A HYBRID APPROACH COMBINING RULE-BASED AND ANOMALY-BASED DETECTION AGAINST DDOS ATTACKS

Chin-Ling Chen[1] and Hsin-Chiao Chen[2]

[1]Department of Information Management, National Pingtung University, Pingtung, Taiwan, 900
[2]Department of Information Management, National Pingtung Institute of Commerce, Pingtung, Taiwan, 900

ABSTRACT

We have designed a hybrid approach combining rule-based and anomaly-based detection against DDoS attacks. In the approach, the rule-based detection has established a set of rules and the anomaly-based detection use one-way ANOVA test to detect possible attacks. We adopt TFN2K (Tribe Flood, the Net 2K) as an attack traffic generator and monitor the system resource of the victim like throughput, memory utilization, CPU utilization consumed by attack traffic. Target users of the proposed scheme are data center administrators. The types of attack traffic have been analysed and by that we develop a defense scheme. The experiment has demonstrated that the proposed scheme can effectively detect the attack traffic.

KEYWORDS

Distributed denial of service, firewall, detection

1. INTRODUCTION

Distributed Denial of Service (DDoS) has caused a serious threat to network security since it has significantly damaged network infrastructure as well as Internet services. DDoS attacks can be categorized into two types: semantic and flooding attacks. Semantic attacks usually exploit some weakness of the target system and implant bot onto it. On the other hand, flooding attack creates a large number of attack network traffic, service requests and connections, and thus consuming a large number of victim resources, such as CPU, bandwidth and internal memory.

Recently, much effort has concentrated on detection methods for flooding attacks. We may categorize them into four main approaches: traceback-based [1-5], rule-based [6-16], protocol-based [17-27] and anomaly-based [28-30]. Traceback-based methods make the victim to identify the attack source as well as attack paths once the attack has been encountered. Among the available traceback methods, Deterministic Packet Marking (DPM) [4] is considered to be a simple and relatively effective traceback scheme. The victim employs a DPM algorithm to identify data marks of suspicious packets and choose the filtering probability of the marked packets, which is based on both arrival rate and attack paths. The data mark rate is adjusted dynamically based on attack frequency. Rule-based detection usually defines some rules (also called signatures) to set normal traffic apart from suspicious traffic. Whenever the methods locates incoming traffic that matches content or condition found in a rule, alert will occur. Rule-based detection is effective in the past when only a few malware strains can be found. However,

DOI: 10.5121/ijnsa.2016.8401

as the number of malware increase dramatically, it is impossible to create and maintain this number of rules.

Scalable detection [6] has proposed a new data structure called partial completion filters (PCFs) that can detect scanning attacks and partial completion attacks in the network. Not only small traffic volumes but also flooding traffics was used on the experiments to demonstrate how PCFs works to achieve high effectiveness. By using group-testing theory, Live Baiting [7] has the advantage of exploiting attackers within incoming traffic with the minimum number of test and low state overhead. The state overhead needed is in the order of number of attackers, rather than number of clients. Neither legitimate requests nor anomalous behaviour are required in model, Live Baiting scales to large services with millions of clients.

Protocol-based flooding attacks can be classified into two categories based on the protocol level that is targeted [17]. They are network/transport-level [18-23] and application-level flooding [24-27]. Network/transport-level flooding has been launched attacks to consume the victims' resource by exploiting the bugs and the weakness of IP, TCP, UDP and ICMP protocols. Similarly, Application-level flooding sends faked application-level protocol requests to the large number of innocent servers (reflectors), which flush packets to the victims.

Anomaly-based detection [28-30] usually discusses the methods for generating statistical data that can be used to perform detection and analysis. Rather than simply alerting whenever some exceptional traffic pattern is observed, an anomaly-based detection is capable of discerning between attack traffic and normal traffic. This type of detection is more powerful, but more difficult to implement.

The proposed scheme combines the best properties of rule-based and anomaly-based detection. For the part of rule-based detection, we set up three criteria for incoming traffic. They are throughput, CPU utilization and memory utilization. We further identify suspicious traffic by examining one criteria or combination of criteria along with ANOVA test for the part of anomaly-based detection. In this study, we mainly focus on network/transport-level flooding attacks to simplify the experiment.

The rest of paper can be organized as follows. Section II describes the proposed scheme and discusses the detection and response mechanism. In Section III, simulation and results are presented. Finally, we draw our conclusions in Section IV.

2. THE PROPOSED SYSTEM

The proposed system can be classified into two parts: detection and response, which can be described as follows.

2.1. Detection

The design principle of detection aims to identify the traffic with suspicious behaviour. Because no single flow may be suspicious, we perform flow aggregation with arrival rate to identify overloading behaviour. Assume F_t is the arrival rate measured at the receiver at time *t*. We have

$$\Delta F_t = \frac{F_t - F_{t-1}}{F_{t-1}} \qquad (1)$$

The receiver first checks whether ΔF_t reaches incremental rate threshold (Th_f). If this does occur, incremental rate counter, C_f, is incremented by 1. Otherwise, C_f is decreased by 1 until it reaches 0. We also set a counter threshold, α_f, to ensure proper provisioning of QoS. When C_f grows up to

be α_f, we may say these aggregate flows enter warning state. The Boolean variable B_f is set to be 1. If C_f reduces to be below α_f, B_f is set to be 0. The flowchart of flow identification is depicted at Figure 1. Legitimate traffic can be considered as attack traffic whenever they show certain overloading behaviour. Therefore, we need another indicator(s) to assist to identify suspicious traffic. Active query, like DNS amplification, usually introduces significant overhead to the interface loading of the victim. Query frequency is in proportion to current network utilization [13]. The network utilization information includes memory utilization and interface loading (CPU utilization), which will be examined in the following.

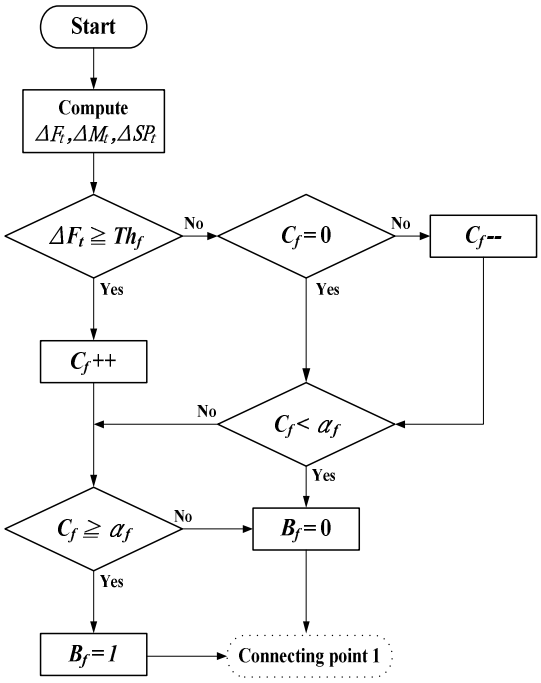

Figure 1. Flowchart of flow identification

Let M_t be memory utilization observed at the receiver at time t. We have

$$\Delta M_t = \frac{M_t - M_{t-1}}{M_{t-1}} \qquad (2)$$

We further check whether ΔM_t reaches incremental memory utilization threshold (Th_m), a value set by experienced operator to avoid memory exhaustion by extremely high traffic volume. If so, the counter of incremental memory utilization, C_m, is incremented by 1. Otherwise, C_m is decreased by 1 until it reaches 0. We use a counter, α_m, to examine the continuity of potential attack. When C_m increases up to α_m, memory utilization enters warning state. The Boolean variable, B_m, is set to be 1. When C_m decreases down below α_m, B_m is set to be 0. Figure 2 represents the flowchart of memory utilization identification.

Again, we set P_t to be the CPU utilization measured at the receiver at time t. We have

$$\Delta P_t = \frac{P_t - P_{t-1}}{P_{t-1}} \qquad (3)$$

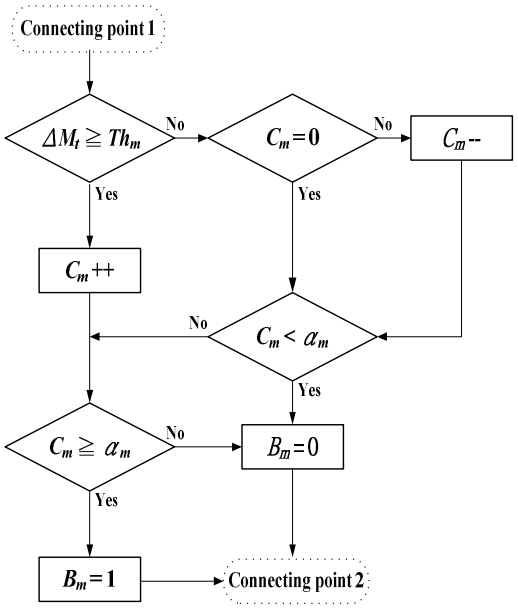

Figure 2. Flowchart of memory utilization identification

Let Th_p and C_p be the incremental threshold and the incremental counter of CPU utilization, respectively. Whenever ΔP_t exceeds Th_p, C_p is incremented by 1. Otherwise, C_p, is decreased by 1 until it reaches 0. When C_p reaches its upper bound threshold,α_p, the CPU utilization enters warning state. The Boolean variable B_p is set to be 1. Otherwise, B_p is initialized to be 0. Figure 3 is the flowchart of interface loading identification. Table 1 represents the status of combination of three parameters, B_f, B_m and B_p. We have three status results: *OK*, *minor warning* and *attack alert*. A flow is considered to be *OK* if passes through all the three parameters' examination. A flow is plausible if only pass one or two parameters' tests. We may consider it as *minor warning* and need one-way ANOVA test (will be described at section 3.3) to further verify. On the other hand, a flow is said to be *attack alert* if fails all the three parameters' tests or two parameters' test (at least including parameter B_p). The reason attack alert always includes parameter B_p is that an increase on CPU utilization of the victim is an indication of DDoS attacks [13].

Minor warning traffic should be further identified by comparing the mean and variance of the throughput of normal traffic. We use a one-way ANOVA (analysis of variance) to test the difference of k group means. In this case, k=2. The attack traffic is generated randomly by TFN2K (Tribe Flood, the Net 2K). We measure using a test statistic that has an F-distribution with (k–1, n–k) degrees of freedom. The null hypothesis will be the throughput means of two population, normal traffic and attack traffic, are equal, and alternative will be that the throughput means of two population differ from each other. We have H_0: $\mu_1 = \mu_2$, where μ_1, μ_2, are the mean throughput of normal traffic and attack traffic, respectively.

$$H_1: \mu_1 \neq \mu_2.$$

The numerator (*MSR*) is the variability between group means. The denominator (*MSE*) measures how much individual observations vary in each group from their group mean estimates. *MSR* is the mean squared treatment and *MSE* is the mean squared error. *MSR* and *MSE* stands for Regression Sum of Squares (*SSR*) and Error Sum of Squares (*SSE*) divided by its degrees of freedom k–1 and n–k, respectively. If the ratio of MSR to MSE is significantly high, we can

conclude that the group means are significantly different from each other. The *F*-statistic is given below.

$$F = \frac{\frac{SSR}{k-1}}{\frac{SSE}{n-k}} = \frac{\sum_j n_j(\bar{x}_j-\bar{x})^2/(k-1)}{\sum_i\sum_j(x_{ij}-\bar{x}_j)^2/(n-k)} = \frac{MSR}{MSE} \sim F_{k-1,n-k},$$

where n is total number of observations, $\bar{x}$ is the overall mean of all observations, and $\bar{x}_j$ and n_j are the mean and the number of observations for the jth group, respectively. We assume the level of significance (α) is 0.05. If the *P*-value computed from the samples is less than the level of significance, α, we have evidence against the null hypothesis. That is, we reject the null hypothesis and say that the result is statistically significant.

2.2. Response

The overall procedure of our system architecture is illustrated in Figure 4. The machine information of symbol A-I are listed in Table 2. The scheme can be divided into three parts: Attacker, Monitoring Server and Victim Host. There are two main modules within the Attacker: a command module (tfn) and a Zombie module (td). The command module is the piece that controls the Zombie. The command module tells the Zombies when to attack and with what exploit. The Zombie runs on a machine in listening mode and waits to get commands from the command module. The Attacker generates randomized UDP flood, TCP/SYN flood, ICMP/PING flood, ICMP/SMURF flood, MIX flood (UDP/TCP/ICMP interchanged), TARGA3 flood (IP stack penetration).

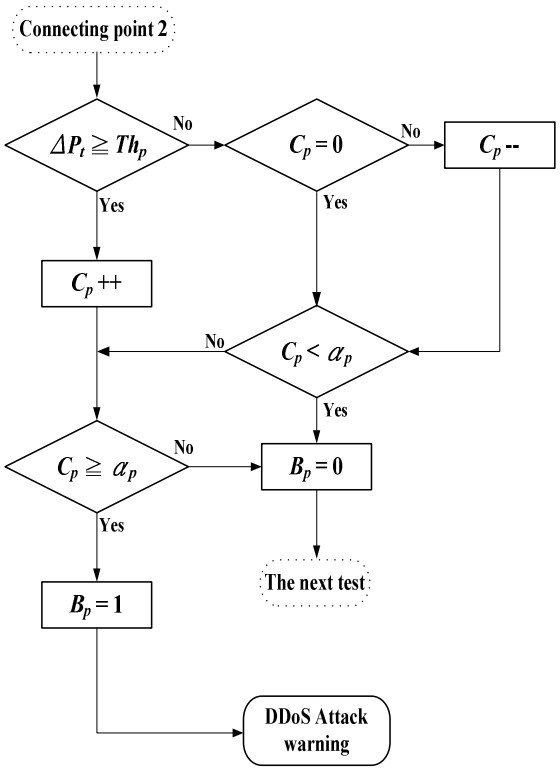

Figure 3. Flowchart of CPU utilization identification

Table 1. Status of combination of three parameters

	B_f	B_m	B_{sp}	Status
C_1	0	0	0	*OK*
C_2	0	0	1	*minor warning*
C_3	0	1	0	*minor warning*
C_4	0	1	1	*attack alert*
C_5	1	1	1	*attack alert*
C_6	1	0	0	*minor warning*
C_7	1	0	1	*attack alert*
C_8	1	1	0	*minor warning*

There are three modules in the Monitoring Server. They are Control module, Mirror module and Statistics module. The Secure Shell (SSH) secures the remote connection to a remote machine. The Control module writes a shell script to SSH command(s) to firewall (Figure 4(a)). The Control module modifies the rule of IPTABLE based on the information provided by both Mirror module (Figure 4(b)) and Statistics module (Figure 4(c)). Port mirroring is used on a switch to send a copy of packets to the mirror module. Mirror module monitors the network traffic and helps the administrators to diagnosis the network performance. Statistics module collects data from victim host and mirror module. We use **AWK** to search for particular strings and modify the data as required. Statistics module then does plotting in **gnuplot** script from **perl**. We use **free** command and **iostat** command to get the information on available RAM and interface loading in the victim host. The victim host output the related information to monitoring server for further diagnosis.

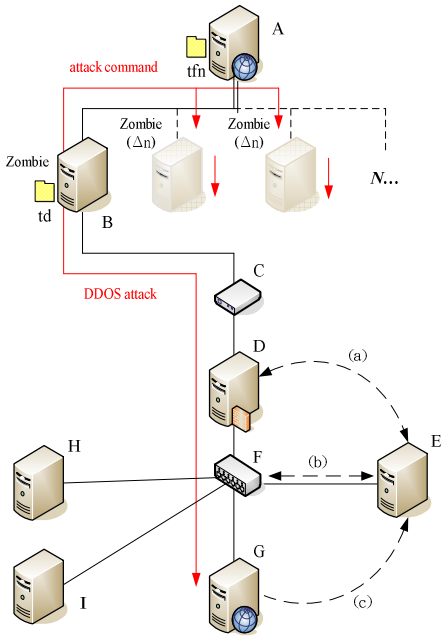

Figure 4. System architecture

Table 2. Machine information

Symbol	Specification	OS	Role
A	Intel® Core™2 Duo E7500 RAM: 8 GB 1066 MHz DDR3 SDRAM	Linux Fedora 20	Attacker
B	Intel® Core™2 Duo E7500 RAM: 8 GB 1066 MHz DDR3 SDRAM	Linux Fedora 20	Zombie
C	DES-3200-26		Switch
D	Intel® Core™2 Duo E7500 RAM: 8 GB 1066 MHz DDR3 SDRAM	Linux Fedora 20	Firewall
E	Intel® Core™2 Duo E7500 RAM: 8 GB 1066 MHz DDR3 SDRAM	Linux Fedora 20	Monitoring Server
F	DES-1024D		Switch
G	Intel® Core™2 Duo E7500 RAM: 8 GB 1066 MHz DDR3 SDRAM	Linux Fedora 20	Victim Host
H	Intel® Core™2 Duo E7500 RAM: 8 GB 1066 MHz DDR3 SDRAM	Linux Fedora 20	Server
I	Intel® Core™2 Duo E7500 RAM: 8 GB 1066 MHz DDR3 SDRAM	Linux Fedora 20	Server

3. EXPERIMENTS AND EVALUATION

In this section, we analyse which combination is appropriate and discuss to take some action to mitigate DDoS attack under each combination type. We implement the attack tool- TFN2K (Tribe Flood, the Net 2K) on the attacker and evaluate the performance of the proposed system through experiments based on the following criteria: 1) Comparison of resource utilization for normal traffic and attack traffic; 2) The minimum cost to detect the attack traffic; 3) Correction of detection result (false positive and false negative). We concentrate on four major DDoS attacks: UDP flood, TCP SYN flood, ICMP flood and MIX flood.

3.1. Comparison of resource utilization for normal traffic and attack traffic

We first observe the real normal traffic traces captured at the campus network. Not only low scale, but also large scale normal traffic are observed in the experiment. Figure 5 and Figure 6 have shown that low scale normal traffics featuring TCP and UDP mix and large scale TCP traffic within 300 seconds, respectively. The throughput of TCP normal traffic drops periodically due to transmission completion. Continuous sending a low rate traffic is another strategy for the attackers to bring down a server. We found that the original TFN2K only generates constant-like rate attack traffic. The reason is that the implementations of rand ( ) in original TFN2K have serious shortcomings in the randomness, distribution and period of the sequence produced. At the start, we use the original TFN2K as the attacker to command one single zombie to generate constant-like rate attack traffic. Therefore, we may observe the stealthy behavior of the attacker.

Various types of attack traffic in term of throughput, system CPU utilization and memory utilization are presented in Figure 7, Figure 8 and Figure 9, respectively. After the initial 30 seconds, the throughput jumps abruptly and achieves the maximum till the end of the attack period (Figure 7). This is because the zombie is programed to direct attack traffic to the victim at its maximum capacity. In Figure 8, we found that the system CPU usage is vulnerable to ICMP flooding. In ICMP flooding, the attacker overwhelms the victim with ICMP echo request packets,

large ICMP packets, and other ICMP types to increase system CPU utilization, thereby slowing down the victim. We can conclude that ICMP flooding is CPU bound attack. The ideal way to deal with such attack is to ban zero-sized UDP packets via **netfilter**. On the other hand, the other three attacks have little effects on system CPU utilization. When flooded with DDoS attack messages, the victim uses up all its memory (Figure 9). The four types of attack traffic constantly and evenly saturate the victim's memory. The victim host is then unable to perform operations that need additional memory.

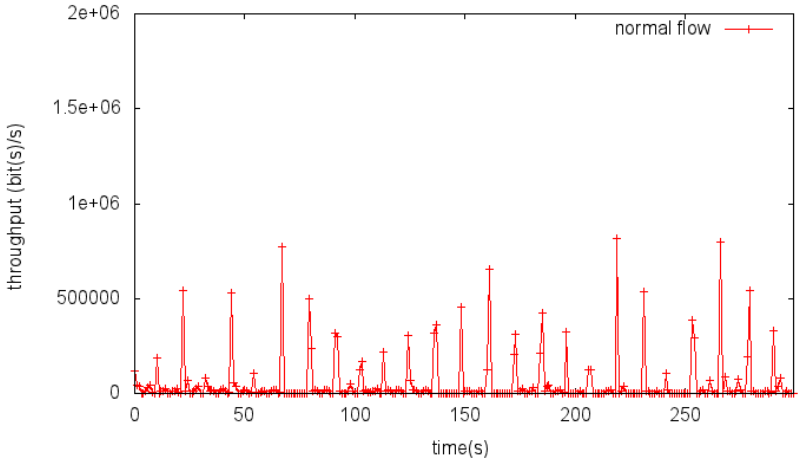

Figure 5. Throughput (bit(s)/s), low scale normal traffic

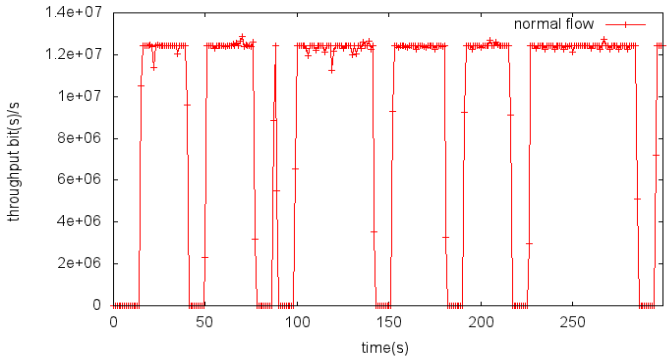

Figure 6. Throughput (bit(s)/s), large scale TCP normal traffic

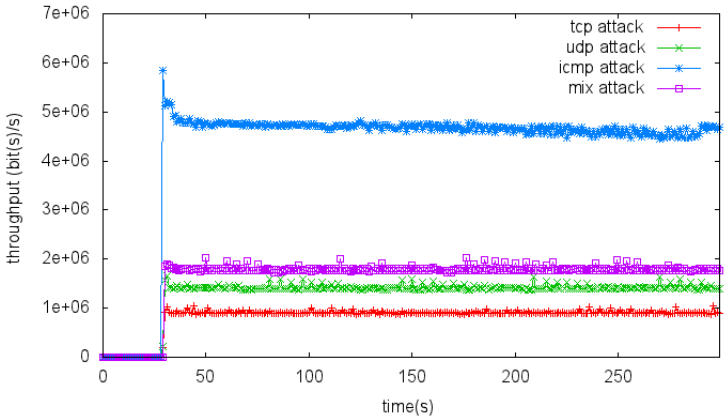

Figure 7. Throughput (bit(s)/s), constant rate attack traffic

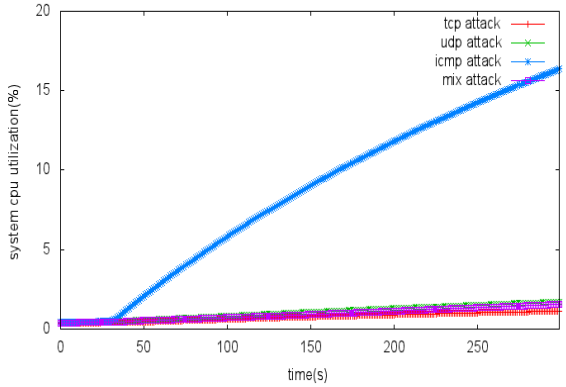

Figure 8. System CPU utilization (%), constant rate attack traffic

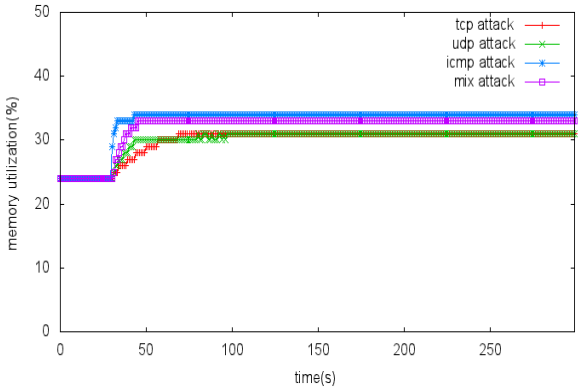

Figure 9. Memory utilization (%), constant rate attack traffic

In order to simulate the real DDoS traffic, we replace rand ( ) with /dev/random in flood.c, ip.c and tribe.c of TFN2K. /dev/random is a special file that serves as a blocking pseudorandom number generator. The attacker triggers n zombies for every $1/\Delta$ second, where n is randomly selected from 1 to 4. On receiving the commands from the attacker, the zombies start to forward the attack traffic to the victim. In Figure 10, we found that the throughput of randomized attack

flooding grows variably over time, which is in contrast to that of Figure 7. Therefore, we may conclude that the throughput on the increase (Figure 10) or a sudden change in the average incoming traffic (Figure 7) cannot be used as the only evidence of a DDoS attack. Figure 11 represents resource utilization with TCP SYN flooding between 40 and 300 seconds at 1 second interval. TCP SYN flooding stimulates a surge of both system CPU utilization and memory utilization. The data shows that the memory utilization grows faster even than we have expected. The reason is that TCP SYN flooding exploits a memory exhaustion issue inherent in the design of the TCP protocol. TCP SYN flooding initiates many connections without completing three-way handshake, until the victim is exhausted and has no memory left to track the TCP connection state for normal traffic.

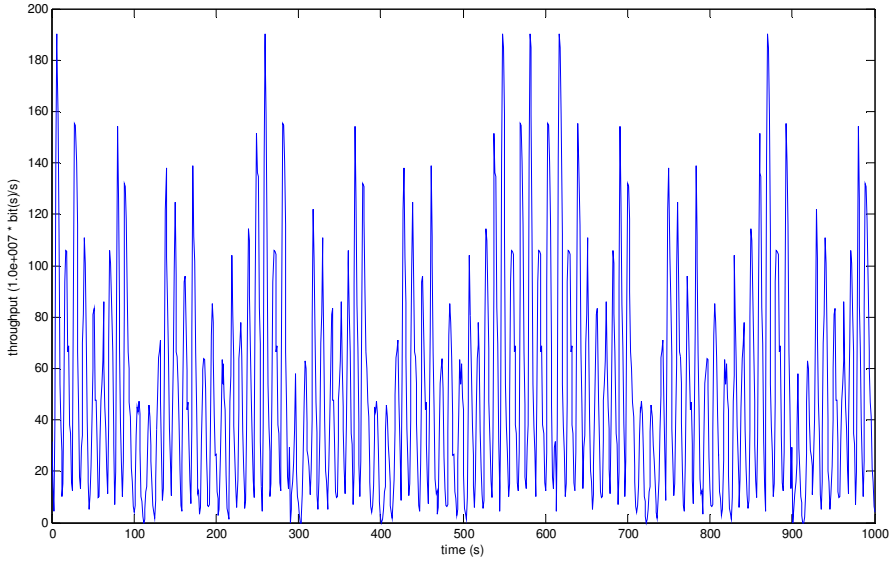

Figure 10. Throughput (bit/s), randomized attack traffic flooding

3.2. The minimum cost to detect the attack traffic

The minimum cost here is said to be *k* and *s*. Let *k* be the minimum percentage of system resource and *s* be the saturation time, respectively, exhausted by the attack traffic, such that the system service is disrupted. Table 3-5 represents the evaluation result of parameter Th_f, Th_p and Th_m, respectively, for four types of attack traffic, which are measured by system operators 10 times. Table 6 indicates the statistics of attack traffic. We run 1,000 samples for each trial. From Table 3, we can find that the lowest value of Th_f, 908,874 bits/s, is for TCP SYN Flood. Therefore, we choose 900,000 bits/s as the recommended value of Th_f. Similarly, the lowest value of Th_p, 2.6×10^{-3} %, is for TCP SYN Flood (Table 4) and the lowest value of Th_m, 1,096 bits/s, is for UDP Flood (Table 5). We choose 3.0×10^{-3} % and 1,000 bits/s as the recommended value of Th_p and Th_m, individually.

Table 3. Evaluation result of parameter Th_f, attack traffic

Type of Attack	Th_f (packets/s)	Th_f (bits/s)
TCP SYN Flood	17,149	908,874
UDP Flood	32,504	1,415,646
ICMP Flood	45,509	4,647,782
Mix Flood	26,131	1,762,852

Table 4. Evaluation result of parameter Th_p, attack traffic

Type of Attack	Th_p (%)	k (%)	Saturation time (s)
TCP SYN Flood	$2.6x10^{-3}$	0.7	38,571
UDP Flood	$4.4x10^{-3}$	1.2	22,500
ICMP Flood	$6.1\ x10^{-2}$	16.6	1,627
Mix Flood	$4.1x10^{-3}$	1.1	24,545

Table 5. Evaluation result of parameter Th_m, attack traffic

Type of Attack	Th_m (bits/s)	k (%)	Saturation time (s)
TCP SYN Flood	1,781	7	40
UDP Flood	1,096	7	65
ICMP Flood	6,995	11	16
Mix Flood	4,578	9	20

3.3. Correction of detection result

Suspicious traffic should be further identified by comparing the mean x and variance D of the throughput of normal traffic. d is represented as the standard deviation. Table 6 and 7 indicates statistics of large scale normal traffic and attack traffic on 10 trials, respectively. The large scale normal traffic are for data obtained in 10-second interval at every o'clock from 8 AM to 5 PM. The attack traffic is generated randomly by TFN2K. We use 360 samples (6/min × 60 min(s)/hour) for each trial. We measure using a test statistic that has an F-distribution with (k–1, n–k) degrees of freedom. In this case, k=2 and n=360. If the $\boldsymbol{P}$-value computed from the samples is less than the level of significance, α, we have evidence against the null hypothesis. That is, we reject the null hypothesis and say that the result is statistically significant. From Table 8, we can find that only the P-value of trial 3 is great than 0.05, we do not reject the null hypothesis. That is, the throughput means of normal traffic and attack traffic is not statistically significant.

Table 6. Throughput statistics of normal traffic on 10 trials

Trial #	Mean (x) (1.0e+008 * b/s)	Standard deviation (s) (1.0e+008 * b/s)	Variance (D) (1.0e+008 * b/s)
1	0.0652	0.0204	4.1695
2	0.0639	0.0203	4.1309
3	0.0649	0.0203	4.1032
4	0.0651	0.0202	4.0728
5	0.0644	0.0206	4.2405
6	0.0655	0.0202	4.0630
7	0.0653	0.0197	3.8744
8	0.0653	0.0204	4.1675
9	0.0649	0.0204	4.1805
10	0.0655	0.0199	3.9488

Table 7. Throughput statistics of randomized attack traffic on 10 trials

Trial #	**Mean (*x*) (1.0e+010 * b/s)**	**Standard deviation (*s*) (1.0e+010 * b/s)**	**Variance (*D*) (1.0e+010 * b/s)**
1	0.0581	0.0468	2.1873
2	0.0511	0.0407	1.6578
3	0.0517	0.0419	1.7575
4	0.0460	0.0366	1.3377
5	0.0589	0.0467	2.1822
6	0.0506	0.0410	1.6798
7	0.0468	0.0388	1.5028
8	0.0485	0.0400	1.5964
9	0.0482	0.0388	1.5018
10	0.0570	0.0463	2.1429

Table 8. One-way ANOVA statistics on 10 trials

Trial #	***F***	**P-value**
1	19.21	1.42197e-05
2	58.86	8.45398e-14
3	2.90	0.0893
4	48.55	9.81709e-12
5	34.45	7.82179e-09
6	40.55	4.23702e-10
7	8.66	0.0034
8	4.21	0.0407
9	26.01	4.77702e-07
10	37.93	1.47365e-09

In this section, we validate and tune the model to find the optimal value of parameter α_f, α_m and α_p. False negative rate (FNR) is the possibilities of identifying attack traffic as non-defective, while false positive rate (FPR) is the possibilities of recognizing normal traffic as defective. We estimate FNR and FPR in the presence of both attacks traffic and normal traffic. Figure 11(a) represents the FNR and FPR for the counter threshold of throughput, α_f, respectively. FNR grows as the value of α_f increases. We found that FNR becomes stable in case of α_f >6. On the other hand, FPR declines slightly in the beginning and remains steady in case of α_f>6. Both FNR and FPR are so high because they mistake attack traffic for benign traffic, and mistake high-rate normal traffic for attack traffic as well. However, if we use one-way ANOVA to further test group variation by considering time-of-day variation of normal traffic, both of the FNR and FPR decrease dramatically (Figure 11(b)). Combing one-way ANOVA is shown to be more effective to differentiate between attack traffic and normal traffic.

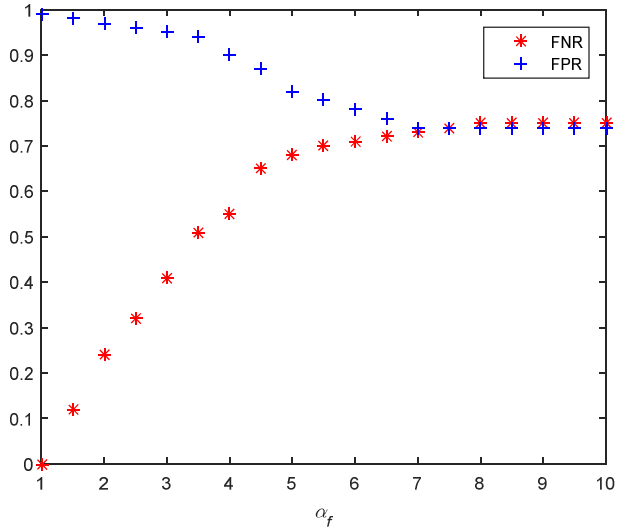

Figure 11(a). FNR and FPR for the counter threshold of throughput (α_f)

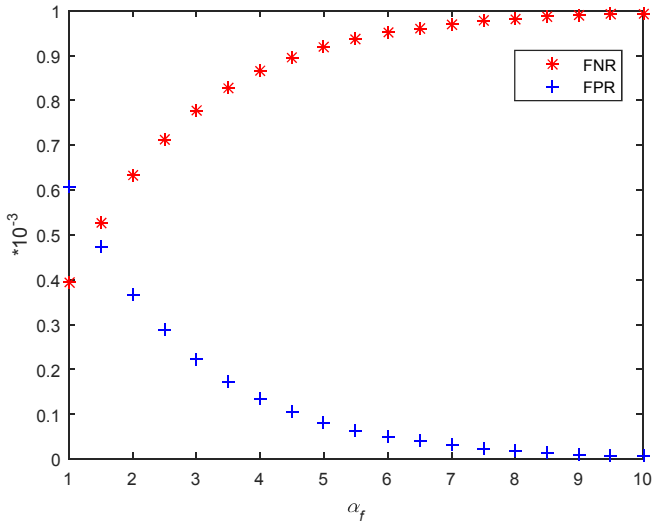

Figure 11(b). FNR and FPR for the counter threshold of throughput (α_f) with AVOVA test

Figure 12(a) shows both of the FNR and FPR for the counter threshold of memory utilization, α_m. FNR surges as α_m rises up to 25. However, FPR plummets more than 50 % as α_m grows up to 50. A lower α_m is fast to detect attack traffic, but may cause erroneous detection of normal traffic as attack traffic (causing higher FPR). On the flip side, a higher threshold value is needed to take time to detect the attack traffic, and thus is easily to identify the attack traffic as the normal traffic erroneously (causing higher FNR). Both of FNR and FPR drop significantly when introducing one-way ANOVA test to further discriminate the normal traffic and the attack traffic (Figure 12(b)).

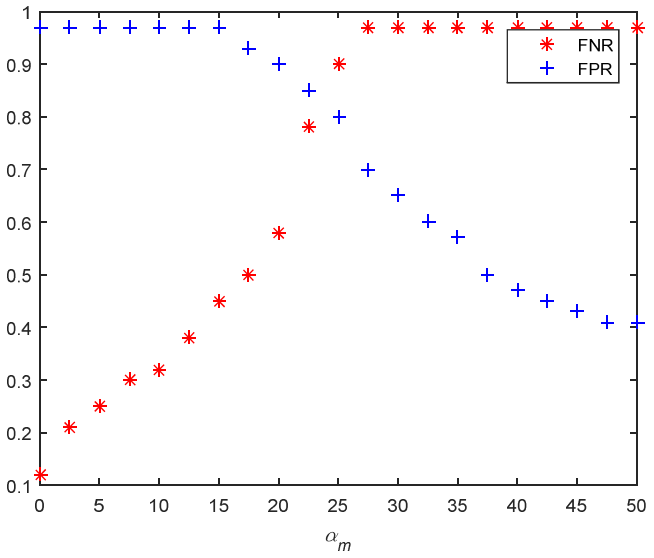

Figure 12(a). FNR and FPR for the counter threshold of memory utilization (α_m)

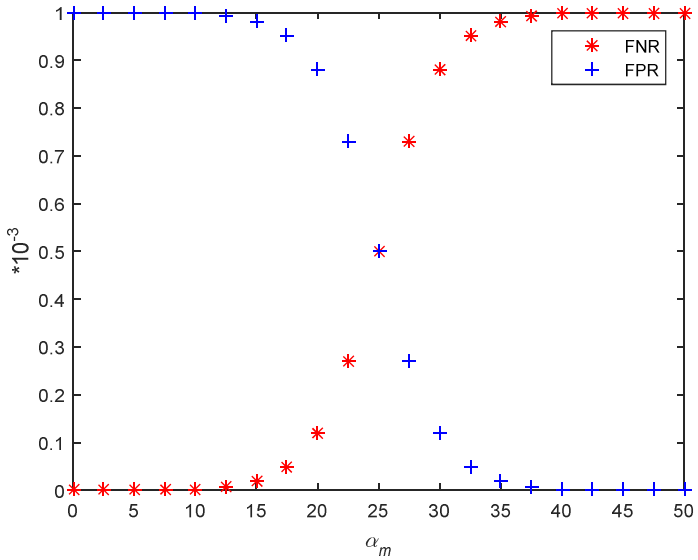

Figure 12(b). FNR and FPR for the counter threshold of memory utilization (α_m) with AVOVA test

Figure 13(a) has shown the FNR and FPR for evaluating the counter threshold of system CPU utilization (α_p). As we can see, increasing the value of α_p does not have high impact on FPR in case of α_p>3. On the other hand, the FNR increases gradually when α_p exceeds 1.5. As mentioned above, both of the FNR and FPR drop significantly if additionally running one-way ANOVA test (Figure 13(b)). We consider α_p=2.5 as an optimal value since it keeps FNR and FPR at low value at the same time.

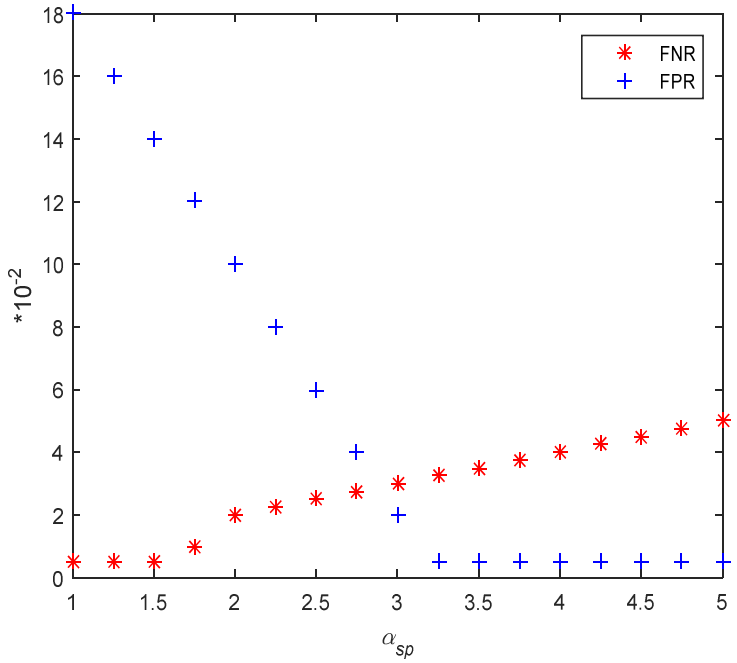

Figure 13(a). FNR and FPR for the counter threshold of system CPU utilization (α_p)

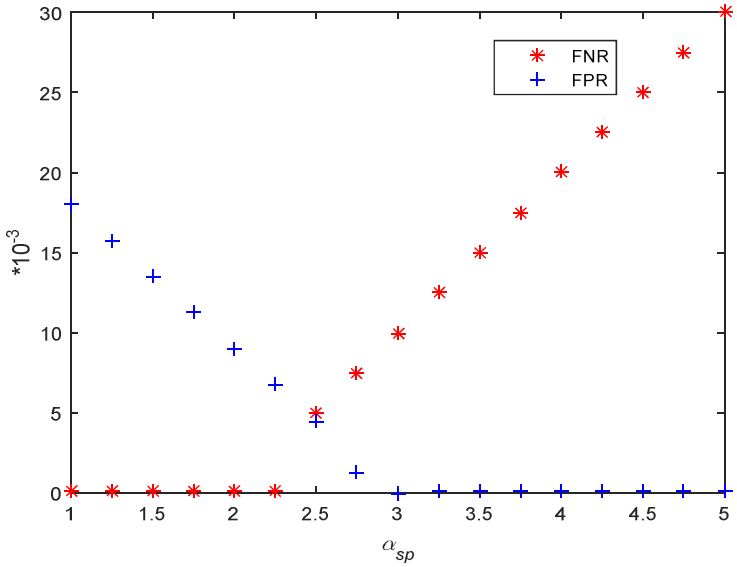

Figure 13(b). FNR and FPR for the counter threshold of system CPU utilization (α_p) with AVOVA test

In the following, we compare the proposed scheme with the other two– Scalable detection [6] and Live bait [7] by studying effectiveness in terms of FPR and FNR. In order to detect attackers in an effective way, we have to choose the detection threshold properly, which can be derived from the results of above mentioned experiments. We pick α_f=1, α_m=25 and α_p=3 as the optimal value of the proposed scheme. As shown in Figure 14(a), the FPR increases as the increasing number of zombies for all of three schemes. The proposed scheme has better performance than the other two since it further analyzes performance in terms of throughput, CPU utilization and memory utilization. Scalable detection has much higher FPR in the presence of a large number of zombies, which cause a higher number of imbalance counter of bad flows [6]. Figure 14(b) has shown that

FNR increases as the number of zombies grows for Live Bait while remains constant for both of the proposed and Scalable detection. The reason is that Live Bait does not effectively deal with increasing number of attackers, which generate low-rate request as legitimate traffic [7]. As mentioned above, the proposed scheme can effectively detect DDoS attacks with additional consideration of ANOVA test.

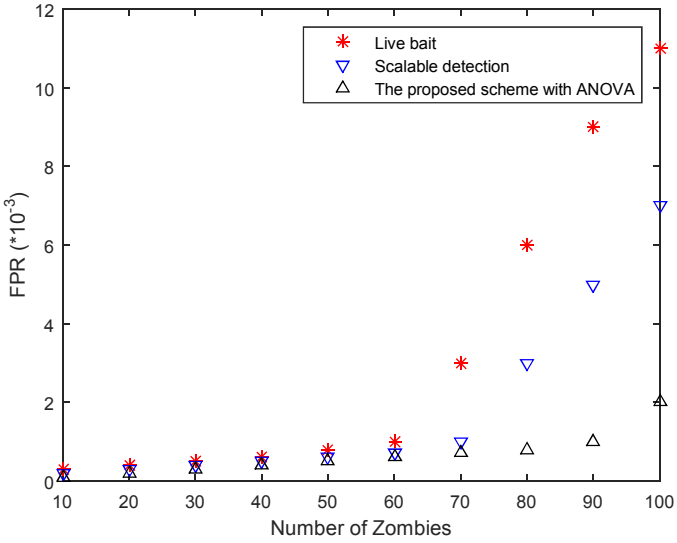

Figure 14(a). FPR comparison

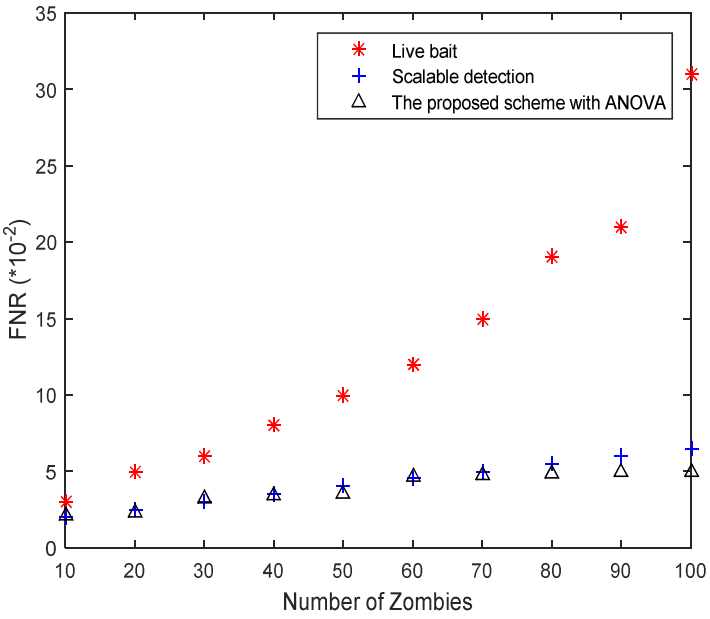

Figure 14(b). FNR comparison

4. CONCLUSIONS

In the paper, we has proposed a novel rule-based DDoS detection scheme along with ANOVA test, in which three types of system resource usage are examined. The strength of this paper is that it makes use of real traffic traces captured at the east campus of Pingtung University. Furthermore, we analyse the performance of the proposed system under the conditions imposed by both of the normal traffic and the TFN2K attack. Secondly, we find the minimum cost, such as the saturation time and critical point, for attack traffic to saturate the victim. Thirdly, a thorough investigation on comparison of the proposed scheme and the other well-known schemes is presented. Our analysis and experiments demonstrate that the proposed scheme can work very well with suitable combination and fine tuning of threshold value. In the future, we would like to implement the real-time traffic detection in a high-speed network.

REFERENCES

[1] Shang, Y., Luo, W. and Xu, S. (2011) "L-hop percolation on networks with arbitrary degree distributions and its applications," Phys. Rev. E 84, 031113.

[2] Yu, S., Zhou, W., Doss, R. and Jia, W. (2011) "Traceback of DDoS Attacks Using Entropy Variations," IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 3.

[3] Soundar Rajam, V. K., Selvaram, G., Pradeep Kumar M. and Mercy Shalinie S. (2013) "Autonomous system based traceback mechanism for DDoS attack," 2013 Fifth International Conference on Advanced Computing (ICoAC).

[4] Yu, S., Zhou, W., Guo, S. and Guo, M. (2013) "A dynamical deterministic packet marking scheme for DDoS traceback," GLOBECOM 2013 - IEEE Global Telecommunications Conference, Vol. 32, No. 1.

[5] Kiremire, A., Brust, M and Phoha, V. (2014) "Topology-dependent performance of attack graph reconstruction in PPM-based IP traceback," CCNC 2014 - 11th IEEE Consumer Communications and Networking Conference, Vol. 11, No. 1.

[6] Kompella, R. R., Singh, S. and Varghese, G (2007) "On scalable attack detection in the network," IEEE/ACM Transactions on Networking, Vol. 15, No. 1, pp14-25.

[7] Khattab, S., Gobriel, S., Melhem, R. and Mosse, D. (2008) "Live Baiting for Service-Level DoS Attackers," INFOCOM 2008, IEEE - 27th Conference on Computer Communications.

[8] Liu, H., Sun, Y., Valgenti V. and Kim, M. (2011) "TrustGuard: A flow-level reputation-based DDoS defense system," CCNC 2011 - 8th IEEE Consumer Communications and Networking Conference, Vol. 8, No. 1.

[9] Yoon, M., Li, T., Chen, S. and Peir, J.-K. (2011) "Fit a Compact Spread Estimator in Small High-Speed Memory," IEEE/ACM Transactions on Networking, Vol. 19, No. 5.

[10] Salah, K., Elbadawi, K. and Boutaba, R. (2012) "Performance Modelling and Analysis of Network Firewalls," IEEE Transactions on Network and Service Management, Vol. 9, No. 1.

[11] François, J., Aib, I. and Boutaba, R. (2012) "FireCol: A Collaborative Protection Network for the Detection of Flooding DDoS Attacks," IEEE/ACM Transactions on Networking, Vol. 20, No. 6.

[12] Gangam, S., Sharma, P. and Fahmy, S. (2013) "Pegasus: Precision hunting for icebergs and anomalies in network flows," IEEE INFOCOM 2013 - 32th IEEE International Conference on Computer Communications, Vol. 32, No. 1.

[13] Wang, Y., Zhang, Y., Singh, V., Lumezanu C. and Jiang, G. (2013) "NetFuse: Short-circuiting traffic surges in the cloud," ICC 2013 - IEEE International Conference on Communications, Vol. 36, No. 1.

[14] Chen, Y., Ma, X. and Wu, X. (2013) "DDoS Detection Algorithm Based on Preprocessing Network Traffic Predicted Method and Chaos Theory," IEEE Communications Letters, Vol. 17, No. 5.

[15] Kiruthika Devi, B.S., Preetha, G., Selvaram, G. and Mercy Shalinie, S. (2014) "An Impact Analysis: Real Time DDoS Attack Detection and Mitigation using Machine Learning," 2014 International Conference on Recent Trends in Information Technology.

[16] Jog, M., Natu M. and Shelke, S. (2015) " Distributed capabilities-based DDoS defense," International Conference on Pervasive Computing (ICPC).

[17] Zargar, S.T., Joshi, J. and Tipper, D. (2013) "A Survey of Defense Mechanisms against Distributed Denial of Service (DDoS) Flooding Attacks," IEEE Communications Surveys & Tutorials, Vol. 15, No. 4.

[18] Liu, Y., Chen, W. and Guan, Y. (2012) "A fast sketch for aggregate queries over high-speed network traffic," IEEE INFOCOM 2012 - 31th IEEE International Conference on Computer Communications, Vol. 31, No. 1.

[19] Sadre, R., Sperotto, A. and Pras, A. (2012) "The effects of DDoS attacks on flow monitoring applications," NOMS 2012 - 13th IEEE/IFIP Network Operations and Management Symposium, Vol. 13, No. 1.

[20] Rontti, T., Juuso, A.-M. and Takanen, A. (2012) "Preventing DoS Attacks in NGN Networks with Proactive Specification-Based Fuzzing," IEEE Communications Magazine, Vol. 50, No. 9.

[21] Khor, S. and Nakao, A. (2011) "MI: Cross-layer malleable identity," ICC 2011 - IEEE International Conference on Communications, Vol. 34, No. 1.

[22] Vashist, A., Chadha, R., Kaplan, M. and Moeltner, K. (2012) "Detecting communication anomalies in tactical networks via graph learning," MILCOM 2012 - IEEE Military Communications Conference, Vol. 31, No. 1.

[23] Wei, W., Chen, F., Xia, Y. and Jin, G. (2013) "A Rank Correlation Based Detection against Distributed Reflection DoS Attacks," IEEE Communications Letters, Vol. 17, No. 1.

[24] Xie Y. and Yu, S.-Z. (2009) "Monitoring the application-layer DDOS attacks for popular websites," IEEE/ACM Transactions on Networking, Vol. 17, No. 1, pp15-25.

[25] Ranjan, S., Swaminathan, R., Uysal, M., Nucci, A. and Knightly, E. (2009) "DDoS-shield: DDoS-resilient scheduling to counter application layer attacks," IEEE/ACM Transactions on Networking, Vol. 17, No. 1, pp26-39.

[26] Wang, J. (2011) "Web DDoS detection schemes based on measuring user's access behavior with large deviation," GLOBECOM 2011 - IEEE Global Telecommunications Conference, Vol. 30, No. 1.

[27] Lua, R.-P., Wah, C. and Ng, W. (2014) "Cornstarch Effect: Intensifying flow resistance for increasing DDoS attacks in autonomous overlays," CCNC 2014 - 11th IEEE Consumer Communications and Networking Conference, Vol. 11, No. 1.

[28] Liu, L., Jin, X., Min, G. and Xu, L. (2012) "Real-Time Diagnosis of Network Anomaly Based on Statistical Traffic Analysis," 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications.

[29] Purwanto, Y., Kuspriyanto, Hendrawan and Rahardjo, B. (2014) "Traffic anomaly detection in DDoS flooding attack," 2014 8th International Conference on Telecommunication Systems Services and Applications (TSSA).

[30] Toulouse, M., Minh, B.Q., Curtis, P. (2015) "A Consensus Based Network Intrusion Detection System," 2015 5th International Conference on IT Convergence and Security (ICITCS).