Stateful or Stateless Flooding Attack Detection?

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*—SYN flooding attacks exploit the 3-way handshake characteristic of TCP connection setup to cause denials of service. Many techniques have been proposed for the detection of flooding attacks; most are stateless while a few are stateful. A stateful method keeps specific information on flows of packets while stateless methods will only keep counters on specific packet features. The low performance impact of stateless methods has led to their predominance in practical deployments. We introduce a methodology to support a comparison between methods, which allows to quantify all key factors which can be used to assess and compare performance and see how they can be built into a metric. In this article, we evaluate and compare the performance of two key DoS detection techniques, one stateless and one stateful, and investigate their relative merits.**

***Keywords*—Denial of Service; SYN Flooding; TCP Handshake; 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 denial of service (DoS) attacks on servers, which have for objective to stop legitimate users from using a service by overloading it with connection establishment requests. A distributed DoS attack (DDoS) occurs when a large number of nodes wage such an attack simultaneously.

SYN flooding attacks represent 90 % of most DDoS attacks [1]. The goal of the attack is to tie the memory of server machines with half–open connections. A large number of attack machines 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.

Many techniques have been proposed for the detection of flooding attacks; most are stateless while a few are stateful. A stateful method keeps specific information on packets crossing the router while stateless methods will only keep counters. The low overhead of stateless methods has led to their predominance in practical deployments; yet we would want to know if a stateful method performs significantly better than a stateless one: As we use more memory and, to a lesser degree, more CPU in stateful techniques, we want significant improvements in detection time, detection rate and rate of false alarms to justify their use. We also want to know if they are more robust towards evasion of detection.

In this article, we describe several stateless and stateful techniques for flooding attack detection and compare the performance of two key techniques, representative of each kind. The originality of this work lies in: (1) reviewing of the state of the art in stateless and stateful attack detection, (2) presenting a method for evaluating performance detection system, and (3) comparing stateless and stateful methods to establish their relative merit.

This paper is organized as follows. Sections II looks over previous work in detection and isolates two key methods for our comparison. In Section III, we introduce elements of comparison between methods and proceed to an evaluation of one stateful technique vs. a stateless one in Section IV. We discuss our results and conclude in Section V.

II. State of the Art

Because of space constraint, we do not cover extensively the full range of stateless and stateful techniques, restricting our study to two, representative techniques and highlight differences which are specific to each kind. The reader will be referred to the bibliography for further reading.

A. State of the Art for Stateless Techniques

Stateless techniques use packet counting and statistical analysis (e.g. CUSUM) to detect an attack. Packet information is tallied in a random variable X_n over an observation period (and not continuously). X_n has taken many forms, based directly on protocols (Blazek and al. [2]), traffic correlation (Jin and al. [3]) or behaviour (Ohsita and al. [4]), the presence of specific packets or packet sequences (Siris and al. [5], Shin and al. [6]).

Wang and al. [7] propose a simple mechanism to detect SYN flooding attacks by monitoring the normal behaviour of TCP. Their stateless Flooding Detection System (FDS) has low computation overhead. For the normal behaviour of TCP, there must be a match between the number of TCP FIN (or RST) packets and TCP SYN packets over all TCP connections. Using the CUSUM method, they

 ISBN: 978-1-61208-133-5

record the number of SYN and FIN (or RST) packets and detect discrepancies. The difference between the number of SYN and FIN (RST) is normalized by an estimated average number of FIN (RST), to ensure that the FDS is independent of sites and access patterns. As most TCP connections last from 12 to 19 s, they set the duration of observation periods to 20 s.

The weakness of stateless detection methods is that the attacker can send a mix of packets to thwart detection. Also, such counting strategies are vulnerable because Internet traffic is bursty and the detection may therefore raise false alarms (see e.g. [8]). They also lead to rather long detection periods.

B. State of the Art for Stateful Technique

Stateful techniques rely on a memory of past events such as occurrence of source addresses (Schuba and al. [9]), analysis of current condition changes in traffic patterns due to congestion (Xiao and al. [10]) or other factors (Gil and Poletto [11], Cheng and al. [12]).

In [13] we have proposed the Unusual Handshake Detection (UHD) method. TCP handshakes whose sequence does not follow the 3 steps standard are recognized as unusual handshakes. Those are typically the result of network congestion and—sometimes—router errors or unreachable ports; but during DDoS, they can also be the result of the attack. This work concentrates on detection from the server side, at the last mile router, and looks at handshakes from that perspective only. A dedicated data structure stores all information on the TCP handshakes. For each flow, 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 are stored. The data structure 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 handshakes.

Stateful techniques require memory to support monitoring. How memory is managed is critical as the available space may be exhausted with increasing traffic [10] [11] [12] [13]. Such detection techniques must therefore be able to detect attacks very quickly to be resource–effective.

III. Framework for Comparaison

As we are interested in comparing stateful and stateless detection, we use and adapt the methodology we have presented in [14] for detection. Our purpose is to quantify all factors which can be used to assess and compare performance. It is also possible to construct an aggregate metric from the different factors used to evaluate the quality of detection to end up with a unique performance number.

In order to protect the victim efficiently, the essential objectives are to detect attacks quickly, with accuracy and with minimal deployment costs. Deployment costs will reflect the complexity of the detection method, measured according to the changes it requires compared to a defenceless service architecture. These overall objectives translate into the following criteria: accuracy, latency—or detection time, deployment cost and robustness, which can be related to specific measures.

A. Performance Measures

a) Detection Rate: The detection rate is the percentage of attacks that are detected as compared to the total number of attacks [5]. This metric—associated with the detection time—validates the detection mechanism for each attack. Similarly, the non-detection rate—or false negative rate—is a way of determining the errors made by defences for not identifying the attacks. It corresponds to the percentage of non-detected attacks compared to the total number of attacks. It is the complement of the detection rate.

b) Rate of False Positives: The rate of false positives or the rate of false detection alarms [15] is another way of assessing detection errors made by identifying an attack when none occurred. This rate is the ratio between the number of erroneously-reported attacks and the total number of attacks. This metric verifies that the detection mechanism does not make (significant) mistakes. For example, we want to know if an increase in traffic or a flash crowd can cause false alarms.

c) Latency: The detection time—or latency—metric reflects the delay in the detection of attacks. The detection time of the attack is the duration of the time interval between the beginning of an attack and its detection. The detection time is important because an attack should be detected before any severe damage is done.

The latency depends on a number of elements: there are architectural constraints, for example a polling cycle to acquire data, and algorithmic constraints, such as the existence of a time window to average the information over several acquisition cycles.

B. Deployment Costs

The deployment costs of the defence system depend on the computation time, the memory overhead, the bandwidth overhead and the system complexity as explained below. In fact, we want to evaluate the increase of these costs due to the deployment of the detection system. To evaluate the different elements, we need to perform two experiments: a first one to find the baseline value of deployment costs, in the absence of attacks and a second one to evaluate the increase from the baseline value.

1) *Computation overhead* in ms: The time required to process the measured data.
2) *Memory* in Kbytes: The storage space necessary for the implementation of the detection mechanism.
3) *Bandwidth overhead* in %: Should the detection method imply the transmission of some form of control messages (e.g. throttle), then this in turn would yield a reduction of the available bandwidth.
4) *Deployment complexity*: The deployment complexity, measured from 1 (low) to 4 (high cost). This measure

 ISBN: 978-1-61208-133-5

Deployment	Cost fct.	Priority
Complexity y	$f_y(y) = y \times Y$	p_y
Bandwidth overhead b	$f_b(b) = B$ (%)	p_b
Computation overhead c	$f_c(c) = C$ (ms)	p_c
Memory m	$f_m(m) = M$ (KB)	p_m

TABLE I: Cost Functions

depends on whether the detection strategy involves one or several nodes and whether it involves numerous and substantial modifications to the network.

Finally, the installation of the detection system should not increase the deployment costs, i.e. it should be integrated with another network device.

C. A Composite Metric

We have shown in [14] how these different measures can be combined into a composite metric, through a weighted sum to emphasise certain metrics. Such a composite gives a global evaluation of the objectives and quality of the method. Here, we show how to effectively compute such a metric. To that end, we need (1) a list of the relative priorities of these elements, (2) a cost function for each element to have uniform comparison units, (3) values for the weighting coefficient determined by the priority list.

The priority values are attributed according to the cost functions. A low priority value represents the cost of an easy deployment. For example, the composite metric for the deployment cost DC is expressed by:

$$DC = \alpha_c \times f_c(c) + \alpha_m \times f_m(m) + \alpha_b \times f_b(b) + \alpha_y \times f_y(y) \quad (1)$$

with computation overhead c, memory m, bandwidth overhead b, and deployment complexity y, and the matching weighting factors $\alpha\_$, the cost functions $f\_$ and the priority p between $[1, 4]$ (see Table I).

Similarly, for the performance measure, we build D as follows.

$$D = \alpha_l \times l + \alpha_n \times n + \alpha_p \times p \quad (2)$$

with latency l in s, rate of false negatives n in %, and rate of false positives p in %. Each performance measure has a priority p between $[1, 3]$. As the performance measure is not a cost, we do not use cost functions. An ideal detection technique must have a short latency l, as well as, a rate of false negatives n and positives p as low as possible.

We develop further in Section IV how the weighting factors can be chosen to build a meaningful composite.

D. Robustness

Robustness is a critical evaluation of a defence and, unlike previous metrics, it is difficult to define it in terms of a specific cost.

What we require, in our evaluation of defences, is the assessment of the effectiveness of the detection as an attack proceeds. In this case we cannot simply reduce such assessment to a unique metric, as we expect performance not to be constant, but to depend on the legitimate traffic load and characteristics.

For different quality factors, e.g. false positives, false negatives, latency, and sensitivity (low threshold) we want to identify the detection weaknesses:

1) **False Positives (or False Alarms)**: which traffic conditions increase the rate of false positives?
2) **False Negatives**: at which rate is it possible to avoid detection?
3) **Latency**: how significantly does detection increase latency?
4) **Sensitivity**: how do we establish a detection threshold?

On this last point, we note that whereas the value of the threshold is not too significant when a server is unloaded, it must be kept as low as possible when we have a high usage, to preserve useful traffic while providing detection. Sensitivity is thus denoted by the ability to set a threshold to allow detection while keeping rates of false positives and false negatives low.

We therefore propose that robustness be examined as a standardized test, at a specific usage level. We must note however that this picture is not complete as stateless measures can be fooled by specific forms of attacks which supply the relative number of TCP messages they expect, hence creating a large number of false negatives. Such an assessment is required to complement sensitivity/quality tests. In the following comparison, we will look at the behaviour of the specific parameters of robustness assessment without attempting to build a composite picture.

IV. Comparison between Stateful and Stateless Detection Methods

We choose the following techniques for a comparison according to the methodology presented above: the FDS of Wang and al. [7] for the stateless case and our own UHD [13] for the stateful one. These techniques use for the detection the behaviour observation of the TCP protocol and the CUSUM algorithm to confirm the attack.

The $\alpha\_$ values are set as follows.

- As memory is cheap, we assign 1 to the priority because it is not a significant contribution to deployment costs. We calculate $\alpha_m = 0.1$.
- As the processors are more and more powerful, for the computation overhead, we assign 2 to the priority and obtain $\alpha_c = 0.2$
- As we want save network resources, for the bandwidth overhead, we assign 3 to the priority and set $\alpha_b = 0.3$.
- To encourage minimal deployment complexity, we assign 4 to the priority $\alpha_y = 0.4$.

These values are chosen because some resources are easy to obtain and are not too significant, while some elements are very important and play a fundamental role in the computation of the deployment cost (DC).

A. Evaluation of FDS

a) Performance Measures: From [7], the detection rate is within the range $[70\%, 100\%]$ and the rate of false positives is null. The detection time is within $[20\,s, 487\,s]$.

 ISBN: 978-1-61208-133-5

We have set an interval for the overall evaluation of detection of $D = [6.6, 160.7]$. But in practice (see Section IV-A0c) the method can trigger false positives during flash crowds.

b) Evaluation of Deployment Costs: The computation overhead for the FDS system represents the time required for packet classification and addition, and is not evaluated in the article. As the stateful technique also needs packet classification, it is not very important to determine its computation overhead. Also, the addition operation time is rather insignificant. The computer overhead is therefore fixed to 0. For storage, FDS uses only two integers for compiling the number of SYN and FIN packets, so the memory is only 8 bytes. There is no bandwidth overhead. We fix the level of the deployment complexity to 1, because it is very easy. With the weighting coefficient and the cost function, we evaluate the composite metric to $DC = 0.1 \times 8 \times 10^{-3} \times f_m(m) + 0.4 \times 1 \times f_y(y)$.

c) Robustness: The count of SYN and FIN packets of the FDS technique is not very reliable for the following reasons:

False Positives or False Alarms. 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. In one observation period we could observe a large number of TCP connections with a duration significantly longer than average, or alternatively a flash crowd, either of which could trigger a false alarm because the FIN 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.

False Negatives. The weakness of counting SYN–FIN pairs is that the attacker can flood a mixture of SYNs and FINs in equal numbers. As the consequence, the clever attacker can evade the FDS detection technique.

Latency. As the observation period corresponds to the duration of TCP connections ($20\,s$), the detection time is the number of the observation periods. In most cases, the detection is therefore triggered after the TCP connection ends.

Sensitivity. FDS fixes the threshold and varies the attack rate to evaluate the detection rate.

For all these reasons, the robustness of FDS is low.

B. Evaluation of UHD

a) Performance Measures: As indicated in the paper, the detection rate is 100 % and the false alarm rate 0 %. The detection time is therefore between $30\,s$ and $70\,s$. With a weighting coefficient of 0.33, we evaluate the overall evaluation in between $[9.9, 23.1]$

b) Evaluation of Deployment Costs: We have used a form of XOR-folding, also called bit-folding or bit-extraction as a hashing function. It is a practical manipulation of bits combining shifts, masking and logical combinations. With XOR-folding, it is easy to construct a function which will be robust to the permutation of information; the only challenge is to find the combination which generates the most dispersion and this can depend on the nature (i.e. regional character) of the server. Such issues are however beyond the scope of this paper.

On the Intel Duo processor at 3.00 GHz used for our experiments, the insertion time and the scanning time are $3.95\,ms$ and $20.62\,ms$ respectively. These values are very small. As the networks are identified by 24 bits of an IP address, the hashing function uses XOR-folding of the two halves of the 24 bits address into 12 bits—for a basic table size of 16384 B. In this case, we have observed on the same data an average length of the chains of 1.65, and an occupancy rate of 31.7 %. Under normal conditions, we require extra memory which amounts to at most 21 KB for the handshake information.

The technique does not use any bandwidth for detection. The deployment is very easy and this technique can be built into a last mile router. With the weighting coefficients and the cost functions, DC evaluates as: $0.1 \times (21 + 16.384) \times f_m(m) + 0.2 \times (3.95 + 20.62) \times f_c(c) + 0.4 \times 1 \times f_y(y)$.

c) Robustness: As the principle of the detection is stateful, the attack can exhaust the memory with enough variety of unusual handshakes. But the detection is fast, and flow information is reset every period, so the attack is detected quickly, before using up all the router memory.

False Positives or False Alarms. As the technique does not count the packets, UHD is not vulnerable to flash crowds and consequently, it does not produce false alarms.

False Negatives. 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 data structure. The server, however, would quite likely reset the connection because of wrong sequence numbers, which would lead again to another form of unusual handshake.

Latency. The detection attack can be caught right from the beginning as the technique observes the TCP handshake. However, as the detection time is linked with the observation period, such a short observation period involves a quick detection time.

Sensitivity. From real traces and with merging fictitious SYN flooding attacks, we have run tests with an attack rate fixed at 25 % of normal traffic. In Figure 1, we show the importance of the value the entropy threshold. A "wrong"—or too tight—value can lead to numerous false alarms. As UHD detects attacks when the entropy value is below the threshold, to evaluate the sensitivity of this value to the detection of false alarms, we measure this rate while increasing the threshold. Moreover, the start value of the threshold represents the detection value of a trace. In Figure 1 we see that, as the threshold increases linearly beyond a threshold of .44, the number of false alarms

 ISBN: 978-1-61208-133-5

increases exponentially. In practical terms, this shows that the number of false alarms is highly sensitive to the level of the threshold and decreases exponentially as the threshold is set lower. This, in turn, means that 1) the threshold does not need to be unduly low to be effective and 2) it can be set to resist to some degree of fluctuations in traffic characteristics.

For all these reasons, the robustness of UHD is high.

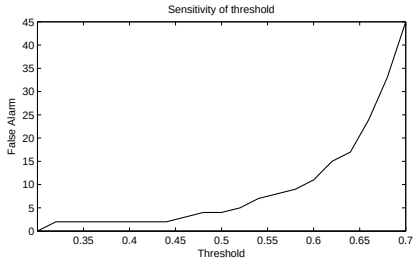

Fig. 1: Number of false alarms as function of threshold

C. Summary

In Sections IV-A and IV-B, we have applied a method for evaluating the performance of the detection mechanisms. For a comparison of the deployment cost of the two techniques, once we know the real value of the cost functions, we can evaluate DC.

The greater value of the DC metric reflects a better evaluation performance for the stateful technique. We can observe a shorter detection time interval for the UHD technique than for the FDS technique and we can conclude that, as the technique is stateful, detection is faster and more accurate. Also, the UHD stateful technique is robust and the FDS mechanism is not as strong as an attacker can evade the mechanism and it can produce false alarms during flash crowds. While stateless methods have varying degree of sensitivity to these issues, they are nevertheless exposed to them.

V. Conclusion and Discussion

From our comparison of two detection techniques and for other techniques, and following our assumption that each technique is representative of its genre, we have observed that stateless detection is slower and less reliable than a stateful detection technique. Also stateless techniques cannot respond to the detection as they do not store information and, as a consequence, cannot as effectively selectively stop the attack packets whereas stateful techniques store data, which can be used to react to the attack once it has been detected such as throttling or blocking attack traffic [16].

Stateful techniques demonstrate significant improvements in (1) the robustness of detection in the presence of detection evasion techniques (false negatives) or errors (false positives), (2) detection time, detection rate and the false alarm rate, and (3) the possibility of using the information collected by the technique to stop or control the attack.

Stateful techniques however use more memory and, to a lesser degree, more CPU cycles and have higher deployment cost: they would tend to require dedicated pieces of equipment whereas stateless techniques would more easily be embedded in routers.

Finally, we should remind the reader that these techniques can be viewed as complementary. While we have established the superiority of stateful techniques close to the edge of the server's network, stateless methods can be useful closer to the core of the network, where resources are scarce, but detection efficiency useful, albeit less critical.

Further research underway aims at better refining and defining this complementarity to extend our framework to hybrid models.

References

[1] D. Moore, G. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," in *USENIX Security Symposium*, August 2001.

[2] 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.

[3] S. Jin and D.S.Yeung, "A Covariance Analysis Model for DDoS Attack Detection," in *IEEE International Conference on Communications*, vol. 4, June 2004, pp. 1882 – 1886 Vol.4.

[4] Y. Ohsita, M. Murata, and S. Ata, "Detecting Distributed Denial-of-Service Attacks by Analyzing TCP SYN Packets Statistically," in *IEEE, GLOBECOM*, December 2004.

[5] V. A. Siris and F. Papagalou, "Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks," in *IEEE, GLOBECOM*, December 2004.

[6] S.-W. Shin, K.-Y. Kim, and J.-S. Jang, "D-SAT: Detecting SYN Flooding Attack by Two-Stage Statistical Approach," in *The Symp. on App. and the Internet, IEEE*, Janurary 2005, pp. 430–436.

[7] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN Flooding Attacks," in *Proceedings of IEEE INFOCOM*, vol. 3, June 2002, pp. 1530 – 1539.

[8] R. Mahajan, S. Bellovin, S. Floyd, J. Vern, and P. Scott, "Controlling High Bandwidth Aggregates in the Network," *SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 62–73, July 2002.

[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 the IEEE Symposium on Security and Privacy*, 1997, pp. 208–.

[10] 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, pp. 709 – 715 Vol. 1.

[11] T. M. Gil and M. Poletto, "MULTOPS: A Data Structure for Bandwidth Attack Detection," in *Usenix Security Symposium*, August 2001.

[12] J.Cheng, J. Yin, Y. Liu, Z. Cai, and M. Li, "DDoS Attack Detection Algorithm Using IP Address Features," in *Lecture Notes in Comp. Sc., Springer Verlag*, vol. 5598/2009, 2009.

[13] M. Bellaïche and J.-C. Grégoire, "SYN Flooding Attack Detection Based on Entropy Computing," in *IEEE GLOBECOM*, November 2009.

[14] ——, "Measuring Defense Systems Against Flooding Attacks," in *IWCMC 2008, International Wireless Communications and Mobile Computing Conference*, August 2008, pp. 600 –605.

[15] R. Chang, "Defending Against Flooding-based Distributed Denial-of-Service Attacks: a Tutorial," *Communications Magazine, IEEE*, vol. 40, no. 10, pp. 42–51, October 2002.

[16] T. Peng, C. Leckie, and R. Kotagiri, "Protection from Distributed Denial of Service Attack Using History-based IP Filtering," in *International Conference on Communications*. IEEE, June 2003, pp. 482 – 486 vol.1.

 ISBN: 978-1-61208-133-5