SYN Flooding Attack Detection Based on Entropy Computing

Conference Paper · January 2010

DOI: 10.1109/GLOCOM.2009.5425454 · Source: IEEE Xplore

CITATIONS
18

READS
434

2 authors:

Martine Bellaiche
Polytechnique Montréal
31 PUBLICATIONS **435** CITATIONS
SEE PROFILE

Jean-Charles Grégoire
Institut National de la Recherche Scientifique
119 PUBLICATIONS **643** CITATIONS
SEE PROFILE

SYN Flooding Attack Detection Based on Entropy Computing

Martine Bellaïche
Génie Informatique et Génie Logiciel
École Polytechnique de Montréal
Montréal, QC, CANADA
email: martine.bellaiche@polymtl.ca

Jean-Charles Grégoire
INRS–EMT
Montréal, QC, CANADA
email: gregoire@emt.inrs.ca

***Abstract*—We present an original approach to detect SYN flooding attacks from the victim's side, by monitoring unusual handshake sequences. Detection is done in real-time to allow quick protection and help guarantee a proper defence. Our detection system uses an entropy measure to detect changes in the balance of TCP handshakes. Experiment results show that our method can detect SYN flooding attacks with better accuracy and robustness than traditional stateless methods, and with manageable overhead.**

***Index Terms*—Denial of Service, SYN Flooding, TCP Handshake, Entropy, Network Security.**

I. Introduction

Most internet based services rely on the TCP protocol. Establishment of TCP connections is based on a handshake, more specifically a 3–way handshake (exchange of 3 packets), to reserve and announce suitable resources at both ends before data exchange can proceed. This mechanism has however proven to be quite vulnerable to attacks. A denial of service (DoS) attack has for objective to stop legitimate users from using a service. A distributed DoS attack (DDoS) extends the concept to a large number of attacking nodes.

SYN flooding attacks represents 90 % of a large number of DDoS attacks [1]. Any Internet-based service using the TCP protocol such as the Web, FTP or mail is a potential target of a TCP SYN flooding attack. The goal of the attack is to tie the memory of server machines with half–open connections. Many spoofed clients send an important number of connection set-up requests to a single server and, consequently, legitimate clients cannot connect any more to the server, whose resources have been depleted.

The server receiving these SYN packets sends a SYN/ACK packet to the spoofed address. The server will never receive the final ACK packet, which would have completed the handshake, because the SYN/ACK packet was sent to the machine whose address was (most often) spoofed and thus ignored. In the best case, the server may receive a RST (connection reset) packet from that machine.

When the server's backlog queue is filled with such SYN packets, it can no longer accept any SYN packet from legitimate users trying to connect. To terminate these waiting connections, the server has a timer to purge from the backlog queue any half–open connection which exceeds the time–out value.

As the attack packets used in the SYN flooding are not different from normal packets, except for the spoofed address source, it is difficult to distinguish them from the normal packet intended for the victim server. This is why SYN flooding attacks are difficult to detect.

In this article, we describe and classify all different forms of TCP handshakes which may occur during a connection setup between a client and a server (e.g. for Web traffic). During a denial of service attack, some sequences of handshakes are unusual, can be related to the attack and used to detect it. We propose an effective detection mechanism, which allows to quickly identify DoS attacks based on the presence of such unusual handshakes. These handshakes are examined through a detection system based on an entropy measure which reflects a variation in information content through time.

Our technique, which we call Unusual Handshake Detection (UHD), is robust in face of network congestion and possible router errors, either of which can cause variations in the behaviour of TCP handshakes. It is meant to be fast, accurate, and simple to implement, aiming to minimize the reaction time required to set up a defence against the attack.

This paper is organized as follows. Section II looks over previous work. Section III exposes the abnormal forms of the TCP protocol handshake. Section IV describes the architecture of our detection technique and further investigates the collection and decision modules. Section V evaluates our detection method. And to finish, section VI presents a discussion and conclusions.

II. Previous Work

Much research has been done on various forms of DoS attacks. Schuba and al. [2] analyse in depth SYN flooding attacks and propose a remedial measure. Their monitor, SYNKILL, has the capability of monitoring and injecting traffic in the network. SYNKILL examines all TCP packets on a LAN and classifies IP addresses as `null`, `good`, `new`,

978-1-4244-4148-8/09/$25.00 ©2009

bad. When it observes bad addresses, SYNKILL sends a RST (TCP reset) packet to the source.

Blazek and al. [3] check the TCP packet control bits, ICMP packets and UDP packets and apply the classical CUmulative SUM (CUSUM) method to detect a change during an observation period.

Wang and al. [4] monitor the normal behaviour of TCP where there must be a correspondence between numbers of TCP FIN packets (or RST) and TCP SYN packets over all TCP connections. They record the number of SYN and FIN (or RST) packets and detect discrepancies.

Jin and al. [5] exploit the principle that the TCP Internet traffic may be described by a degree of correlation between the control bits of TCP packets. On the basis that abnormal traffic's correlation will differ from normal traffic's, the correlation indicates a change of activity in Internet traffic. The authors analyse the control bits' covariance matrix to detect a change.

Siris and al. [6] count the number of TCP SYN packets and propose to apply two algorithms of SYN flooding attack detection over the number of SYN packets: the adaptive threshold algorithm and the CUSUM algorithm.

Ohsita and al. [7] demonstrate that the SYN packet arrival rate of the normal traffic may be modelled by a normal distribution. A normal traffic corresponds to TCP flows having completed their handshake and terminated with a FIN or RST packet. A SYN flooding attack is detected when there is an important variation between the normal distribution and the SYN packet arrival rate.

Lim and Uddin [8] validate the semantics of proper TCP behaviour, namely that the arrival of a SYN packet implies also later arrival of ACK packets from/to the same source. The authors evaluate the difference between SYN and ACK packets normalized by a Exponential Weighted Moving Average (EWMA) of SYN packets.

Xiao and al. [9] determine if a detected half-open connection results from network congestion or from SYN flooding attacks. The authors have developed an active method to verify either condition: network congestion is estimated by a method which sends ICMP echo packets to routers to record the round–trip time (RTT) between a client and the server. This technique's challenge is to obtain the RTT when there is network congestion.

Shin and al. [10] count two parameters: the number of SYN packets and its ratio to the number of other TCP packets (i.e. excluding SYN). During an increase of the count of TCP SYN packets, there will be a SYN flooding attack if the ratio changes quite significantly. They have developed the D-SAT system which counts two parameters: the number of TCP SYN packets and the ratio between the number of SYN packets and all other TCP packets. The CUSUM method is applied to detect a parameter's change. Thereafter, they identify the victims by classifying TCP flows by destination IP address and by finding the flows with the largest values for those parameters.

The main idea of Al-Duwairi and al. [11] is to intentionally reject the first SYN packet for every connection request. The retransmitted SYN packet is going to be accepted only if it satisfies the TCP time–out mechanism. This method introduces a latency in connection set–up time, and we can have a memory overflow of the SYN packets rejected. Furthermore, if the attacker knows the detection system, it can quite simply send 2 spoofed SYN packets separated by a time–out.

Most of the strategies described above operate on TCP's normal behaviour by counting the number of packets. The weakness of these detection methods is that the attacker can send a mixture of different packets to thwart the detection. Also the style of strategy is vulnerable because the Internet traffic is bursty and the detection may therefore translate into false alarms (see e.g. [12]).

III. Usual and Unusual TCP Handshakes

In the analysis of packet traces containing no flooding attacks, we notice that there are TCP handshakes whose sequence does not follow the 3 steps standard. We name them unusual handshakes. Normally, those are the result of the network congestion and—sometimes—router errors or unreachable ports, but during DDoS attack, they can also be the result of the attack. To detect TCP handshake anomalies, it is necessary to identify the various TCP handshake sequences which can be part of a SYN flooding attack. We concentrate here on detection from the server side and look at handshakes from that perspective only. Our detection system will have to fix an upper bound delay value d (section IV-A). In the list below, we present the unusual handshake sequences which can be part of a DoS attack on a server victim. These handshake sequences are observed at the last mile router.

- **[SYN].** The server receives a SYN packet, but it cannot answer any more because it is overwhelmed. This connection will be ended after server time–out, as described above.
- **[SYN (Client, Server), RST (Client, Server)].** This handshake sequence cannot be identified as part of a denial of service attack, because it is the client who decided to terminate the connection request.
- **[SYN (Client, Server), RST (Server, Client)].** This sequence probably means that the server is the victim of a denial of service attack, since it cannot reply to the legitimate client any more. This sequence can also appear with a SYN/ACK, as **[SYN (Client, Server), SYN/ACK, RST (Server, Client)].** This situation arises because the server does not receive the corresponding ACK after the time-out, so the server closes the connection by sending a RST packet.
- **[SYN, SYN/ACK].** The server waits in vain for the ACK packet, either because the IP source address is spoofed, or because the ACK packet is rejected due to network congestion. This sequence can correspond

978-1-4244-4148-8/09/$25.00 ©2009

to a denial of service attack. This connection will be ended after server time–out.

- **[SYN, SYN/ACK, RST (Client, Server).** This handshake sequence can correspond to a DDoS attack. At the reception of the SYN/ACK packet, the client host then transmits a RST packet to the server, because it never sent a SYN packet.
- **[SYN, SYN/ACK, ACK].** This connection is a complete 3–way handshake sequence.

The detection module is placed at the last mile router and recognizes unusual handshakes as possible DDoS attacks, namely : (SYN, d), (SYN (Client, Server), RST (Server, Client)), (SYN, SYN/ACK, d), and (SYN, SYN/ACK, RST (Client, Server)) where d corresponds to a detection delay. d could be equal to the time–out of the server on the connection but, as we may not want to wait that long, we hold separately RTT information on flows to keep a tighter estimate of d (see below in section IV-A).

IV. Detection Technique Architecture

Our detection method supervises the TCP handshakes' unusual behaviour. The support architecture is broken down into 3 modules: the collection module, the decision module and the monitoring module. Again, we assume that detection is done in the victim's network, possibly at the last mile router.

The collection module acquires all the information necessary for the detection technique and organizes it according to the detection tactics. The decision module proceeds to the identification of an attack. It consults the information gathered by the collection process and, according to the method chosen, makes the decision to report an attack if necessary. The monitoring module executes the decision module, typically after the expiration of the observation period.

A. Collection module

The collection module watches passively the Internet traffic and collects all the information in the data structure developed in Bellaïche and al. [13].

- It classifies IP packets by filtering them to select only TCP SYN, TCP ACK and TCP RST packets, and stores the TCP flow information according to the last packet seen [1].
- It keeps track of an estimate of TCP connection latency (RTT, plus delay for memory allocation for the TCP data structure) per source network, to set the detection delay for the unusual handshake.

The data structure is a hashtable where the key represents the 24 bits prefix of source IP addresses, i.e. the source network's address. Each slot of the hashtable is a linked list of the flows from a given source subnetwork. For each flow, we store the IP source and destination addresses, the source and destination ports, the arrival time of the last SYN packet of a new TCP flow and the flag of the TCP packet. Initially, each slot of the hashtable is empty.

[1]Another method filters IP packet in order to preserve only the TCP control packet whose IP packet size is between 40, 80 bytes.

B. Decision module: Anomaly Detection Algorithm

The decision module uses an entropy measure derived from the number of usual and unusual handshake sequences for detecting attacks. Under normal traffic, the entropy value runs greater than a lower bound, but during SYN flooding attacks, we show that the change in the number of unusual handshakes will cause a variation in this value, bringing it below this bound. Note that, beyond SYN flooding attacks, another possible cause of variation in the entropy measure would be a simultaneous increase in the congestion of the different transit networks. It is however extremely unlikely that such events would occur simultaneously.

a) Entropy: Let us recall that, in information theory, the Shannon entropy or information entropy is a measure of the uncertainty associated with a random variable. In the detection of a flooding attack, the entropy detection method is mainly used to calculate the distribution randomness of some information extracted or derived from IP packet headers. For example, Feinstein and al. [14] compute and examine the entropy of source addresses with a window size of 10,000 packets.

In this paper, the number of unusual handshakes is chosen as the random variable. Recall that, if the information source has n independent symbols each with probability of choice p_i, then the entropy H is defined as [15]

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

To calculate the entropy, we choose four forms of unusual handshakes:

- p_1 corresponds to the fraction of the (SYN, d) handshake sequences over all unusual handshakes.
- p_2 corresponds to the fraction of the (SYN (Client, Server), RST (Server, Client)) handshake sequences over all unusual handshakes.
- p_3 corresponds to the fraction of the (SYN, SYN/ACK, d) handshake sequences over all unusual handshakes.
- p_4 corresponds to the fraction of the (SYN, SYN/ACK, RST (Client, Server)) handshake sequences over all unusual handshakes.

b) Alarm detection: First, we evaluate the threshold, that is, the lower bound T of the entropy to distinguish traffic without attack. For the detection, we could simply observe threshold violations but, following [6], we prefer to use Repeated Threshold Violations (RTV) which are better to avoid false alarms.

The difficulty is to choose an appropriate value for k, the number of repetitions, as it has an impact on detection latency. We must also be careful that some attacks, e.g.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

pulsing attacks, could have a window of activity overlapping a number of intervals less than k, and thus not be reported. Another difficulty is the case where some p_i will be 0, i.e. if some form of handshake is not present. We assume that there is a diversity (i.e. 2 or more) of unusual handshakes present in traffic, which we have observed in practice, and ignore null values in our computation.

C. Monitoring module

The monitoring module requires several elements. The most important is the data structure that keeps all TCP handshake information. The RTV algorithm is used to detect an attack with the threshold T, so we also have to choose a value for k. With P representing the duration of the monitoring module's observation period, we compute each p_i and the entropy H, and apply the RTV algorithm at every P.

Having defined all the elements of our detection technique architecture we evaluate in the next section our algorithm's detection performance with the following metrics: detection time, detection rate, false positive rate and memory space.

V. Evaluation of our Detection Method

In this section, we justify that the observation of the handshake sequence is an appropriate choice for the detection of SYN flooding attacks.

Our detection scheme can be evaluated according to detection time, detection rate, false positive rate and memory space. We have set up an experimental environment with several simplifications. First, we use a trace of bidirectional TCP traffic to represent normal traffic conditions. Furthermore, since we do not need to reproduce the distributed nature of the attack, it is not necessary to configure a particular (Internet) network topology. Our system of defence does not require packet path characteristics, only estimates of round–trip times, which can be chosen arbitrary—albeit consistent per source networks—for our tests. Finally, synthetic SYN flooding attacks, as described below, are merged with the traces.

As our traces came from real networks (UCLA [16]), it should come as no surprise that they already contained several attacks. It is indeed very difficult to obtain traces completely exempt from attacks, and it is thus important to consider their presence at the time of the performance evaluation. In our case, they had an impact on setting the threshold value.

A. Characteristics of DDoS Attacks

To test our detection system, we initialize the DDoS attack according to the following parameters to emulate a SYN flooding attack.

- The attack packets are—obviously—of type TCP SYN.
- To reproduce a situation which would lead to a real DDoS attack, we set an attack rate of 25 % of the mean SYN packet rate of normal traffic[2]. This rate is constant for the duration of the attack.
- The source address and the source port of the attack packets are generated with a uniform distribution [1].
- The value of the other fields of the attack packets are assigned arbitrary values, because they do not have any bearing on our detection method.
- The SYN flooding attacks starts at 100 s. The attack's duration is 300 s (5 minutes) and the inter–arrival time between two attacks is a constant value of 600 s [1]. There are 10 attacks in total (for the duration of the trace).

B. Parameters of the Detection System

Our detection technique requires setting some parameters. The choice of the detection parameters is very important and must be done very carefully.

- Before merging attack packets into the normal trace, we set the entropy threshold T to 0.35 for UCLA trace 9 and 0.3 for UCLA trace 5 — threshold which is the lower bound for an entropy measure without attacks and, as shown in figure 1, well below most measures. Remember that it is important to choose an adequate value for the threshold so that it does not produce false alarms in the absence of attacks, yet be high enough to avoid false negatives. Obviously, a suitable value is derived from the observation of various traces of normal traffic.
- The minimum number k of consecutive violations of the threshold is fixed to 3, to obtain a quick detection but also to identify pulsing attacks.
- The observation period P of the monitoring module is set at 10 s.

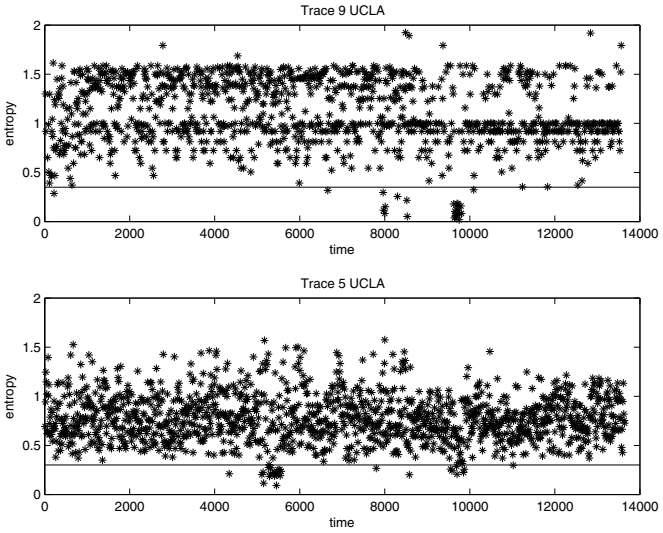

Fig. 1: Entropy without attack.

[2]The attack must be significant to have an impact, yet not dominate traffic, to remain difficult to detect.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

In figure 1, the x axis represents the time, the y axis the entropy measure and the horizontal line the threshold T. Entropy is computed for 2 different traces[3], named trace 5 and trace 9, taken form a UCLA dataset, as already mentioned.We can observe that there are some attacks in the UCLA traces, e.g. a short one in trace 9, in $[8000, 8500]\,s$ and an attack of $180\,s$ in $[9620, 9800]\,s$. In trace 5 of the figure, there are two real attacks in the intervals $[5110, 5560]\,s$ and $[9550, 9870]\,s$.

C. Performance Evaluation

As our objective is to quickly detect attacks and without any—or minimal—error, we evaluate detection performance by measuring the detection time, the detection rate and the rate of false alarms.

The performance metrics are estimated by merging attacks with the normal traffic trace. An attack is detected when the detection algorithm is executed.

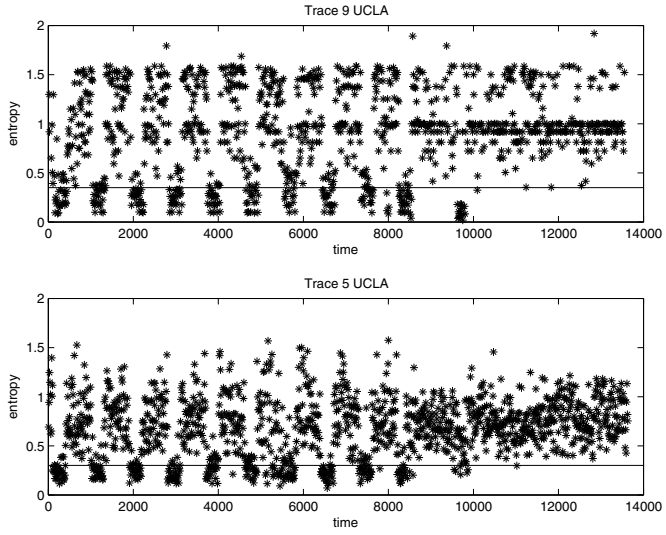

Fig. 2: Entropy with attack.

In figure 2 we observe 10 attacks. Entropy is measured 10 times below the threshold.The detection rate is 100 % and the false alarm rate 0 %. We also notice that an attack is detected between $[3, 7]$ observation periods of the monitoring module[4] after the beginning of the attack. This represents a fast detection knowing that a DDoS attack lasts an equivalent of 30 observation periods. The empty data structure occupies 67 MB of memory. Under normal conditions, we require extra memory which amounts to at most 16 Kbytes for the handshake information.

[3]We have conducted many more experiments with other traces and our detection system gave good results in all cases. However due to the page limitation, these results could not all be included in the paper.

[4]Since the observation period is $10\,s$, the detection time is therefore between $30\,s$ and $70\,s$.

D. Worst case: a clever attacker

Of course, if the attacker knows the principle of the detection, he can try to send a flood of SYN packets followed by ACK packets to keep a reasonable balance of SYN vs. non–SYN packets. This will be undetected unless we also keep track of TCP sequence numbers in the collection module. The server, however, would quite likely reset the connection because of wrong sequence numbers, which would lead again to another form of unusual handshake. As the principle of the detection is stateful, the attack can exhaust the memory with enough variety of unusual handshakes. But our detection is fast, and flow information is reset every period, so the attack will be detected quickly, before using all the router memory.

E. Comparaison with FDS

As introduced in section II, Wang and al. [4] have proposed a simple mechanism: the Flooding Detection System (FDS). The technique is characterized by its statelessness and a low computation overhead. The FDS utilizes the TCP SYN-FIN pairs' behaviour for SYN flooding detection. The random variable CUSUM is applied to the difference between the number of SYNs and FINs (RSTs) normalized by an estimated average number of FINs (RSTs). The observation period is set to 20 seconds and represents the average duration of TCP connections. We propose that our stateful technique is better, for a

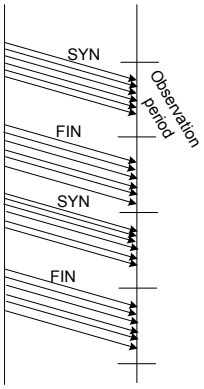

Fig. 3: Count of SYN and FIN Packets in the FDS technique.

variety of reasons:

- The FDS does not consider whether a SYN packet is retransmitted, which does not follow proper TCP behaviour that associates one FIN packet for one SYN packet. This discrepancy can lead to false alarms. Our detection takes retransmission into account.
- Their method depends on the average duration of the TCP connection. As we can observe in figure 3, in one observation period we could observe a large number of TCP connections with a duration significantly longer than the average, or alternatively a flash crowd, either of which could trigger a false alarm because the FIN

978-1-4244-4148-8/09/$25.00 ©2009

packet will be counted in a later observation period and, as a consequence, would lead to an imbalance between the count of SYN and FIN packets.
- Our detection technique is faster than the FDS technique, because the latter depends on the duration of the TCP connection and ours relies only on the duration of the TCP handshake. The detection time lies in the range of $[20\ s, 480\ s]$

VI. Conclusion and Discussion

In this article, we have described the architecture of a fast and precise DoS detection method. It contains 3 elements: collection, decision and monitoring modules, and their relevant configuration parameters (threshold for the entropy, observation period, number of consecutive violations). We have shown that the number of unusual handshakes is a good choice for a random variable for detection because it reflects the exploitation of the TCP protocol for flooding attacks. It is however required that several forms of unusual handshakes are present in the traffic, which has been our experience in practice.

Since the data structure is organized by source networks, after detecting the attack we can throttle or block the traffic from that source network for which the unusual handshakes have increased. However, clearly, a detailed description and evaluation of the above procedure falls beyond the limits of the present conference paper and is the subject of another study.

Our UHD detection method introduces several innovations and benefits:

- It exploits a new decision variable based on an entropy measure.
- It is independent from the traffic volume and does not produce false alarms during flash crowds [17].
- It does not introduce any additional traffic in the network [9].
- It is independent from the attack packets' characteristics [14].
- It is autonomous, so, it does not need other information from other routers.
- It is stateful, because it stores the TCP flow information to identify the nature of the handshake.

UHD would preferably be implemented on the last mile routers. It is not recommended to install the UHD on core routers, as it could not then detect all flooding sources and properly protect the server victim. As packets belonging to the same flow can follow different paths and as the UHD must monitor the exchange of the TCP protocol handshake packet, the UHD installation in core routers cannot count all the exchanged packet handshakes. On the other hand, detection at the victim is easier, because the attack traffic is aggregated.

The stateless detection methods identified in the section II are attractive because they consume little memory and are very easy to implement. On the other hand, their detection is slower and less reliable than with our stateful detection technique: detection of an attack is more precise with a stateful method and also results in fewer false alarms.

References

[1] D. M. G. Voelker, and S. Savage, "Inferring Internet denial-of-service activity," in *USENIX Security Symposium*, August 2001.

[2] 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 the IEEE Symposium on Security and Privacy*. IEEE Computer Society, 1997, p. 208.

[3] R. B. Blazek, H. Kim, B. Rozovskii, and A. Tartakovsky, "A novel approach to detection of denial of service attacks via adaptive sequential and batch sequential change point detection methods," in *Workshop on Information Assurance and Security, IEEE*, June 2001.

[4] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," in *Proceedings of IEEE INFOCOM*, 2002.

[5] S. Jin and D.S.Yeung, "A covariance analysis model for DDoS attack detection," in *IEEE International Conference on Communications*, vol. 4, June 2004.

[6] V. A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks," in *IEEE, GlobeCom*, December 2004.

[7] Y. O. M. Murata and S. Ata, "Detecting distributed denial-of-service attacks by analyzing TCP SYN Packets statistically," in *IEEE, GlobeCom*, December 2004.

[8] B. Lim and M. Uddin, "Statistical-based SYN-flooding detection using programmable network processor," in *Third International Conference on Information Technology and Applications ICITA*, July 2005.

[9] B. Xiao, W. Chen, Y. He, and S. E.H.-M., "An active detecting method against SYN flooding attack," in *Proceedings. 11th International Conference on Parallel and Distributed Systems*, July 2005.

[10] S.-W. Shin, K.-Y. Kim, and J.-S. Jang, "D-SAT: detecting SYN flooding attack by two-stage statistical approach," in *The Symposium on Applications and the Internet, IEEE*, Janurary 2005.

[11] B. Al-Duwairi and G. Manimaran, "Intentional dropping: a novel scheme for SYN flooding mitigation," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, vol. 4, March 2005.

[12] R. Mahajan, S. Bellovin, S. Floyd, J. Vern, and P. Scott, "Controlling high bandwidth aggregates in the network," in *Computer Communications Review*, July 2002.

[13] M. Bellaïche and J.-C. Grégoire, "Monitoring TCP handshakes on DDoS flooding," in *Submitted to Security and Communications Networks, John Wiley & Sons,*, July 2009.

[14] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *Proceedings in DARPA Information Survivability Conference and Exposition*. IEEE Computer Society, April 2003.

[15] C. Shannon and W. Weaver, "The mathematical theory of communication," in *University of Illinois Press*, 1963.

[16] "UCLA packet traces," http://lever.cs.ucla.edu/ddos/traces/.

[17] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proceedings of ICNP*, November 2002.

978-1-4244-4148-8/09/$25.00 ©2009