JST: Smart Systems and Devices
Volume 31, Issue 1, May 2021, 001-008

Web Application DDoS Attack Defense Using Access Correlation

Nguyen Thi Thanh Tu[1*], Nguyen Thanh Tung[1], Tran Manh Thang[2]
[1]Hanoi University of Science and Technology, Hanoi, Vietnam
[2]Authority of Information Security, Ministry of Information and Communications, Hanoi, Vietnam
[*]Email: tu.nguyenthithanh@hust.edu.vn

Abstract

Web application distributed denial-of-service attack (Web-App DDoS Attack) is a common dangerous attack that hackers use to attack the information systems of organizations. Web application is often hackers' target because this kind of application is an external interface of an organization to provide the organization's activities services. In addition, due to the emergence of weaknesses and security holes in applications and operating systems, hackers can easily create a large-scale botnet for more effective Web-App DDoS Attack. In fact, there have been many research projects related to the defense against this type of attack. However, DDoS attacks still cause serious damage to the systems of organizations due to the fact that the attack methods are increasingly sophisticated and constantly changing. In this study, we propose a method for Web-App DDoS Attack mitigation on the basis of analyzing the relationship among the requests sent to the Web application to find out the source IP address of malicious requests and to perform mitigation. Our method provides a set of criteria that allows determining whether a source IP address is normal or malicious in a short period of time. The criteria also make it difficult for hackers to change the attack methods to overcome the characteristics of the criteria.

Keywords: DDoS Attacks, Flood Attack, Web-App DDoS Attack

1. Introduction

DDoS attack is a dangerous form of network attack that destroys the availability of a server or an information system by depleting resources or bandwidth of the system. DDoS attack is usually performed through a collection of hunched computers in a botnet and controlled by a C&C Server. Web application is a popular application to provide online services of many different types of industries and fields such as Economy, Politics, Finance, E-commerce, Transport ... Most Web applications providing services over the Internet using the HTTP/HTTPS protocol. Therefore, in addition to dealing with other types of network attacks (SQL Injection, XSS…), the Web application also has to face the DDoS attack. To perform a DDoS attack on a Web application, hackers take advantage of the botnet to send overflowing requests to the server, which would cause server overload and fall into a state of denial-of-service. Hackers often intentionally make real requests, like requests sent from a normal computer to bypass security devices such as Firewall and IDS/IPS.

In the previous study [1], we proposed a model to quickly detect the source of DDoS attacks on the Web application on the basis of combining two criteria: the frequency of access and the correlation between requests sent from a source IP. However, in the previous research, we only proposed ideas and solutions to implement them in a very basic and not specific way.

In this study, we propose a specific solution to solve the problem of detecting the source of the attack request on a Web application using the correlation criterion based on a basic idea: Requests sent from attackers will have the same characteristics and repeat because the attackers and the programs are controlled by the same C&C server. Whereas access machines normally have a high degree of randomness in the frequency of incoming access and requests. Here we call our recommended method AntiDDoS-AC for short.

Therefore, we seek to build correlation datasets among requests sent from the same machine in a normal operating state. Correlated data set will be used to detect the source of the attack request when a DDoS attack occurs on a Web server. We also propose different criteria to make it difficult for hackers to falsify the correlated data set during its construction.

2. Related Work

Authors Qin Liao et al. (2014) propose a set of features used to detect App-DDoS attacks on the basis of a Web Server access log analysis [5]. The authors use 9 features retrieved from the access log. The features are divided into groups of required time, frequency, and length. For experimental evaluation data, the authors use ClarkNet-HTTP datasets. This

ISSN: 2734-9373
https://doi.org/10.51316/jst.150.ssad.2021.31.1.1
Received: 15 July 2020; accepted: 3 September 2020

data set are collected over 2 weeks from a Web Server with 3,328,587 normal requests. For the attack data, the authors simulate the attacks and mixes them with the ClarkNet-HTTP dataset.

The authors compare the efficiency of the three methods namely Naive Bayes, RBF Network, and C4.5 with the data set obtained and with 14 features that the authors proposed. Authors Ko Ko Oo et al (2015) propose a method to prevent DDoS attacks on the application layer using the Hidden Semi-Markov model [6]. The authors use 03 groups of criteria on frequency of sending requests to Web applications, Web page reading time and order of received requests.

From these 3 groups of criteria, the authors choose 7 specific criteria (the total number of received packets, the total number of data sizes received, the average size of the packet, the frequency of packets received, data size per unit time, degree of variation in duration and packet size) to provide input to the authors method. The authors also proposed an algorithm model including 5 steps to distinguish attacks or normal packets.

The authors create test data to evaluate his method using Hping3 tool. Authors K. Munivara Prasad et al. (2018) propose a method to prevent DDoS attacks on Web applications using the machine learning method [2]. The authors use the K-Means algorithm to group sessions connected to Web Server into groups over time. In addition, the authors also use the criteria of the request length, the proportion of number of packets, and the time interval between requests sent to the Web Server to establish attack detection thresholds. The authors use LLDOS 2.0.2 test data set [3,4] to evaluate experimentally the proposed method.

3. Web App-DDoS Attack Prevention Model and Method

3.1. Operating Model and Principle Of Operation

In this section, we propose the operating model and principle of operation of the AntiDDoS-AC method. The operation of this system is described below.

The AntiDDoS-AC method has two main functions: DDoS Detection (DDoS Detection) and DDoS Prevention (DDoS Prevention) through finding sources of sending attack requests to update to BlackList. Separating the detection and prevention functions into two independent functions allows optimal performance and efficiency, because there are less system resources devoted to the DDoS Detection function than to the function DDoS Prevention. On that basis, the components of the AntiDDoS-AC method are built on two independent DDoS Detection and DDoS Prevention functions whose principle of operation is as following.

When the system operates in a normal state, the AntiDDoS-AC method will passively monitor the network connection to collect Web application access information and store it in the database. This database is data input for two functions DDoS Detection and DDoS Prevention. After a certain period of time, this database is retrieved to check whether the system is under a DDoS attack through the DDoS Detection function.

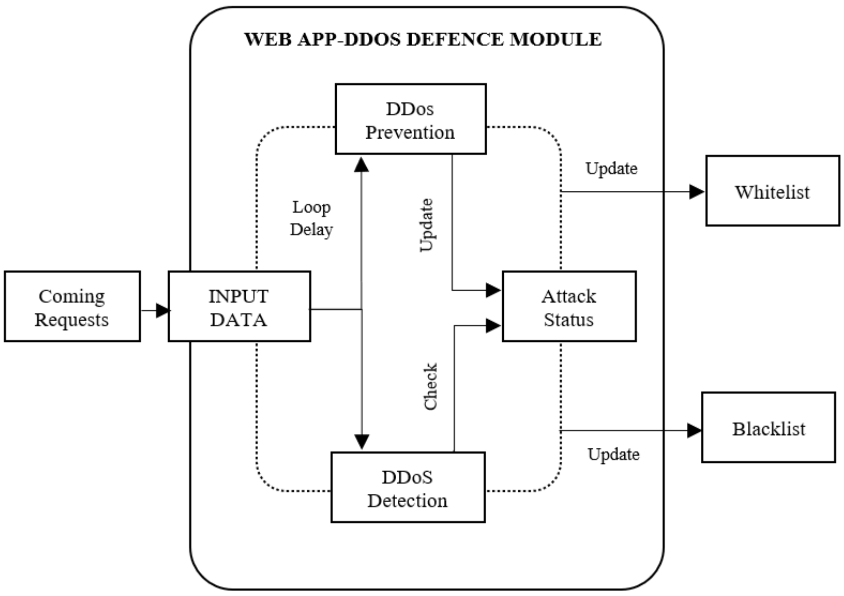

Fig. 1. Functions in AntiDDoS-AC component

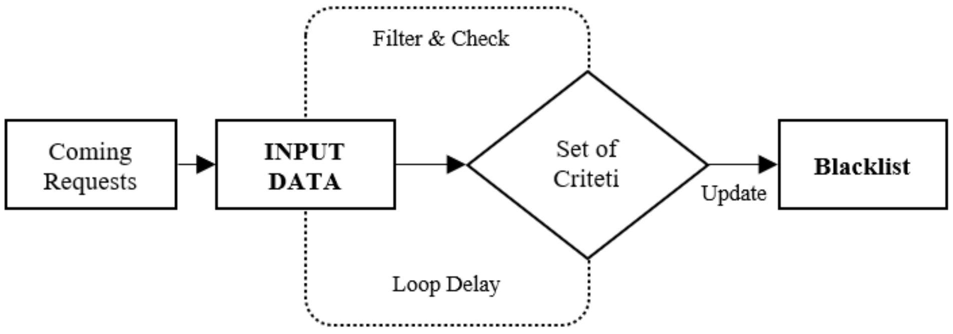

Fig. 2. Basic model of AntiDDoS-AC method

In case, DDoS Detection detects an attack, it will set the Attack Status flag to 1 to start DDoS Prevention function. DDoS Prevention function works in an infinite loop and constantly check Attack Status, if this value is set to 1, DDoS Prevention will perform prevention function through finding the source of the attack request and update to Blacklist. After finding out the list of source IPs sending the attack request, the AntiDDoS-AC method will send to the network device (Router) / security device (Firweall) to perform blocking. In case the DDoS Detection function does not detect the attack, the collected data set will be used to update the Whitelist. The Whitelist list is used to allow the source IPs in this list to have priority access to a Web application when an attack occurs [8, 9].

3.2. AntiDDoS-AC Method

The basic idea of the AntiDDoS-AC approach is that when the attack occurs, the requests sent to the server will include both the attack requests and the normal requests. In which, the requirements for the attack make up the majority. Therefore, the problem that the AntiDDoS-AC method has to deal with is finding out the attack sender to put in the Blacklist and the normal source to put in the Whitelist. The basic idea of the AntiDDoS-AC method is described as shown below.

In this study, the AntiDDoS-AC method uses two correlation criteria. This criterion is built on the basic idea that when observing the set of requests received from the average user's computer, they will have a certain correlation and randomness. While the requests received from the attacker will have different characteristics than usual (for example, these machines send only one request repeatedly or a group of requests according to the control command of the C&C server). Therefore, when the system operates in a normal state, we seek to construct a correlated data set to identify the source of the attack request and normal request when the attack occurs.

3.3. Building Correlation Criteria in AntiDDoS-AC Method

The correlation criterion (we call it SAT2 for short, as inconsistent with our previous research [1]) which is built on the basic idea that for normal users, when accessing a Web application, they will access various contents on the server (surf the web). Therefore, when looking at the server side, we can see the correlation of requests sent to the server from each source IP. For example, when a user sends an *rA* request to the server, he or she then sends an *rB* request to the server as well. The question is how to build correlated datasets to ensure that hackers can hardly send false requests on purpose to the server during normal operation and use the request set during the attack.

To solve this problem, during normal operation, we provide a set of criteria that apply to each source IP sending the request and the set of requests sent from that source IP. Only when the source IP and the requests simultaneously satisfy the conditions, can they be included in building the correlated request set using the Association Rule algorithm. From there, the proposed method includes 02 processes of building correlation sets and detecting attack requirements, which are described in detail as below:

a) Build correlation data set

To determine whether a request is anomalous or not, we cannot rely on each request individually, because requests used in DDoS attack are generated as normal requests. Therefore we need to find the correlation between them to find the anomaly. Through experimental observation, we found that there is a difference between requests sent from a normal computer and requests sent from an attacker. Specifically: for a normal computer, when accessing a Web page, the user will send different requests to the Web server to access different information on a Web page (illustrated as shown below). Whereas the requests sent from the attacker are usually the same requests or groups (because they are controlled by the same C&C server) or having the same repetitive pattern of change.

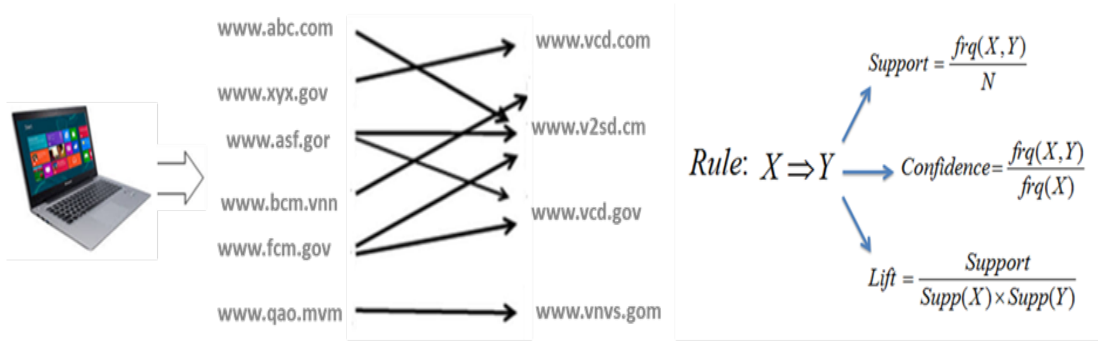

Fig. 3. Illustration of correlation request submission

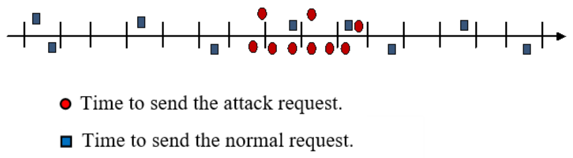
● Time to send the attack request.
■ Time to send the normal request.

Fig. 4. Illustration send request

As per our above analysis, the problem is how to build clean correlation data sets that hackers can hardly attack into the training phase with false requests. In order to solve the problem, we build a set of criteria applied to the source of requests and corresponding requests to verify whether the requests are sent from ordinary users. After specific period of time, we pull the information out for verification. The qualified requests and senders will be input to the Association Rule algorithm to create correla data set. The criteria we use are as following.

b) Random criteria regarding time of access

This criterion is to figure out whether URL sent from different source IP are eligible to build correlated datasets. We observed that, if the system operates smoothly, any requests from different IP sources can be sent randomly in time to the server. However, if the system is attacked, departure time of requests will no longer be random because they are all forced to send simultaneously as shown below.

According to the above observation, for this criterion, we use Entropy to calculate the randomness in time to send the requests. We use unit *TT* and this unit is divided into smaller units Δt (the value of Δt is set according to each specific protection-needed system by the administrator). And the value of Entropy *H (X)* is calculated based on the arrival probability of each request being sent within a certain time and divided as mentioned above. Let *n* be the required number of submissions per sampling, so the Entropy of the *H (X)* value is calculated as follows:

$$H(X) = -\sum_{i=1}^{n} p_i log_2 p_i$$

We need to determine the Entropy value for the point of arrival of normal mode requests. We have sampled the last *k* times. Let H_T be the Entropy value calculated after k sampling times. Therefore, we can calculate the value.

$$H_T = \sum_{n=1}^{k} H(X_n)$$

From the analysis above, we realize that the ability of randomness happening to the arrival of the requests is higher than that towards the time of the DoS / DDoS attack. Therefore, we choose threshold value $Thd = H_T/\alpha$. Then, Entropy calculated with sampled H_{data} data will be calculated and compared with Thd value. If $H_{data} < Thd$, then DoS / DDoS attack has occurred. α is the value applied by the administrator in order to be able to change accordingly protection-needed server.

c) The validity of each source IP sending the request

In the next step, we need to determine whether the source IPs which have satisfied the above criterion meet the set of criteria below or not. In the case that the source IPs satisfy the following set of criteria, these IPs will be put on the Whitelist list and requests sent from this source will be constructed with the correlative data set. The criteria include:

- Requests that are not sent again in *Δs* time. We find that the normal user when accessing a website takes time to read the content of that page and do not resubmit the request within a certain period of time. However, with attacking computers, the same request can be sent continuously during a small amount of time. Therefore, we believe that the normal source should satisfy the condition of not resending sent requests within a certain period of *Δs* time. The Δs value may vary from system to system based on monitoring the system's performance without being hacked. In this study, we choose *Δs* = 5s, based on the collected attack log data to evaluate the proposed solution.

- Average arrival time deviation of requests will be sent from a D_{TR} source IP: The moment when requests are sent from the same source IP from a normal machine will have some degree of randomness, while requests are sent from the attacker has a certain degree of correlation (due to being sent through a same malicious program). To identify clean sending sources, we determine the total time deviation calculated by the time of arrival of each request. With normal sending sources, this value will have a certain magnitude, while with attackers this value will below. We use an M_R array of n elements to store information about the duration of requests received from a source IP. For example with normal source M_R = [12,3,5,10,…,n]), with attack source M_R = [2,2,3,2,…,n]). Let i be the ith request received and T_i the time the request is received. From there the total time deviation between the requirements is calculated by the following formula:

$$D_{TR} = \frac{\sum_{i=1}^{n} T_{i+1} - T_i}{n}$$

A T_{TR} threshold is set to determine whether the arrival times of requests received from a source IP satisfy a normal condition or not. If the total deviation $D_{TR} > T_{TR}$, the arrival time of requests received from an IP is then deemed to satisfy the condition. In this study, we choose the value T_{TR} = 10s, on the attack log data we use the experimental evaluation model of the proposed solution. This value may vary depending on each protected system.

- D_L url length average deviation criterion: Normal machines when sending a request to the server usually have different lengths (calculated by the number of characters in the url) due to the different content visited on the server. While the lengths of the requests sent from attackers is usually the same and repeatable. Therefore, the D_L deviation of the attack machines will be much smaller than the normal ones.

- Similar to the above approach, we use the M_L array to store the length of n requests sent to the server from a source IP. For example with normal source M_L = [12,30,50,120,…,n]), with attack source M_R = [30,30,55,55,…,n]). Let i be the ith request received and C_i be the length of each request received. Average deviation D_L is determined by the following formula:

$$D_L = \frac{\left[C_i - D(c)\right]^2}{n} \qquad D(c) = \frac{\sum_{i=1}^{n} C_i}{n}$$

From there, for each source IP with inbound request set, we determined that this D_L value must be greater than a pre-set T_{DL} threshold based on each specific system that needs to be protected. In this study, we choose T_{DL} = 60 characters on the attack log data, to evaluate the proposed solution experimentally. After finding the source IPs and requests sent from those IPs satisfing the same conditions above, the source IP will be put into the Whitelist and the data set including the IP and requests will be used as input of the Association Rule algorithm to construct the correlation data set.

To determine the input parameters for the Association Rule algorithm (level of support and reliability), we collect logs of access data on the actual network environment at Hanoi University of Science and Technology when the system is not under attack. Then, we apply the Association Rule algorithm to the data set with different levels of support and reliability to find out a suitable value for the largest correlation set. Through experiments, we determined the level of support equal 2 and reliability greater than 50% are the appropriate values for the largest number of correlation sets. However, for each system that requires different level of protection, those values may be changed accordingly.

3.4. Design the Data Structure for the Algorithm

The second issue that comes along with this criterion is how to quickly to detect whether requests are offensive or normal ones, in the case that the server receives a lot of requests when an attack occurs. To solve this, we design 02 structured data tables (T_R và T_A) with the algorithm that allows storage and quick searching for a set of requests satisfying correlated conditions when the server receives many requests at the same time.

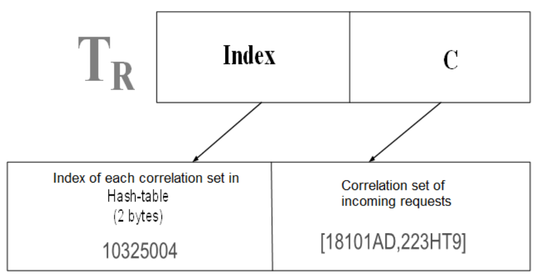
T_R
Index
C
Index of each correlation set in Hash-table (2 bytes)
10325004
Correlation set of incoming requests
[18101AD,223HT9]

Fig. 5. Structure of T_R data sheet

T_R datasheet has the structure as shown above, used to store information about the hash value of each url and corresponding index in the hash-table. We use T_R table separately to store index and hash information (url) due to large number of urls which need to store information. Storing that information separately will optimize storage space and processing time. We will present in details as shown in the algorithm below.

We use the T_A data sheet, structured as shown in figure 6, to store information to be processed when the first request from a source IP arrives. The information about the correlated request set in the T_R table is then transferred to the T_A table to verify whether that subsequent requests from each incoming source IP are normal or attack. Information in the table only has a lifetime of Δtc based on the time the test takes place and the time when the server received the first request.

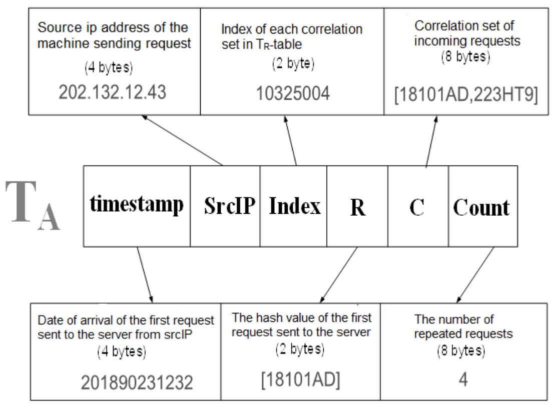
Source ip address of the machine sending request
(4 bytes)
202.132.12.43
Index of each correlation set in TR-table
(2 byte)
10325004
Correlation set of incoming requests
(8 bytes)
[18101AD,223HT9]
TA
timestamp
SrcIP
Index
R
C
Count
Date of arrival of the first request sent to the server from srcIP
(4 bytes)
201890231232
The hash value of the first request sent to the server
(2 bytes)
[18101AD]
The number of repeated requests
(8 bytes)
4

Fig. 6. Structure of T_A data sheet

In this study, we choose the value *Δtc* = 300s. This means we assume that in about 5 minutes the user sends another request to the server and we can verify whether that the source IP is normal or not. 300 seconds is the amount of time required to verify if a sender is normal or not. However, if the IP is attacking, it will be verified as soon as the next request is sent to the server without wasting *Δtc* (300s).

3.5. Detect the Source of the Attack Using the Correlation Data Set

a) Some concepts used in the method

FT threshold to detect suspect attack set: FT attack detection threshold is pre-set threshold to define the same requests that the server receives in a unit of time.

Set of suspected RA attack requests: A request that satisfies a condition with the frequency of sending a FT request from the same source IP address.

The BL list is used to store the source IP addresses information identified as the attack sources after requests sent from these source IP addresses have been checked against the set of criteria. For each different attack request *R(i),* we have a list of different source IP addresses *S(s,i)* and corresponding to the frequency of sending requests *f(s,i).*

The list of sources sending BlackList attack requests is denoted $BL = \{B_1, B_2, ..., B_n\}$, $Bi = [S(s,i),R(i),f(s,i)]$.

D_{Inp} Input Data Set: Structured data set used to store the request information sent to the server. For each new request sent to the system, D_{Inp} will be updated.

Data set to store suspect attack requests D_{Sus}: This data set has the same structure as the D_{Inp}, used to store all information related to the request which are considered suspected attack after satisfying the required sending frequency greater F_T, after each interval check for the condition of D_{Inp}.

b) Set the input parameters for the algorithm:

Value Δs = 0.5s, test time with new request for each SrcIP; The value Count = 3, the value specifies the number of repeated requests in a period of Δs from a srcIP; The above values are pre-set and may vary by system.

```
Algorithm: FDDA-S_AT2
For each $R_i$ in $D_{Sus}$ do
        copy $D_{Sus}[srcIP,\ url]$ to $T_A$
        copy $TR[Index,\ C(srcIP)]$ to $T_A$
    end for
    while true do
        $T_C = Get(TimeNow)$
        if $(T_C - timestamp(srcIP) \geq \Delta s)$
            get $R_A$ in $D_{Inp}$ where $(timestamp(srcIP) \geq T_C - \Delta s)$
        end if
        for each $R_k$ in $R_A$ do
            if $hash(R_k)$ in $C(srcIP)$
                add $srcIP$ to $Whitelist$
            else
                if $(hash(R_k) = C(srcIP))$
                    update $Count(srcIP)$
                    if $(Count(srcIP) \geq 3)$
                        add $srcIP$ to $Blacklist$
                    end if
                end if
            end if
        end for
        if $(T_C - timestamp(srcIP) \geq 300)$
            remove $T_A(srcIP)$
        end if
        delay $\Delta s$
    end while
End
```

Fig. 7. AntiDDoS-AC algorithm

The AntiDDoS-AC-STA2 algorithm is performed with the following steps:

Step 1: For each request in the DSus corresponding to the corresponding source IP, define the index and the corresponding set of that source C(srcIP) in the T_R, then input into the T_A as well as update the corresponding timestamp.

Step 2: Periodically after each Δs period, check in the D_{Inp} table if there are new requirements for each srcIP in T_A. If yes, check the following steps.

Step 3: Calculate the hash value of the request. If the hash value of the incoming request exists in the C(srcIP) correlation set then put that IP in Whitelist.

If the hash value matches *R* value then update count value for that srcIP and check if count value (srcIP) ≥ 3 or not. If so, put that IP in Blacklist.

The IPs put in Whitelist and Blacklist will be deleted in D_{Inp} and D_{Sus} correspondingly to release memory.

6. Experimental and Result

6.1. Creat Experimental Data

For experimental evaluation, we have built a botnet on a virtual environment on 03 highly configurable physical servers. Experimental system with a scale of 13 Cisco routers and 45 client computers, installed with Win XP and Centos operating systems. These computers are infected with malicious code and under the control of a C&C server to perform TCP Syn Flood attack on Web Server. Routers are set up in a virtual environment using GNS3 software [7] with the Cisco 7200 Router series.

After performing a simulated attack for a period of 3 minutes, we obtained a data set stored as PCAP which contained 66,851 packets. The resulting database got 1,310 requests, including both attack and normal requests, and was stored in MySQL database.

We perform log sampling accessing website of Graduate Training Institute (Hanoi University of Science and Technology) from October 17, 2018, 18:21:22 to October 18, 2018, 09:20:35. As a result, we obtained 6,686 requests from 372 different srcIPs.

6.2. Experimental Evaluation

We perform an experimental testing evaluation with a dataset using the following parameters used as follows:

- *Threshold of detection of suspicious FT attack requests (requests / second).*
- *The quantity required in the DSus.*
- *The quantity of suspicious requests to attack RA.*
- *The correct detection rate of the sources sending attacks: Detection Rate - DR.*
- *The false detection rate (define a normal request as an attack one): False Positive - FP.*

Experimental testing results prove a high detection rate of our method as shown below:

Table 1. Experimental testing results of the AntiDDoS-AC method

F_T	D_{Sus}	R_A	DR	FP
50	250	185	75.32%	1.28%
100	500	365	92.56%	1.11%
150	750	668	94.08%	0.89%
200	1000	897	88.75%	1.47%
300	1310	863	67.89%	1.49%

From the above experimental testing results, it shows that the rate of detecting the source of the attack request depends on the FT value. This value should be adjusted to suit each server that needs to be protected. If the FT value is set too small, the false detection rate will increase. If the FT value is set too large, attack senders with low request frequency will be omitted and the rate of detecting DR attack sources will decrease as well.

6.3. Comparison of the effectiveness of the AntiDDoS-AC Method with Other Methods

Table 2. Comparison of experimental testing results among AntiDDoS-AC, KNN, and NB

Methods	Detection Rate	False Positive
KNN [10]	89.03%	1.03%
NB [9]	92.47%	1.47%
AntiDDoS-AC	93.75%	0.89%

Results of comparison regarding processing time is shown in figure 8:

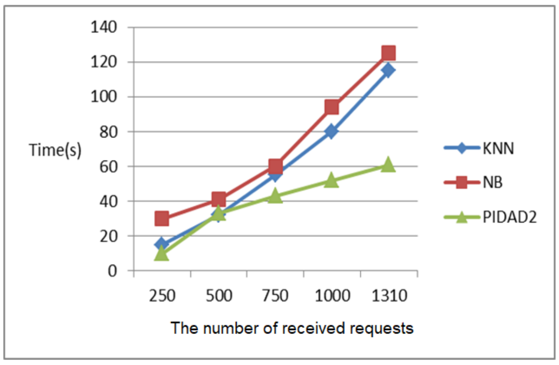

Fig. 8. Test data processing time of AntiDDoS-AC, KNN, and NB

The results showed that, on the same test data set, the AntiDDoS-AC method has a higher rate of detecting attack requests (Detection Rate - 93.75%) and the false detection rate (False Positive) is 0.89%, compared to the two compared methods.

7. Conclusion

In this study, we propose a specific solution to solve the problem of detecting the source of the attack request on the Web application using the correlation criterion with the proposed method, AntiDDoS-AC. Experimental testing results show that our proposed method has a higher rate of detection of attack sources with less amount of time needed. However, the experimental results are only evaluated on the data set generated from a self-built test environment.

In our next research, we will continue to study new criteria to make our methods more accurate and effective. In addition, we will keep on studying experimental evaluation on many different data sets which will surely enhance the degree of objectivity in evaluating the proposed method.

Acknowledgements

We would like to express our sincere thanks to Assoc. Prof. Nguyen Khanh Van and Assoc. Prof.

Nguyen Linh Giang for their valuable suggestion. Our gratitude extends to colleagues of the Information Network Center for coordinately implementing this experimental research. Besides, we are thankful for the support from the Department of Information Security - Ministry of Information and Communications through the National Science and Technology Mission: *"Research and develop a Reference Framework for Information Security for e-government"*, code: KC.01.07 / 16-20.

References

[1]. T.M. Thang, Van K. Nguyen (2017), FDDA: A Framework for Fast Detecting Source Attack in Web Application DDoS Attack, SoICT 17: Eighth International Symposium on Information and Communication Technology, December 7–8, 2017, Nha Trang City, Viet Nam. ACM, New York, NY, USA, https://doi.org/10.1145/3155133.3155173.

[2]. K. Munivara Prasad, A. Rama Mohan Reddy, K. Venu Gopal Rao, An Experiential Metrics-Based Machine Learning Approach for Anomaly Based Real Time Prevention (ARTP) of App-DDoS Attacks on Web, Artificial Intelligence and Evolutionary Computations in Engineering Systems pp 99-112, March 2018.

[3]. M.I. MIT, in Darpa Intrusion Detection Evaluation. Retrieved from Lincoln Laboratory: https://www.ll.mit.edu/ideval/data/1998data.html.

[4]. D.M. Powers, in Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation, 23rd international conference on machine learning (Pitsburg, 2006).

[5]. Qin Liao, Hong Li, Songlin Kang, Chuchu Liu, Feature extraction and construction of application layer DDoS attack based on user behavior, Proceedings of the 33rd Chinese Control Conference, July 2014.

[6]. Ko Ko Oo, Kyaw Zaw Ye, Hein Tun, Kyaw Zin Lin and E.M. Portnov, "Enhancement of Preventing Application Layer Based on DDOS Attacks by Using Hidden Semi-Markov Model", Genetic and Evolutionary Computing pp 125-135, August 2015.

[7]. https://www.gns3.com/

[8]. K. J. Higgins, Researchers to Demonstrate New Attack That Exploits HTTP, Nov. 01, 2010, [online] http://www.darkreading.com/vulnerabilitymanagement/167901026security/attacksbreaches/228000532/index.html.

[9]. RioRey, Inc. 2009-2012, RioRey Taxonomy of DDoS Attacks, RioRey Taxonomy Rev 2.3 2012, 2012. [online]. http://www.riorey.com/x-resources/2012/RioRey Taxonomy DDoS Attacks 2012.pdf.