithm
$fail(\,)$:	fail function of AC algorithm
$output(\,)$:	output function of AC algorithm
$\Gamma_e(\,)$:	Preprocessing procedure for data stream, including the extraction of the packet frame length information, and quantification
$\Lambda_\omega(\,)$:	Waveform generation process for the packet frame length information
$\Delta_c(\,)$:	Process to convert the waveform information into short patterns
$\Phi_e(\,)$:	Establishment of AC automaton
$M_c(\,)$:	Matching process by feeding the detecting data into AC(-histogram) automaton and as a function it will return with a result of true or false
$O_c(\,)$:	Process for counting the number of occurrences of each pattern in the establishment of the AC-histogram automaton
$\Psi_e(\,)$:	Establishment of the AC-histogram automaton
$H_e(\,)$:	Establishment of the histogram
d:	Calculation of Euclidean distance
$:=$:	Assignment operation
$=$:	Truth assertion
$\mathbf{1}$:	Decision-making of matching successful result
$\mathbf{0}$:	Decision-making of mismatch result
$\mathbb{m}$:	Total number of data for raw histogram information
$\mathbb{n}$:	Ideal histogram interval number after processing
$\mathbb{c}_i$:	Occurrences of the value
$\mathbb{m}_i$:	Number of accumulated data which is less than or equal to the value i
$\mathbb{p}_i$:	Statistics of the ith interval
μ:	Consolidated matching value of decision threshold
μ_i:	Match situation of each feature
w_i:	Weight of each feature
t:	Difference between waveforms
x_i:	Value of each sample's interval
s_i:	Standard value of each interval.

Conflict of Interests

The authors of the paper do not have a direct financial relation that might lead to a conflict of interest for any of the authors.

References

[1] H. Mousannif, I. Khalil, and G. Kotsis, "The cloud is not "there", we are the cloud!," *International Journal of Web and Grid Services*, vol. 9, no. 1, pp. 1–17, 2013.

[2] B. R. Chang, H. F. Tsai, and C. M. Chen, "Evaluation of virtual machine performance and virtualized consolidation ratio in cloud computing system," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 4, no. 3, pp. 192–200, 2013.

[3] S. Yoon, B. Kim, J. Oh, and J. Jang, "High performance session state management scheme for stateful packet inspection," in *Managing Next Generation Networks and Services*, vol. 4773 of *Lecture Notes in Computer Science*, pp. 591–594, Springer, Berlin, Germany, 2007.

[4] V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, New York, NY, USA, 1998.

[5] M. Fugate and J. R. Gattiker, "Anomaly detection enhanced classification in computer intrusion detection," in *Pattern Recognition with Support Vector Machines*, vol. 2388 of *Lecture Notes in Computer Science*, pp. 186–197, Springer, Berlin, Germany, 2002.

[6] D. S. Kim and J. S. Park, "Network-based intrusion detection with support vector machines," in *Information Networking*, vol. 2662 of *Lecture Notes in Computer Science*, pp. 12–14, Springer, Berlin, Germany, 2003.

[7] L. Xie, D. Zhu, and H. Yang, "Research on SVM based network intrusion detection classification," in *Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD '09)*, vol. 7, pp. 362–366, Tianjin, China, August 2009.

[8] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '02)*, pp. 1702–1707, Honolulu, Hawaii, USA, May 2002.

[9] W. Hu, Y. Liao, and V. R. Vemuri, "Robust support vector machines for anomaly detection in computer security," in *Proceedings of the International Conference on Machine Learning and Applications (ICMLA '03)*, pp. 168–174, CSREA Press, 2003.

[10] D. S. Kim, H.-N. Nguyen, S.-Y. Ohn, and J. S. Park, "Fusions of GA and SVM for anomaly detection in intrusion detection system," in *Advances in Neural Networks—ISNN 2005*, J. Wang, X. Liao, and Z. Yi, Eds., vol. 3498 of *Lecture Notes in Computer Science*, pp. 415–420, Springer, Berlin, Germany, 2005.

[11] J. Cannady, "Artificial neural networks for misuse detection," in *Proceedings of the National Information Systems Security Conference*, pp. 368–381, 1998.

[12] R. P. Lippmann and R. K. Cunningham, "Improving intrusion detection performance using keyword selection and neural networks," *Computer Networks*, vol. 34, no. 4, pp. 594–603, 2000.

[13] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, and M. Embrechts, "Network-based intrusion detection using neural networks," in *Proceedings of the Intelligent Engineering Systems through Artificial Neural Networks*, vol. 12, pp. 579–584, November 2002.

[14] I. Lorenzo-Fonseca, F. Maciá-Pérez, F. J. Mora-Gimeno, R. Lau-Fernández, J. A. Gil-Martínez-Abarca, and D. Marcos-Jorquera, "Intrusion detection method using neural networks based on the reduction of characteristics," in *Bio-Inspired Systems: Computational and Ambient Intelligence*, J. Cabestany, F. Sandoval, A. Prieto, and J. M. Corchado, Eds., vol. 5517 of *Lecture Notes in Computer Science*, pp. 1296–1303, Springer, Berlin, Germany, 2009.

[15] M. Moradi and M. Zulkernine, "A neural network based system for intrusion detection and classification of attacks," in *Proceedings of the IEEE International Conference on Advances in Intelligent Systems-Theory and Applications*, pp. 15–18, Kirchberg, Luxembourg, November 2004.

[16] G. Wang, J. Hao, J. Mab, and L. Huang, "A new approach to intrusion detection using artificial neural networks and fuzzy clustering," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6225–6232, 2010.

[17] F. Jensen, *Bayesian Networks and Decision Graphs*, Springer, New York, NY, USA, 2001.

[18] P. Spirtes, T. Richard-Son, and C. Meek, "Learning Bayesian networks with discrete variables from data," in *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining*, pp. 294–299, 1995.

[19] D. Bulatovic and D. Velasevic, "A distributed intrusion detection system based on bayesian alarm networks," in *Secure Networking—CQRE [Secure] '99*, vol. 1740 of *Lecture Notes in Computer Science*, pp. 219–228, Springer, Berlin, Germany, 1999.

[20] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," in *Proceedings of the 19th Annual Computer Security Applications Conference*, pp. 14–23, Las Vegas, Nev, USA, December 2003.

[21] D. J. Burroughs, L. F. Wilson, and G. V. Cybenko, "Analysis of distributed intrusion detection systems using Bayesian methods," in *Proceedings of the 21st IEEE International Performance, Computing, and Communications Conference*, pp. 329–334, Phoenix, Ariz, USA, April 2002.

[22] A. Cemerlic, L. Yang, and J. M. Kizza, "Network intrusion detection based on bayesian networks," in *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE '08)*, Redwood City, Calif, USA, 2008.

[23] D. E. Knuth, J. H. Morris, and V. R. Pratt, "Fast pattern matching in strings," *SIAM Journal on Computing*, vol. 6, no. 2, pp. 323–350, 1977.

[24] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM*, vol. 20, no. 10, pp. 762–772, 1977.

[25] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

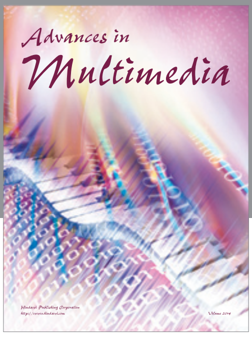
[26] M. Alicherry, M. Muthuprasanna, and V. Kumar, "High speed pattern matching for network IDS/IPS," in *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP '06)*, pp. 187–196, Santa Barbara, Calif, USA, November 2006.

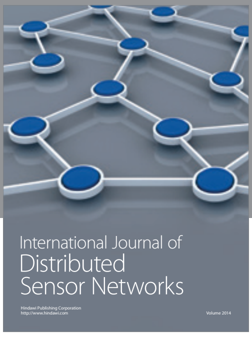
[27] T. Ambwani, "Multi class support vector machine implementation to intrusion detection," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 3, pp. 2300–2305, July 2003.

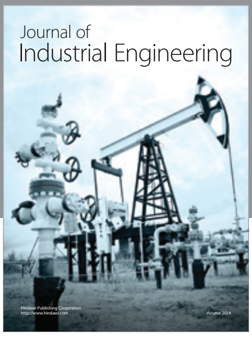
[28] S. M. Bellovin, "Security problems in the TCP/IP protocol suite," *Computer Communication Reviews*, vol. 19, no. 2, pp. 32–48, 1989.

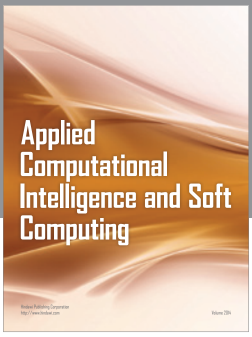
[29] V. Das, V. Pathack, S. Sharma, M. V. V. N. S. Srikanth, and G. T. Kumar, "Network intrusion detection system based on machine learning algorithms," *International Journal of Computer Science & Information Technology*, vol. 2, no. 6, pp. 138–151, 2010.

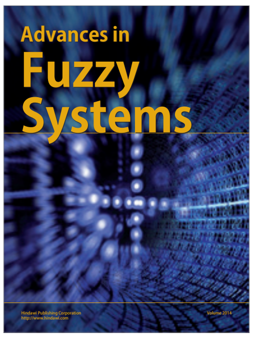
[30] P.-E. Danielsson, "Euclidean distance mapping," *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 227–248, 1980.

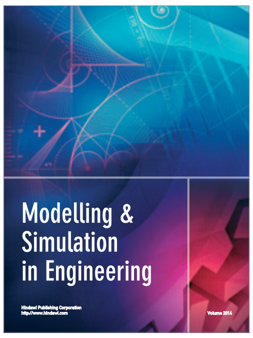












Hindawi

Submit your manuscripts at
http://www.hindawi.com