CBF: A Packet Filtering Method for DDoS Attack Defense in Cloud Environment

Qi Chen, Wenmin Lin, Wanchun Dou *
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
e-mail: adios737@gmail.com, linwenmin_cn@126.com, douwc@nju.edu.cn *

Shui Yu
School of Information Technology
Deakin University
Burwood, Australia
e-mail: syu@deakin.edu.au

***Abstract*—Distributed Denial-of-Service attack (DDoS) is a major threat for cloud environment. Traditional defending approaches cannot be easily applied in cloud security due to their relatively low efficiency, large storage, to name a few. In view of this challenge, a Confidence-Based Filtering method, named CBF, is investigated for cloud computing environment, in this paper. Concretely speaking, the method is deployed by two periods, i.e., non-attack period and attack period. More specially, legitimate packets are collected at non-attack period, for extracting attribute pairs to generate a nominal profile. With the nominal profile, the CBF method is promoted by calculating the score of a particular packet at attack period, to determine whether to discard it or not. At last, extensive simulations are conducted to evaluate the feasibility of the CBF method. The result shows that CBF has a high scoring speed, a small storage requirement and an acceptable filtering accuracy, making it suitable for real-time filtering in cloud environment.**

Keywords-Distributed Denial-of-Service Attack; Filtering; Confidence; Correlation Pattern; Network Security; Cloud Environment

I. INTRODUCTION

A. Current Status of Related Research

Cloud computing is a long-held dream of computing as a utility. As discussed in [1], it has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Nowadays, it is evolving as a key computing platform for sharing resources including infrastructure resources, software resources, application resources and business processes [2]. However, with large amount of resources online, these cloud systems are facing severe security problems.

Distributed Denial-of-Service (DDoS) attack can be considered as a major threat to cloud computing. The attackers often compromise vulnerable hosts, called *zombies*, on the network and install attack tools on them. These zombies together form a *botnet* and will generate large amout of distributed attack packets targeting at the victims under the control of the attackers. This attack will block the legitimate access to the servers, exhaust their resources such as network bandwidth, computing power and even lead to great financial losses as shown in [3].

In recent years, many researches on DDoS defense have been carried out and many successful schemes have been put forward. There are approximately three major branches of the research: attack detection [4] [5] [6], attack filtering [7] [8] [9] [10] [11] [12], and attack traceback [13] [14] [15].

As mentioned in [7], the branch of attack filtering can be roughly categorized into three areas based on the point of protection: source-initiated, path-based and victim-initiated. The method proposed in this article is in victim-initiated area, which filters incoming attack packets from victim side. In this area of research, a number of brilliant approaches have already been proposed.

PacketScore [7] generates value distributions of some attributes in the TCP and IP headers, and then uses Bayes' Theorem to score packets. PacketScore has a pretty high filtering accuracy and it is also easy to be deployed. But since its scoring and discarding are related to attack intensity, it is not suitable for handling large amount of attack traffic. Also it has some costly operations in scoring, which leads to low process efficiency in real-time filtering.

ALPi [8] is an improvement of PacketScore. Two schemes LB and AV which uses leaky buckets and value variances of attributes respectively are proposed and are evaluated by comparison with PacketScore. Hop-Count Filtering (HCF) [9] uses the relationship of source IP address and TTL value to carry out filtering. After building an IP to hop-count mapping, it can detect and discard spoofed IP packets with about 90% accuracy. It is effective and easy to be deployed but it is vulnerable to distributed attacks because of its assumption about spoofed IP traffic.

Our method aims at mining the correlation patterns, which refer to some simultaneously-appeared characteristics in the legitimate packets. [16] [17] use the document popularity and user browsing behaviors to detect attack packets, which reflect some correlation patterns between packets in a flow. But these patterns are mainly in application layer, making these methods mostly effective for application layer DDoS.

B. Motivation

Considering more and more resources being shared in cloud platforms, especially in an elastic cloud environment which could nearly provide unlimited capabilities [18], the effect of DDoS attacks can be not only much more powerful and influential, but also in much wider range. In view of this challenge, this paper aims at proposing a method for cloud security, which means to be much quicker in responding, easier to be widely deployed and more powerful in ability than before.

C. The Organization of the Paper

This paper is organized as follows: In Section 2, we will introduce some basic concepts and give an overview of our method, Confidence-Based Filtering. In the next three sections, we focus on the details of some important parts in the method. In Section 3, the nominal profile structure along with a feasible storage saving strategy is discussed. In Section 4 and 5, we talk about the score calculation details and the discarding strategy in filtering. In Section 6, the performance of our method under different types of DDoS attacks are evaluated based on real world traffic. Section 7 discusses some important issues about the ability of the method, and then Section 8 gives a brief conclusion and the direction of future work.

II. An Overview of Confidence-Based Filtering Method

A. Basic Concepts

1) Key Terms

To help illustrate our method, some key terms used in this paper are summarized in Table I.

2) Correlation Pattern

In order to discriminate attack packets from legitimate ones, the method proposed in this paper utilizes correlation patterns. The concept of correlation refers to the situation that some interior characteristics take places at the same time in the packet flows. So the basic assumption of this method is that there are indeed some unique correlation patterns in legitimate packet flows. Fortunately, this assumption can be valid in most circumstances. In user browsing behaviors,

TABLE I. Key Terms Appeared in This Paper

Terms	Description
n	The number of the attributes under consideration in the method
A_i	The i-th attribute in the packet, $(1 \le i \le n)$
m_i	The number of values which attribute A_i can have
$a_{i,j}$	The j-th value of attribute A_i, $(1 \le j \le m_i)$
t	A time interval in packet flows
N_n	The total number of packets in the packet flow in one time interval t
$N(A_i = a_{i,j})$	The number of packets whose attribute A_i has value $a_{i,j}$ in this packet flow in one time interval t
$N(A_r = a_{r,x}, A_s = a_{s,y})$	The number of packets whose attribute A_r has value $a_{r,x}$, attribute A_s has value $a_{s,y}$ in this packet flow in one time interval t
p	A packet in the packet flows
$p(i)$	The value of attribute A_i in packet p

when a person logs on a certain website, his/her focuses tend to make up a certain pattern. For example, since the majority of NBA fans who live in Los Angeles love the team Los Angeles Lakers, the website of ESPN will have more packets containing correlations between visits of Lakers webpage and the IP addresses from the area around Los Angeles. Considering that there are a large amount of correlation patterns like this or even more complicate ones, it is quite hard for attackers to notice and mimic these patterns when carrying out DoS or DDoS attacks. Thus, using this kind of patterns to judge the legitimacy of packets can be feasible.

In this method, we focus our probe on transport and network layers. The correlation patterns in these two layers are the co-appearances between attributes in IP header and TCP header. These attribute pair patterns are distinctive because certain characteristics of the operating system, network structure and even hobbies of users can affect the values of these attributes, and thus make some attribute pairs related. In [9], the hop-count filtering constructs an IP2HC table which maps source IP addresses to TTL values, and filters attack packets by checking the validation of TTL according to the source IP address. It can be seen that the key point of its success is utilizing the correlation pattern between TTL and source IP address. So it is reasonable to generalize this idea to all correlation patterns between attributes in IP header and TCP header.

3) Confidence and CBF Score

In this part, we will first introduce two concepts: the one named confidence for measuring correlation patterns, and the one named CBF score for judging the legitimacy of packets. With the concept of CBF score, we will define the CBF legitimacy of a packet.

The concept of confidence reflects how much trust we can put on a correlation pattern between an attribute pair. In this paper, we define it formally as follows,

Definition 1. (Confidence): Confidence is the frequency of appearances of attributes in the packet flows. The confidence (*Conf* for short) for single attributes and attribute pairs are calculated as (1) and (2),

Confidence for single attributes:

$$Conf(A_i = a_{i,j}) = \frac{N(A_i = a_{i,j})}{N_n} \quad (1)$$

, where $i = 1, 2, 3, \ldots, n, j = 1, 2, 3, \ldots, m_i$,

Confidence for attribute pairs:

$$Conf(A_{i_1} = a_{i_1,j_1}, A_{i_2} = a_{i_2,j_2}) = \frac{N(A_{i_1} = a_{i_1,j_1}, A_{i_2} = a_{i_2,j_2})}{N_n} \quad (2)$$

, where $i_1 = 1, 2, 3, \ldots, n$, $i_2 = 1, 2, 3, \ldots, n$, $j_1 = 1, 2, 3, \ldots, m_1$, $j_2 = 1, 2, 3, \ldots, m_2$,

In (1) and (2), the meanings of the variables are listed in Table I.

Indicated by Def.1, the more times an attribute pair appears in the legitimate packet flows, the higher confidence value of this pair we can get. The concept of confidence is the basis of the calculation of CBF score and the whole filtering process, so we name our method Confidence-Based Filtering, CBF for short.

With confidence values of attribute value pairs, the legitimacy criterion of a packet is defined as follows,

Definition 2. (CBF Score): CBF score for a packet is the weighted average of the confidence of the attribute value pairs in it. The CBF score for a packet p is calculated as (3):

$$Score(p) = \frac{\sum_{k=1}^{d} W(A_{k_1}, A_{k_2}) Conf(A_{k_1} = p(k_1), A_{k_2} = p(k_2))}{\sum_{k=1}^{d} W(A_{k_1}, A_{k_2})} \quad (3)$$

In this definition, d is the total number of the attribute pairs involved in the calculation of score. A_{k_1} and A_{k_2} are two attributes in the k-th attribute pair. $W(A_{k_1}, A_{k_2})$ is the weight for the k-th attribute pair. Considering the range of each confidence value is in [0, 1], the range of $Score(p)$ is also in [0, 1].

Thus, in order to calculate CBF scores of packets, we need to prepare the confidence of each attribute value pair in legitimate packet flow beforehand. In our method, we design a dataset for these confidence values, named *nominal profile*. The generating details of it are discussed in Section 3.

In (3), the attribute pairs which cannot be easily copied by attackers will be given a high weight. Thus, higher score of a packet corresponds to more frequently-appeared and difficultly-copied correlation patterns, and thus more likely to be legitimate. So we can choose a *discarding threshold* to make the judgment of filtering. In view of this, the legitimacy of the packets is defined as follows,

Definition 3. (CBF Legitimate Packet): The legitimate packet in CBF is the one whose CBF score is above the *discarding threshold*.

So on the contrary, those packets with scores lower than the *discarding threshold* are regarded as attack ones.

B. Confidence-Based Filtering

The overall process of CBF method can be divided into two periods: non-attack period and attack period. An outline of our method is shown in Fig.1. The details of it will be introduced in the following sections.

At non-attack period, the main target is to generate nominal profile. For incoming packets, our method firstly extracts the needed attribute value pairs from them. With (2) in Def.1, the number of appearances of these value pairs will be counted and their confidence values are calculated. Then these confidence values are used to update nominal profile.

At attack period, most packets are not legitimate, so CBF will stop generating nominal profile. Like that at non-attack period, extracting the attribute value pairs from the incoming packets is the first step. With these value pairs, our method searches nominal profile for their confidence values in legitimate flows. Then CBF score, the filtering criterion, is calculated using (3) in Def.2. After a packet discarding strategy is selected, CBF will judge the legitimacy of the packet based on Def.3, and decide to let it pass or not.

III. GENERATING NOMINAL PROFILE

A. Nominal Profile Structure

In this part, we will introduce the structure of nominal profile. Firstly, we select six candidate single attributes as shown in Table II. Then, we combine every two (not the same) of the six attributes and get 15 attribute pairs. After combination, the values of attribute pair will have 32-bit sizes since the 6 single attributes all have the sizes of no more than 16-bit. Table III shows an example of the nominal profile structure which contains two attribute pairs (TTL, packet size) and (TTL, source IP address).

The overall constructing of the nominal profile is divided into small time intervals, which are called windows. The size of a window can be set to fixed ones or dynamic ones. In each time interval t, our method CBF counts the number of the value appearances of these 15 attribute pairs, and then use Def.1 to calculate the confidence values. At the end of each time interval, the new confidence values are used to update the nominal profile. In order to minimize the false negative rate, the highest confidence value of an attribute value pair in the nominal profile is stored, which means the updating only takes place when the new confidence value is higher than the one stored in the nominal profile.

TABLE II. SINGLE ATTRIBUTES SELECTED FROM IP/TCP HEADER

Location	Attribute	Description
IP Header	Total Length	The length of the datagram, measured in octets, including internet header and data.
	Time to Live (TTL)	The maximum time the datagram is allowed to remain in the internet system
	Protocol Type	The next level protocol used in the data portion of the IP datagram
	Source IP Address	The destination IP address (our method uses the 16-bit prefixes of it)
TCP Header	Flag	Control bits that indicate different connection states or information about how a packet should be handled
	Destination Port Number	The destination port number

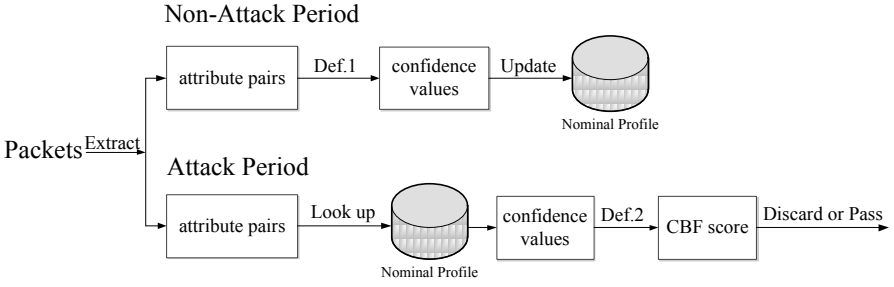


Figure 1. Outline of Confidece-Based Filtering

TABLE III. EXAMPLE OF NOMINAL PROFILE WITH TWO ATTRIBUTE PAIRS

Attribute value pair		TTL, packet size		TTL, TCP flag	
1	1	TTL=1, packet size=1	0.1%	TTL=1, TCP flag =1	0.01%
1	…	TTL=1, packet size=…	…	TTL=1, TCP flag =…	…
1	255	TTL=1, packet size=255	1.5%	TTL=1, TCP flag =255	1.2%
2	1	TTL=2, packet size=1	0.1%	TTL=2, TCP flag =1	0.05%
…	…	…	…	…	…
255	255	TTL=255, packet size=255	0.3%	TTL=255, TCP flag =255	0.08%

B. *Profile Storage Saving*

In order to construct the nominal profile, CBF calculates the confidence values of every attribute value pairs and stores them in certain data structure. However, this may incur storage problem. The common strategy for storing them will use a 3-dimension array. The first dimension is for attribute pair and has the length of 15. The second dimension is the value set of certain attribute pair, which has 32-bit size. The third dimension is the confidence value dimension and the size of it depends on the precision requirement of confidence values. If we use 32-bit for the third dimension, the overall needed storage will be $15\times2^{32}\times4$ bytes, which equals to 240 Gbytes. This amount of storage cannot be feasible in practice.

For this problem at storing step, we can use iceberg-style profiles [21]. In this implementation, we only store the confidence values of attribute value pairs which are higher than a predetermined threshold, e.g., 0.001 percent. We call this threshold *minconf*, which means the minimum confidence value in the nominal profile. In this way, the size of data needed to store is cut down shapely. With the usage of hash functions to search and store them, this storage problem seems to be successfully solved. However, this cannot be the complete solution. At the counting step, we also need the same size of storage to prepare spaces for counting the attribute pair value appearances. The 240 Gbytes counting space will also not be affordable.

The other part of the solution is to generate confidence of attribute value pairs by confidence of single attribute values. If the confidence of one attribute value in an attribute value pair is not greater than *minconf*, the confidence of the combination of this value pair will still not be greater than *minconf*. So we can firstly count the number of appearances of single attribute values and calculate the confidence of them using (1) in Def.1. Then we get the candidate attribute value pairs from the combination of only single attribute values whose confidence values are greater than *minconf*. So at the counting step, we will only prepare storage spaces for the candidate attribute value pairs instead of all possible ones. So at the last step, we select the attribute value pairs in the candidate ones whose confidence values are higher than *minconf* to update the iceberg-style profile.

Table IV shows the storage requirements for 3 minutes data in a trace recorded in WAWI Traffic Archive [19]. We set the window size to 5 seconds and use 32-bit to store each confidence value and each counting space. For measuring storing period storage, we count the number of confidence values which are actually stored in iceberg-style profile after processing all 3 minutes data. For counting period, we calculate the average number of needed counting spaces for candidate attribute value pairs in each window. The result in the table shows that even using extremely low *minconf* like 0.00005, the storage usage at storing period and counting period will not exceed 1 Mbyte, which is much less than 240 Gbytes. And the storage requirement of a proper *minconf* like 0.001 is around 8 Kbytes at storing period and 4.5 Kbytes at counting period, which is feasible in most cases.

TABLE IV. PROFILE STORAGE REQUIREMENTS FOR DIFFERENT *MINCONF* VALUES AT STORING AND COUNTING PERIOD

minconf	Storing Period		Counting Period	
	Number of confidence values	*Size of confidence values (Kbyte)*	*Average number of counting spaces*	*Size of counting spaces (Kbyte)*
0.01	177	0.691	175.393	0.685
0.001	2213	8.645	1138.607	4.448
0.0005	5242	20.477	2100.714	8.206
0.0001	54120	211.406	9080.893	35.469
0.00005	210900	823.828	15978.429	62.416

This sharp cutting down in storage also indicates that the frequently-appeared attribute value pairs only make up a small share of all possible value pairs, thus they build up valid patterns for filtering. As shown in Section 6, a *minconf* around 0.0005 can be effective in filtering. So the core storage size for CBF is only about 20 Kbytes, which makes it easy to be deployed in cloud platforms.

C. *Non-Attack Period Process Details*

Step 1: Count the number of appearances of single attribute values and then calculate the confidence of them;
Step 2: Select the single attribute values with confidence higher than *minconf* to generate the candidate attribute value pairs;
Step 3: Scan the packet flow for the second time to count the number of appearances of the candidate attribute value pairs and calculate their confidence; Then use the confidence values which are higher than *minconf* to update the nominal profile;

Figure 2. Details of CBF in one window at non-attack period

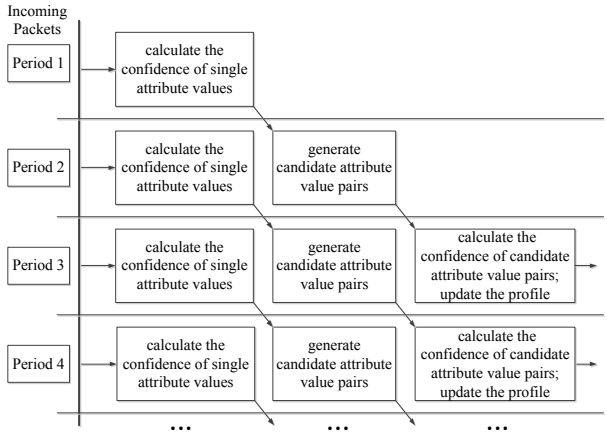


Figure 3. Pipeline implementation time line for CBF at non-attack period

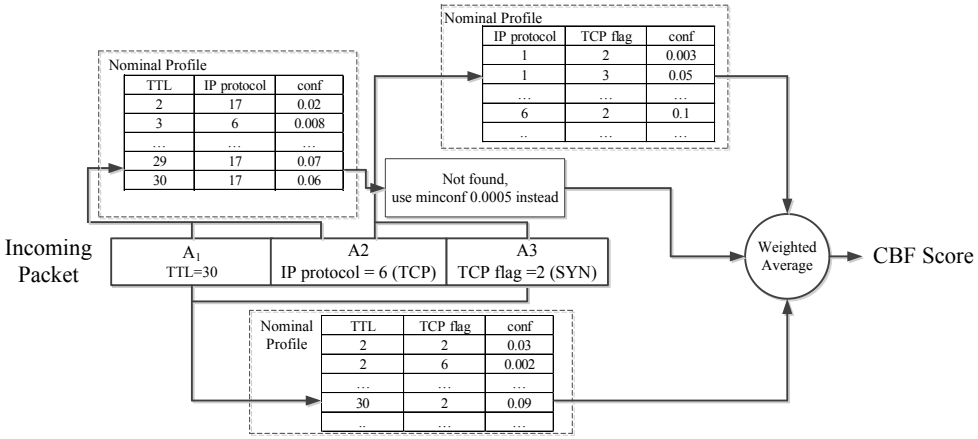


Figure 4. Example of the scoring process in CBF

Based on the solution of storage saving, a more specified process at non-attack period during one time window is described in Fig.2.

This is a 3-pass process and it can be largely accelerated if being carried out in parallel. As shown in Fig.3, the single attribute value counting, the candidate attribute value pair generating and the second time scanning can be put in a pipeline implementation, which will make CBF more suitable for real-time filtering and cloud computing.

IV. CALCULATING CBF SCORE

Indicated in Fig.1, in attack period CBF will firstly look up the nominal profile for the confidence values corresponding to the attribute value pairs in the current packets and then calculate the scores for them. In most cases, the confidence values of frequently-appeared attribute value pairs will be found in nominal profile successfully. But considering that we use iceberg-style profile, the confidence of some rarely-appeared attribute value pairs will be absent. In this case, we will use *minconf* value instead when these confidence values are required in score calculation.

The adjustments of the attribute pair weights will take into consideration the unique characteristics of the operating system, the network structure and many other elements. The general idea is to make more outstanding the correlation patterns which are less possible to be copied by attackers and more related to the inherent features of the server. For example, when under a denial-of-service attack, the source IP address in a packet is spoofed in most cases. So we can give the attribute pairs including source IP address a higher weight. On the other hand, we can give the attribute pairs including protocol type or TCP flag a lower weight because the ranges of their values are limited, thus it is easy for attackers to guess.

Fig.4 gives an example of the scoring process. In this figure, we assume that only 3 single attributes are involved in CBF filtering, which are TTL, IP protocol and TCP flag respectively. The scoring process starts from looking up the confidence of attribute value pairs in nominal profile. Because of the iceberg-style storing, we cannot find the confidence of the value pair in which TTL is 30 and IP protocol is 6. So we use *minconf* to represent its possible confidence value. Then a weighted average calculation is carried out with these confidence values to generate the CBF score for this incoming packet. If the weights for (TTL, IP protocol), (IP protocol, TCP flag) and (TTL, TCP flag) are 5, 1 and 3, the CBF score for the packet in the example is given by

$$\frac{(5\times 0.0005+1\times 0.1+3\times 0.09)}{(5+1+3)}=0.0414 .$$

The scoring part of CBF only requires a few looking-ups in nominal profile and some arithmetic operations. The asymptotic time complexity of CBF at this period is in O(1), so it will be fast enough even if large amount of packets burst in when under denial-of-service attack.

V. DISCARDING STRATEGY

After the CBF scores of packets are generated, we will use them to distinguish attack packets from legitimate ones. According to Def.3, CBF will only accept the packets with scores greater than the *discarding threshold*. Thus for the example in Fig.4, if the *discarding threshold* is 0.03, the packet will be judged legitimate. On the other hand, if it is 0.05, the packet will be an attack one.

The *discarding threshold* can be fixed based on the CBF score distribution of legitimate packets. According to Def.2, the CBF score is independent from the utilization of the victim, so the fixed *discarding threshold* is feasible if the distribution of the scores is known. And the processing speed will be very high with a fixed *discarding threshold*.

Also dynamic *discarding threshold* can be adopted. Like the load-shedding algorithm used in [20], we can use current utilization of the victim and the maximum utilization to generate the amount (Φ) of suspicious traffic that needs to be discarded. We can generate the cumulative distribution function (CDF) of the scores in current time window and decide the *discarding threshold* using Φ. However, this may incur the additional scores counting and CDF computing, which will be slightly slower than a fixed one.

VI. PERFORMANCE EVALUATION

In this section, we will use real world statistics to test the filtering method CBF. The data in the MAWI Traffic Archive [19] is adopted and the test environment is a 2.26 GHz Intel Core 2 Duo processor with 2 Gbytes Memory. The simulation programs were written in C++. We will firstly introduce the simulation conditions including the data source, the parameter selection for the method and different attack types. The result is shown and analyzed by taking into consideration of the comparison with PacketScore [7].

A. Simulation Conditions

1) Data

We select the data from MAWI Working Group Traffic Archive [19]. The part of data used in this section is collected from 14:00:00 to 14:15:00 on Jan 1, 2006. There are about 6587564 packets (2395.28Mbytes) contained in this data set and the average rate is 22.33Mbps. Every second, the data set has around 6000 to 7000 packets.

2) Parameters

The window size is set to 5 seconds, and the value of *minconf* is set to 0.005. Under this circumstance, the storage will be around 20 Kbytes at storing period and 8 Kbytes at counting period, which is affordable in normal servers. Our method spends around 0.4 seconds to process data during each time window. We believe this time can be minimized sharply after using pipeline implementations shown in Fig.3 and optimizations of the programs.

The weights in score calculation are set higher in the attribute pairs containing source IP address, TCP server port number or TTL value, and set lower in those only with TCP flag, IP protocol type and packet size. For the fast response at attack period, fixed *discarding threshold* is adopted.

In the comparison with PacketScore, the window size of PacketScore is set to 5 seconds and the threshold for iceberg-style profile is 0.01. In our implementation, after discarding percentage is selected by a load-shedding algorithm [20], CDF is used to calculate the discarding threshold of the score. We use the same six single attributes shown in Table II like those in CBF to carry out PacketScore filtering.

3) Attack Types

In this evaluation, we simulate the following types of attacks:

a) Generic attack

All attributes in the attack packets are selected randomly in their allowable ranges.

b) TCP-SYN Flood attack

The TCP SYN flag is set in each attack packets and the packet lengths of them are set to be 40. Other attributes are selected randomly.

c) SQL Slammer Worm attack

The IP protocol type is UDP, the destination port is set to 1434 and the packet size is between 371 to 400 bytes. Other attributes are selected randomly.

d) Nominal attack

Every attributes in the packets are selected randomly in smaller value ranges, which contain the most frequently-appeared values of this attribute at non-attack period. This attack supposes that the attackers know the value distributions of the single attributes and mimic it to carry out attack.

e) Mixed attack

In this attack, the attack type of each packet will be selected randomly from the four types above.

The score calculating and packet discarding of CBF are not affected by the intensity of the attack and the changing frequency of the attack types. Thus in this evaluation, we will not largely focus the tests of CBF on changing the type and intensity of attacks like [7] and [8].

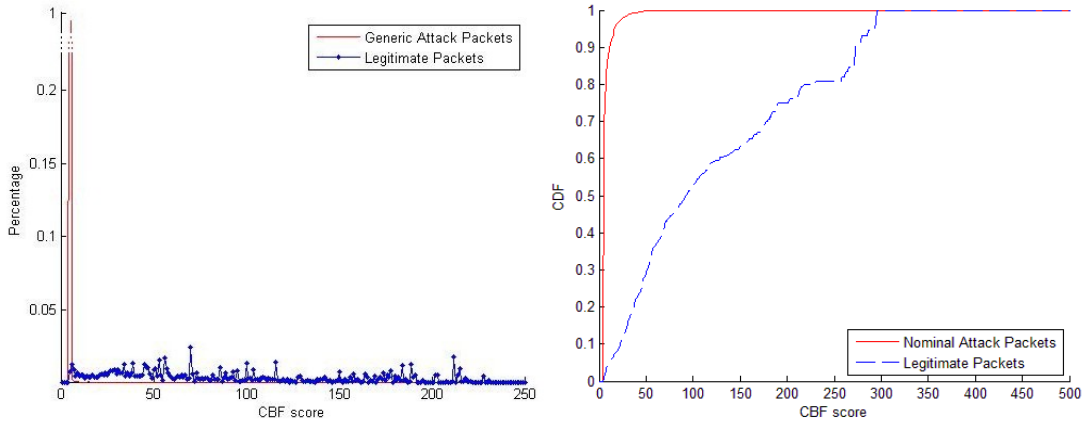
B. Simulation Result and Analysis

Fig.5 (a) shows the score distribution of generic attack and the legitimate flow using more than 100,000 packets data. To avoid the trouble with decimal scores, we multiply the original CBF scores with 10,000 when shown in the graph. Since the legitimate attribute pair patterns cannot be easily copied, most generic attack packets only have scores which consists of basic confidence *minconf*, 0.0005. For legitimate packets, high scores around 20 to 100 take place because they have more frequently-appeared attribute value pairs. Fig.5 (b) shows the cumulative distribution function (CDF) of the CBF scores of nominal attack and the legitimate flow (generic attack is not chosen here because its CDF curve is too steep to see a clear distribution). It illustrates more clearly that the majority of attack packets are concentrated in the low-score region.

To evaluate CBF in a more quantified way, we will test its performances of false positive (FP) rate and false negative (FN) rate when filtering. CBF and the classic scheme PacketScore are both filtering methods, both use attributes in TCP and IP headers to build nominal profile and both score packets to distinguish attack ones from legitimate ones. So we pick up it to make comparison when analyzing the ability of CBF.

Table V shows the result of their performances. The *discarding threshold* values for discarding in CBF are chosen to make the best performance among all possible ones. Since the CBF scores are not affected by attack intensity, the FP and FN rates are almost the same when there are 5 times and 10 times amount of attack packets than normal.

In most cases, these two methods share similar filtering abilities. In generic attack, CBF has a lower false positive rate because it is quite hard to generate the accurate attribute value pairs in random approach. In false negative rate, PacketScore has a better performance in SQL slammer worm attack but a worse one in mixed attack.



(a) (b)

Figure 5. CBF score distribution of attack flow and legitimate flow

TABLE V. THE PERFORMANCE OF CBF AND PACKETSCORE UNDER DIFFERENT ATTACK TYPES

Attack type	Attack Intensity	False Positive Rate(%)		False Negative Rate(%)	
		CBF	*PacketScore*	*CBF*	*PacketScore*
Generic	5×	0.513	3.266	0.695	0.0173
	10×	0.516	1.729	0.692	0.0432
TCP-SYN Flood	5×	7.701	3.571	7.775	1.249
	10×	7.703	1.956	7.770	1.542
SQL Slammer Worm	5×	1.521	3.473	3.883	0.000
	10×	1.524	1.988	3.881	0.000
Nominal	5×	5.229	5.032	6.925	9.519
	10×	5.234	2.929	6.915	13.462
Mixed	5×	4.564	4.771	6.524	7.601
	10×	4.565	2.653	6.524	9.543

In TCP-SYN flood, the performance of CBF has some degradation. It results from the situation that the TCP-SYN packets may be also frequent in legitimate time. But the approximate 7.7 percent false positive rate and false negative rate can also be considered as an effective filtering in practice.

PacketScore has a worse performance in false negative rate in nominal attack compared to our method. This is because we assume the attackers have the information of the single attribute value distributions in this attack. For CBF, its filtering can be ineffective if the attackers find the correlation patterns of the attribute pairs, but these data are quite impossible to be fully collected in practice.

At attack period, CBF are quite faster than PacketScore due to the simplicity of score calculation. Table VI shows the process time in one time window (5 seconds) at attack period

TABLE VI. THE COMPARISON OF CBF AND PACKETSCORE IN PROCESS TIME AT ATTACK PERIOD

Attack Intensity	Process Time in 1 Time Window (second)	
	CBF	*PacketScore*
1×	0.332	0.495
5×	1.073	1.432
10×	1.919	2.564
20×	3.661	4.895

for CBF and PacketScore. Since CBF has no concept of time window at attack period, we measure the time that CBF processes the same amount of packets as those in a 5 second window of PacketScore instead. Due to the limitation of our experiment environment, we believe that the process time in the table for both methods can be reduced largely by optimizations and hardware supports.

Since the discarding period of PacketScore requires packet counting and CDF calculating, its process time in the table under all attack intensity conditions is higher than that of CBF, which only need a few looking-ups to generate score. For CBF, the most costly operation is to search the confidence values in nominal profile, so it can still be faster if a better hash function is adopted.

VII. DISCUSSION

CBF utilizes the attribute value pairs in TCP and IP headers to construct correlation patterns. In Section 6, these patterns are tested to be effective in distinguishing attack packets from legitimate ones under different types of denial-of-service attack. As shown in evaluation results, the most outstanding advantages of CBF are its high efficiency at attack period and small storage requirements for nominal profile. These features make CBF powerful especially in attacks with extremely large amount of traffic. In filtering ability, CBF does not have a strictly high accuracy compared to the previous researches. But the FP and FN rates at present are no more than 8 percent, which has already been acceptable in most cases.

Indeed, CBF can be ineffective if the attack packet flows mimic the correlation patterns of legitimate flows. However, in order to carry out large quantities of packets as fast as possible, even finding out the value distribution of single attributes will be too costly for the attacker. Thus the case that attackers have the complete attribute pair distributions will not be quite possible in practice. The situation that the single attribute value distribution is known by attackers is simulated in nominal attack in Section 6 and CBF takes on a

good performance by maintaining FP and FN rates around 5 to 6 percent.

In the situation that a distributed attack is carried out, all the source IP addresses will not be spoofed in the attack flow. But CBF can still successfully defeat this attack because the ability of CBF depends on the co-appearance of two attributes. That means even if the source IP address is authentic, the attack packets need to have the right attribute which frequently appears along with that source IP address as well. Considering the difficulty of that, CBF will also be quite effective when dealing with distributed attacks.

Flash crowds are the situations that a large number of legitimate customers happen to visit a server at the same time period. For CBF, it will not confuse flash crowds with denial-of-service attack. Since the filtering of CBF will not be affected by the number of packets, the packets sent by legitimate customers will have the frequent correlation patterns as usual. Thus these packets will also get a high CBF score to avoid being blocked.

VIII. Conclusion and Future Work

The key concept of CBF is correlation pattern, which is the co-appearance of attribute pairs in our implementation. We introduced confidence to represent the distribution of attribute value pairs and then devised a feasible approach to generating the nominal profile in order to store these confidence values. With the nominal profile, CBF can calculate scores for incoming packets at attack period to conduct filtering. Since the confidence reflects the frequency of appearances of the attribute value pairs, packets with more attribute value pairs of higher confidence will get higher score, which means more legitimate in this method. As shown in Section 3 and Section 6, CBF has a small storage size, an acceptable filtering accuracy, and a high scoring speed, which together make it a practical DDoS defending method in cloud platforms.

In the future, a more flexible discarding strategy to set the *discarding threshold* is required. The candidate one should not be so time-consuming that CBF loses its advantage of fast response at attack period. Also we will work on a more theoretical way of choosing the weights for each attribute pairs in CBF score calculation. The ideal strategy is adjusting the weights automatically based on the condition of the network. Finally, some optimizations and a better hash algorithm should be adopted to further accelerate the speed and the filtering accuracy of CBF.

Acknowlegements

The authors would like to express their thanks to the numerous valuable help given by Y. Sun, D.-C. Zhan, and many other friends and professors. This paper is partly supported by the National Science Foundation of China under Grant No. 61021062, 61073032 and 60736015, and Jiangsu Provincial NSF Project under Grants BK2008017.

References

[1] M. Armbrust et al., "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp.50-58, 2010.

[2] L. Zhang, and Q. Zhou, "CCOA: Cloud Computing Open Architecture," Proceedings of the IEEE International Conference on Web Services, pp.607-616, 2009.

[3] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems," ACM Computing Surveys, vol. 39, no. 1, p.3, 2007.

[4] A. Chonka, J. Singh, and W. Zhou, "Chaos Theory Based Detection against Network Mimicking DDoS Attacks," IEEE Comm. Letters, vol. 13, no. 9, pp.717-719, 2009.

[5] Y. Xiang, K. Li, and W. Zhou, "Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics," IEEE Trans. Information Forensics and Security, vol. 6, no. 2, pp.426-437, 2011

[6] H. Liu, and M.S. Kim, "Real-Time Detection of Stealthy DDoS Attacks Using Time-Series Decomposition," Communications (ICC), 2010 IEEE International Conference, 2010.

[7] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "PacketScore: A Statistics-Based Packet Filtering Scheme against Distributed Denial-of-Service Attacks," IEEE Trans. Dependable and Secure Computing, vol. 3, no. 2, pp.141-155, 2006.

[8] P.E. Ayres, H. Sun, H. J. Chao, and W. C. Lau, "ALPi: A DDoS Defense System for High-Speed Networks," IEEE J. Selected Areas Comm., vol. 24, no. 10, pp.1864-1876, 2006.

[9] H. Wang, C. Jin, and K.G. Shin, "Defense against Spoofed IP Traffic Using Hop-Count Filtering," IEEE/ACM Trans. Networking, vol. 15, no. 1, pp.40-53, 2007.

[10] P. Du, and A. Nakao, "DDoS Defense Deployment with Network Egress and Ingress Filtering," Communications (ICC), 2010 IEEE International Conference, 2010.

[11] Z. Duan, X. Yuan, and J. Chandrashekar, "Controlling IP Spoofing through Interdomain Packet Filters," IEEE Trans. Dependable and Secure Computing, vol. 5, no. 1, pp. 22-36, 2007.

[12] F. Soldo, A. Markopoulou, and K. Argyraki, "Optimal Filtering of Source Address Prefixes: Models and Algorithms," Proc. IEEE INFOCOM, 2009.

[13] M.T. Goodrich, "Probabilistic Packet Marking for Large-Scale IP Traceback," IEEE/ACM Trans. Networking, vol. 16, no. 1, pp.15-24, 2008.

[14] Y. Xiang, W. Zhou, and M. Guo, "Flexible Deterministic Packet Marking: An IP Traceback System to Find the Real Source of Attacks," IEEE Trans. Parallel and Distributed Systems, vol. 20, no. 4, pp.567-580, 2009.

[15] S. Yu, W. Zhou, R. Doss, and W. Jia, "Traceback of DDoS Attacks Using Entropy Variations," IEEE Trans. Parallel and Distributed Systems, vol. 22, no. 3, pp.412-425, 2011.

[16] Y. Xie, and S. Yu, "Monitoring the Application-Layer DDoS Attacks for Popular Websites," IEEE/ACM Trans. Networking, vol. 17, no. 1, pp.15-25, 2009.

[17] Y. Xie, and S. Yu, "A Large-Scale Hidden Semi-Markov Model for Anomaly Detection on User Browsing Behaviors," IEEE/ACM Trans. Networking, vol. 17, no. 1, pp.54-65, 2009.

[18] W. Dou, L. Qi, X. Zhang, J. Chen, "An Evaluation Method of Outsourcing Services for Developing an Elastic Cloud Platform", Journal of Supercomputing, (published online, DOI: 10.1007/s11227-010-0491-2), 2010.

[19] MAWI Traffic Archive, [Online]. Available: http://tracer.csl.sony.co.jp/mawi/

[20] S. Kasera et al., "Fast and Robust Signaling Overload Control," Proc. Int'l Conf. Network Protocols, 2001

[21] B. Babcock et al., "Models and Issues in DataStream Systems," ACM Symp. Principles of Database Systems, 2002.