D-SAT: Detecting SYN flooding Attack by Two-stage statistical approach

Seung-won Shin, Ki-young Kim, Jong-soo Jang
Electronics and Telecommunications Research Institute in Korea
E-mail: {swshin, kykim, jsjang}@etri.re.kr

Abstract

We propose D-SAT (Detecting SYN flooding Attack by Two-stage statistical approach) system that is simple and robust approach to detect SYN flooding attacks by observing network traffic. Instead of managing all ongoing traffic on the network, D-SAT only monitors SYN count and ratio between SYN and other TCP packets at first time. And it detects SYN flooding and finds victims more accurately in its second stage. To make the detection mechanism robustly and easily, D-SAT uses CUSUM (cumulative sum) approach in SPC (statistical process control) [2] [11] [12]. It makes the detection mechanism much more generally applicable and easier to implement. D-SAT also employed AFM (Aggregation Flow Management) for finding victims quickly and accurately. The trace-driven simulation results demonstrate that D-SAT system is efficient and simple to implement and prove that it detects SYN flooding accurately and finds attack in a very short detection time.

1. Introduction

Many sites on the Internet have been the targets of (distributed) denial of service (DoS) attacks, among which TCP SYN flooding attack is the most prevalent [1]. To defend SYN flooding attacks, a lot of defense mechanisms have been presented, for example Syn cookies[5], Syn cache[7], SynDefender[6], Syn proxying[8] and Synkill[9]. All of these approaches should be installed at the firewall of the victim server or inside the victim itself. Above approaches defend victims efficiently, but they do not prevent network from lots of wasted SYN packets, because the defense line is close to the victim and network resources are wasted by huge amount of SYN packets. To compensate these, Haining W et al proposed new technique defending SYN flooding attack at the leaf router. Their approach protects network resources from exhaustion with very simple and robust mechanism [2]. We also made our focus like Haining W et al, because it protects victim and also avoids network resources consumption. However, although their approach is simple and efficient to detect SYN flooding and protect network resources, it has a problem to maintain all incoming TCP sessions. We examined our traffic traces and found that hundred thousands sessions have to be maintained. Hence, system trying to store all of session information should have large storage spaces. To overcome this, researches about session management are proposed using timeout or other threshold value, but these makes decrease the possibility to detect all suspicious flows. For this reason, we tried to find method to detect SYN flooding without storing huge amount of sessions.

In this paper, we present and evaluate D-SAT (Detecting SYN flooding Attack by Two-stage statistical approach) system for detecting SYN flooding efficiently. Our main focus is to solve problem mentioned above paragraph and detect SYN flooding without too much burden. D-SAT system detects SYN flooding by two-stage approach. At the first stage, instead of monitoring all ongoing packets on the network or the victim server itself, D-SAT system only counts the received number of TCP SYN packets and other TCP packets (TCP packets without SYN flag) and stores them. These received SYN count information and ratio between TCP SYN packets and other TCP packets are only parameters that D-SAT system inspects ahead. If SYN flooding attacks happen, SYN number increases and D-SAT finds it by above parameters. SYN count information shows us dynamic behavior of TCP SYN. However, it presents energetic changes in most periods [4], so we add the ratio information between SYN packets and other TCP packets to our consideration. Because the ratio information does not change seriously in normal status [4], it can compensate the instability of SYN count. If serious change happens in both parameters, D-SAT system thinks that it is possible to occur SYN flooding

Proceedings of the 2005 Symposium on Applications and the Internet (SAINT'05)
0-7695-2262-9/05 $ 20.00 IEEE

and goes to the next stage. In second stage, D-SAT system tries to find victims and detect whether SYN flooding really happens or not. Under SYN flooding attack, the victim server receives much more SYN packets than other servers. D-SAT searches those servers receiving lots of SYN packets and compares receiving rate with other servers. If the comparison results say that a few servers receive too much than others, D-SAT concludes that it is an outbreak of SYN flooding and those servers may be victim.

The trace-driven simulation showed that D-SAT system is very efficient to detect SYN flooding and protect network resources with little burden. Simulation results presented that D-SAT system feels the sign of SYN flooding in a mili-second order and find victim server in a second order. In addition to this, because D-SAT system only stores state information when it feels SYN flooding, it can reduce session management overhead until SYN flooding happens.

The rest of this paper is organized as follows. In Section 2, we describe the D-SAT system and explain statistical approaches that we used. Section 3 shows the simulation environment and analysis results and our discussions are also provided. Finally, future works and conclusion are drawn in Section 4.

2. D-SAT system

Unlike the traditional network-based intrusion detection systems that exist close to or on the victim, the defense line of D-SAT system is at the leaf router like Haining W et al's proposal [2]. The main goal of D-SAT system is to protect both of victims and network resources without high cost. The two-stage detection approach of D-SAT system makes this possible. In normal status, D-SAT system does not monitor the entire flows on the network. It just watches SYN packets and other TCP packets and stores count about them. If D-SAT system becomes aware of the beginning of SYN Flooding, then it investigates all TCP flows and inspects whether SYN flooding comes about, finally it uncovers which is victim. Consequently, it does not need lots of storage area for session information, until it tries to find attack. For detecting SYN flooding without all session information, D-SAT system uses two parameters such as, received SYN number and the ratio between SYN packets and TCP packets with the exception of SYN packets. The tally of received SYN shows us imminent happening of SYN storm and the ratio gives us a broad and much more steady situation of network [4]. D-SAT system also employs CUSUM approach to test parameters. This method can detect drifts from normal state [4]. If D-SAT finds a symptom of SYN flooding by above statistical process control method, it stores session information and uncovers which is the victim.

2.1 First stage: feel the sign of SYN flooding

At first stage, D-SAT system tries to feel the sign of SYN flooding. D-SAT system monitors SYN packets and other TCP packets and stores both items in every 10ms. And it applies statistical process control approach to these parameters. The detail process and statistical algorithm are explained in following sections.

2.1.1 CUSUM algorithm

The CUSUM (cumulative sum) chart was specifically designed to detect drifts from target; it does this by plotting the cumulative errors (sum of differences from target) of a set of previous measurements [3]. There are two ways to represent CUSUM, the tabular (or algorithmic) cusum, and the V-mask form of the cusum. Of the two representations, the tabular cusum will be preferable, because V-mask scheme is much less sensitive to shifts in the process mean [11].

The tabular (or algorithmic) cusum is like followings. Let x_i be the ith observation on the process. When the process is in control and stable, x_i has a normal distribution with mean u and standard deviation σ. We assume that either σ is known or that an estimate is available. Sometimes we think of u as a target value for the quality characteristic x_i. This viewpoint is often taken in process industries when the object is to control x_i to a particular target value. If the process drifts or shifts off this target value, the cusum will signal, and an adjustment is made to some variable to bring the process back to target. The tabular cusum works by accumulating derivations from u that are above target with one statistic C_i^+ and accumulating derivations from u that are below target with another statistic C_i^-. The statistic C_i^+ and C_i^- are called one-sided upper and lower cusums, respectively [11] [12]. They are calculated as follows:

$$C_i^+ = \max[0, x_i - (u + K) + C_{i-1}^+] \quad \text{-- (1)}$$

$$C_i^- = \max[0, (u + K) - x_i + C_{i-1}^-] \quad \text{-- (2)}$$

,where the start values are $C_0^+ = C_0^- = 0$.

K is usually called the reference value (or the allowance or the slack value), and it is often chosen about halfway between the target u and the out-of-

control value of the mean u' that we are interested in detecting quickly. If the shift is expressed in standard deviation unit as $u' = u + \tau\sigma$ (or $\tau = |u' - u| / \sigma$), then ***K*** is one-half the magnitude of the shift or ***K*** = $|u' - u| / 2$.

2.1.2 How to feel the sign of SYN flooding

To detect SYN flooding, we used received SYN number and ratio between SYN packets and other TCP packets information. The parameters that D-SAT has are like followings.

[*Parameter A*]
Received number of SYN packets

[*Parameter B*]

$$\frac{\text{Received SYN packets}}{\text{Received TCP packets except SYN packets}}$$

Both parameters are scanned and stored in every 10 ms, because the unit of time interval employed in D-SAT system is 10 ms. The Parameter A reflects current change of TCP SYN packets, so we can see the alteration of SYN packets by this parameter at once. However, it can also make us confuse deciding, because it varies very dynamically [4]. To compensate this, we add Parameter B. Parameter B does not fluctuate seriously and keeps its status for a while [4], so it can give us much more stable and static view of SYN packets. Due to their characteristics, we give a name in each parameter as; Parameter A – the dynamic view of SYN packets, Parameter B – the static view of SYN packets.

D-SAT system applies CUSUM method into both parameters to find change. As it uses CUSUM to test Parameter A (the dynamic view of SYN packets), it can get the knowledge of the sudden change of current SYN packets. Also, it employs CUSUM for Parameter B (the static view of SYN packets) to know whether the transition occurs or not. To find the change of each Parameter, we used CUSUM method like followings.

We tried to get the C_i^+ value, because we just know the sudden increase of SYN packets. Hence, our CUSUM approach is limited in finding C_i^+. From the equation of (1), we match the variables of equation (1) to our data. The x_i means both Parameters and u means the target value that we define. This target value is very important, because it is the criterion to find change. We decided this value larger than 20 for Parameter A, because we know that the received number of SYN packets in 10 ms does not exceed the 20 in most cases from our previous research [4]. For Parameter B, we chose the value larger than 0.2, because it does not exceed 0.2 in our previous research [4]. Therefore the final equation for our test is as follows.

[For Parameter A]

$$C_i^+ = \max[0, x_i - (20 * \alpha + K) + C_{i-1}^+] \text{ -- (3)}$$

,where x_i is the received number of SYN packets and α is an integer larger than 1 and ***K*** is a real number larger than 0

[For Parameter B]

$$C_i^+ = \max[0, x_i - (0.2 * \beta + K) + C_{i-1}^+] \text{ -- (4)}$$

,where x_i is the ratio between SYN packets and other TCP packets and β is an integer larger than 1 and ***K*** is a real number larger than 0

From Equation (3) and (4), we get the C_i^+ value that shows the change of increasing of both parameters. In addition to the C_i^+, we add new parameter AN_i and BN_i for each parameter A and B to validate change easily. The AN_i and BN_i are:

AN_i = 1 if C_i^+ of parameter A > 0, or 0 if C_i^+ of parameter A = 0 -- (5)

BN_i = 1 if C_i^+ of parameter B > 0, or 0 if C_i^+ of parameter B = 0 -- (6)

If the events that the AN_i and BN_i are larger than 0 continues, then we know that the change of increasing keeps going. Therefore, if the cumulative sum of both AN_i and BN_i are larger than threshold value, we think the number of SYN packets are suddenly growing and can assume that SYN flooding happens.

2.2 Second stage: detect SYN flooding and find victims

Based on the result of first stage, D-SAT decides whether monitoring all flows or not. If it thinks that SYN flooding happens (the cumulative sum of both AN_i and BN_i are larger than threshold value), it changes its mode into second-stage. In second-stage, it monitors all TCP flows and manages them through AFM (Aggregation Flow Management).

2.2.1 AFM (Aggregation Flow Management)
To find victims and validate attacks, D-SAT system stores TCP flows classified by destination IP address, after detecting the sign of SYN flooding at the first stage. In most SYN flooding attack, lots of attackers send SYN packets to target, so it makes a lot of TCP flows or generates new flows on the network. And there are huge amount of flows that have various kinds of source IP addresses and restricted target addresses. Based on this belief, we make D-SAT handle TCP flows by destination IP address. Although D-SAT has source IP address and Source and Destination Port and other flow information, it collects TCP flows by destination IP address.

2.2.2 Find Victims
Two parameters that we have used to detect SYN flooding are also employed in second stage. D-SAT searches AFM table and uncover what flows have the large value of each Parameter than other flows. After that, it sorts flows by each values and calculate the ratio of them. If servers are targeted by SYN flooding attackers, they receive much more SYN packets than other servers. Hence, in the case of victims, they show us large number of each parameter's value and finally we just find them. This kind of process is summarized as following pseudo-code and equations. (We assume that there are ***m*** flows on the network)

/* Variable Definition */
A(n) = the sum of Parameter A value of n_{th} flow for ϕ seconds,
B(n) = the sum of Parameter B value of n_{th} flow for ϕ seconds

/* Pseudo-Code Start */
```
i = 0;
while (flow exist)
      i = i + 1;

/* Store element that has maximum value */
      Top_Flow_A(i) = max [A(0), A(1), … A(m)];
      Top_Flow_B(i) = max [B(0), B(1), … B(m)];

/* Delete current element that has maximum value */
      Delete_List_From_A(Top_Flow_A(i));
      Delete_List_From_B(Top_Flow_B(i));
End
```
/* Pseudo-Code End */

From above pseudo-code, we can get ***Top_Flow_A(i)*** and ***Top_Flow_B(i)*** array of elements that have maximum value of each parameter from 0 to ***m***. Using these arrays, D-SAT calculated the ratio like equation (7) and (8).

$$Ratio\_A(i) = \frac{Top\_Flow\_A(i)}{Top\_Flow\_A(i-1)} \quad \text{-- (7)}$$

,where $i > 2$

$$Ratio\_B(i) = \frac{Top\_Flow\_B(i)}{Top\_Flow\_B(i-1)} \quad \text{-- (8)}$$

,where $i > 2$

It is very easy to understand both equations. If ratio value is smaller than 0.5 at i_{th} point, we easily know that the SYN sending rate of i_{th} flow is just smaller than 50% of $(i-1)_{th}$ flow. And finally D-SAT finds the point that both values of Ratio_A(i) and Ratio_B(i) are smaller than threshold value. D-SAT thinks that the servers have flows before that point are victims, and other flows after the point are not bothered by SYN flooding.

3. Simulation and Results

To evaluate and validate the D-SAT system, we have made trace driven simulation experiments. We have collected traces from A Research Institute network. Our simulation program is written in C language and matlab script language and running on Linux platform and Matlab program. Our test results are divided into each stage; the first stage that feels the SYN flooding, the second stage that find victims.

3.1 Traffic Traces and Attack Traces
For this analysis, we have used traces collected on A Research Institute (about 2000 people work for it) in Korea. These traces were taken on April 28, 2003 and April 29, 2003 on a 150 Megabit Ethernet connecting A Research Institute to the Internet. Traces have about 6 or 7 million packets. Traffic data were dumped by tcpdump [10] tool and stored by its own format. We think that there is only few malicious packets in our traces, because its internal network is protected by high performance IDS (Intrusion Detection System) and Firewall.
Although we got the normal traffic traces, we could not obtain SYN flooding attack traces. We made attack traffic by IXIA traffic generator [13]. It can produce various kinds of traffic and can control traffic sending rate. Recent experiments with SYN flooding attacks on commercial platforms [14] show that the minimum flooding rate to devastate an unprotected server is 500

SYN packets per second. So we make IXIA sends more than 500 SYN packets in a second with our background traffic. We made 4 SYN flooding attack traces with changing sending rate from 500 to 1200. And our attack traces have 4 target victims. The background traffic of them has a lot of normal IP packets and its send rate is about 12000 packets per second. UDP and TCP and ICMP packets are in the background traffic, but most packets are TCP.

3.2 Simulation Environment

Our simulator is composed of two main components. The first component is Data Extractor. It parses the traffic traces and extracts various kinds of parameters such as, time, received TCP and SYN number in 10 ms and IP Address. This component is written in C language and running on Linux platform. The other component is Data Analyzer. Data Analyzer reads parameters extracted from Data Extractor and apply our two-stage approach to them. It is running on Matlab program.

3.3 First-stage results: Feeling SYN Flooding Attack

For the first time, we have applied proposed CUSUM approach on all the normal traffic traces without attack. Figure 1 and Figure 2 represent the normal traffic behavior of both parameters. The test statistics, C_i^+ of each parameter and $\boldsymbol{AN_i}$ and $\boldsymbol{BN_i}$, for all traces are plotted in those Figures. In Figure 1, C_i^+ of Parameter A and $\boldsymbol{AN_i}$ are represented and C_i^+ of Parameter B and $\boldsymbol{BN_i}$, are shown in Figure 2. We selected α as 1.5 and $\boldsymbol{K}$ (the shift size) as 2 in Equation (3). For Equation (4), we chose β as 1.5 and $\boldsymbol{K}$ as 0.1. For all the traces tested, C_i^+ of each parameter and $\boldsymbol{AN_i}$ and $\boldsymbol{BN_i}$, are mostly zeros. The isolated bursts in C_i^+ are always much smaller than the value that can change each $\boldsymbol{AN_i}$ and $\boldsymbol{BN_i}$. So, no detection signals are reported. We also applied our proposed scheme on the attack traces. This simulation results of both parameters are plotted in Figure 3, and it shows that the CUSUM results exceeds the flooding threshold "25" (SYN threatening is increasing for 0.25s) for both of $\boldsymbol{AN_i}$ and $\boldsymbol{BN_i}$. The test values increase when the SYN flooding attack comes about. Once the flooding attack is detected, D-SAT system moves to second stage. At this point, we have to think about other important issue of detection time. In some sense, the detection time becomes another question, because it can affect to the performance of defense system. If detection system did not find SYN flooding attack for a long time, it could not protect victims from the attack (it was already damaged by attack). Because of this, defending system has to detect attack as soon as possible. D-SAT finds attack very quickly within a second at its first stage, because our proposed scheme is very sensitive to abrupt change. The differences between attack initiating time and detection time are summarized in Table 1.

3.4 Second-stage results: Detecting SYN Flooding and Find victims

To verify victims, D-SAT gathers flow information during 10 seconds (it means ϕ is equal to 10 seconds) and manages with the help of AFM. In Figure 4, we present top 5 flows sorted by the accumulation of both parameters. The simulation results of Ratio_A(i) and Ratio_B(i) are shown in Figure 5. From Figure 5, we easily know that top 4 flows are most dominant on the network and other flow is so trivial. Our attack traces have SYN flooding packets to 4 different victims. So, we compared our detection results with 4 victims and confirmed D-SAT system verified 4 victims exactly (D-SAT finds 4 victims that have destination IP Address such as, 192.168.10.12, 192.168.10.13, 192.168.10.14, 192.168.10.15).

4. Conclusion and Future Works

This paper presented D-SAT system that is a simple and robust SYN flooding detection mechanism without serious flow management algorithms. D-SAT uses two-stage detection approach. At its first stage, it just watches SYN count and ratio value and detects the initiation of SYN flooding. It finally decides SYN flooding and finds victims at its second stage. The distinguishing features of D-SAT include, it does not monitor all flows until SYN flooding starts and requires low storage spaces and computation power; it is immune to SYN flooding itself by its two-stage approach; CUSUM approach is employed, making it robust and simple. We demonstrated the efficiency and robustness of D-SAT by using trace-driven simulations. The simulation results prove that D-SAT detects SYN flooding accurately and finds attack in a very short detection time

Our future work will include making D-SAT system in independent program and combining with other statistical approach and using testing other network traffic traces. And, we will implement our system into productive software and install it on real network.

References

[1] D. Moore, G. Voelker, and S. Savage, "Inferring Internet denial of service activity", In Proceedings of USENIX Security Symposium, 2001

[2] Haining Wang, Danlu Zhang, Kang G. Shin, "Detecting SYN Flooding Attacks", In Proceedings of IEEE INFOCOM, 2002

[3] David Drain, "Statistical Methods for Industrial Process Control", Chapman & Hall, 1997

[4] Seung-won Shin, Ki-young Kim, Jong-soo Jang, "Analysis of SYN Traffic: An Empirical Study", Technical Document in ETRI, 2004

[5] D. J. Berstein and Eric Schenk, "Linux Kernel SYN Cookies Firewall Project", http://www.bronzesoft.org/projects/scfw

[6] Check Point Software Technologies Ltd. SynDefender http://www.checkpoint.com/products/firewall-1

[7] J. Lemon, "Resisting SYN Flooding Dos Attacks with a SYN Cache", In Proceedings of USENIX BSDCon'2002, 2002

[8] Juniper Networks Integrated Firewall Appliance, http://www.juniper.net

[9] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram and D. Zamboni, "Analysis of a Denial of Service Attack on TCP", In Proceedings of IEEE Symposium on Security and Privacy, 1997

[10] tcpdump/libpcap, http://www.tcpdump.org

[11] Douglas C. Montgomery, "Introduction to Statistical Quality Control", WILEY, 2001

[12] Douglas M. Hawkins, David H. Olwel, "Cumulative Sum Charts and Charting for Quality Improvement", Springer, 1998

[13] IXIA traffic generator, http://www.ixiacom.com

[14] T. Darmohray and R. Oliver, "Hot Spares for Dos attacks", ;login, 25(7), July 2000

ID	Attack Start Time	Detect Time	Time Difference
Attack 1	29.57 s	29.88 s	0.31 s
Attack 2	64.26 s	64.68 s	0.42 s
Attack 3	88.98 s	89.27 s	0.29 s
Attack 4	55.69 s	55.95 s	0.26 s

Table 1: Attack Start and Detect Time

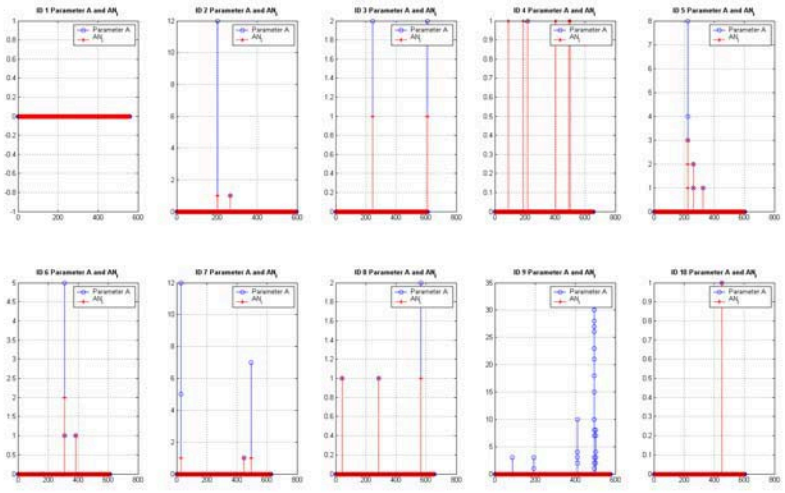

Figure 1. CUSUM test results of Parameter A (normal traffic traces)

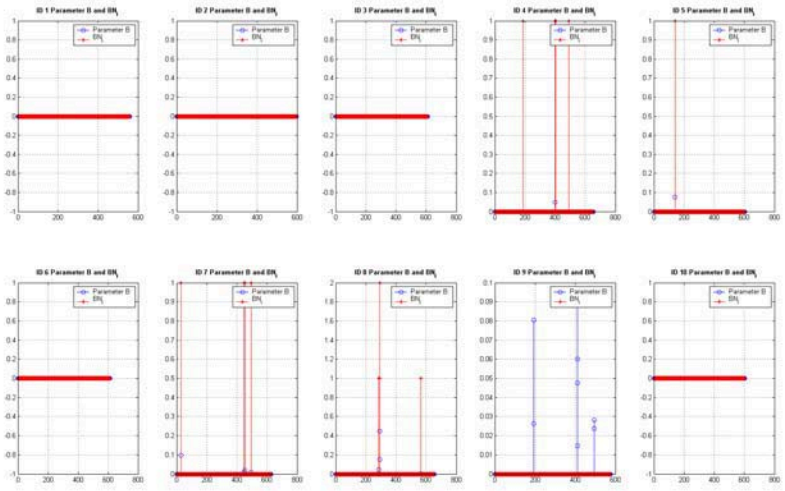

Figure 2. CUSUM test results of Parameter B (normal traffic traces)

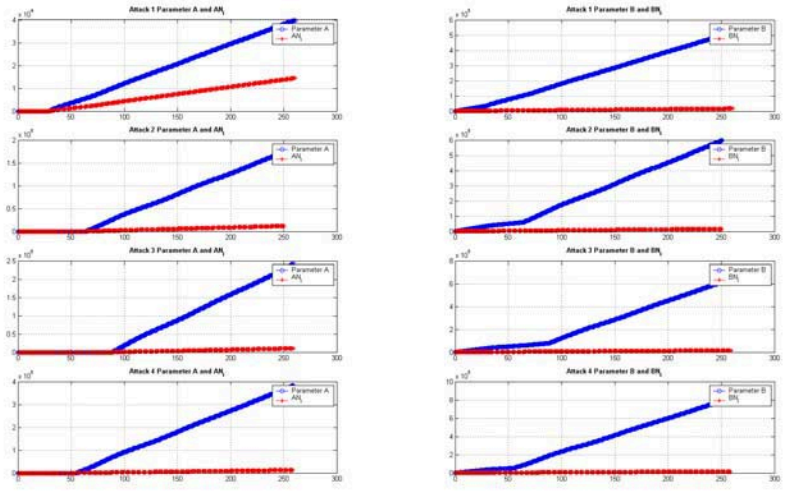

Figure 3. CUSUM test results of Parameter A and Parameter B (Attack traffic traces)

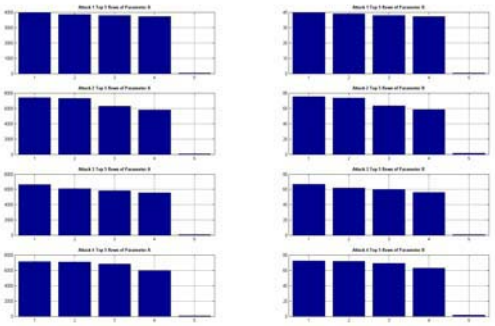

Figure 4. Parameter A and Parameter B of Top 5 flows

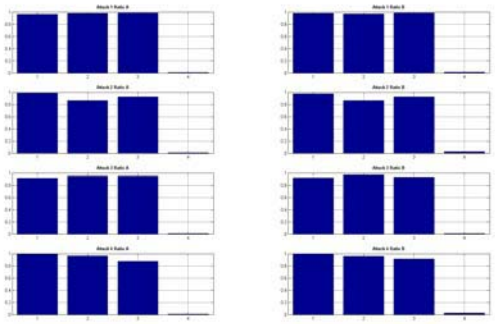

Figure 5. Ratio_A and Ratio_B of Top 5 flows