Enhancement of the Detection of the TCP SYN Flooding (DDoS) Attack

Assist.prof. Hamid M. Ali
Computer engineering dept.
University of Baghdad

Dr. Ibraheem K. Ibraheem
Computer engineering dept.
University of Baghdad

Sarah W.A. Ahmad
M.Sc. student
Computer engineering dept.

ABSTRACT

The major of DDoS attacks use TCP protocol and the TCP SYN flooding attack is the most common one among them. The SYN Cookie mechanism is used to defend against the TCP SYN flooding attack. It is an effective defense, but it has a disadvantage of high calculations and it doesn't differentiate spoofed packets from legitimate packets. Therefore, filtering the spoofed packet can effectively enhance the SYN Cookie activity. Hop Count Filtering (HCF) is another mechanism used at the server side to filter spoofed packets. This mechanism has a drawback of being not a perfect and final solution in defending against the TCP SYN flooding attack. An enhanced mechanism of Integrating and combining the SYN Cookie with Hop Count Filtering (HCF) mechanism is proposed to protect the server from TCP SYN flooding. The results show that the defense against SYN flood DDoS attack is enhanced, since the availability of legitimate packets is increased and the time of SYN Cookie activity is delayed.

Keywords: DDoS attack, TCP SYN flooding attack, spoofed packets, SYN Cookie mechanism, HCF mechanism

الخلاصة

معظم الـ(DDoS attacks) تستخدم الـ(TCP protocol) والـ(TCP SYN flooding attack) هو الاكثر شيوعا من بينهم. تستخدم طريقة الـ(SYN Cookie) للحماية من الـ(TCP SYN flood). الـ(SYN Cookie) طريقه دفاع فعالة ولكن نقطة ضعفها ان حساباتها تستغرق وقت طويل ولاتمييز الـ(Packets) المزورة عن الشرعية، لذا، تصفية الـ(Packets) المزورة ممكن ان يحسن فعالية الـ (SYN Cookie) بصورة فعالة. الـ(HCF) هي طريقة اخرى تستخدم عند جهة الـ(Server) لتصفية الـ(Packets) المزورة، ونقطة ضعفها انها لا تعتبر حل كامل ونهائي لحماية الـ(Server) من الـ(TCP SYN flooding attack). تم اقتراح طريقه مكونه من تكامل ودمج الـ(SYN Cookie) والـ(HCF)لحماية الـ(server) من الـ(TCP SYN flooding). حيث اظهرت النتائج تحسن في الدفاع ضد الـ(SYN flooding)، وذلك بزيادة عدد الـ(Packets) الشرعية اضافة الى تاخير وقت تفعيل طريقة الـ(SYN Cookie).

INTRODUCTION

Availability requires that computer systems function normally without loss of resources to legitimate users. One of the most challenging issues to availability is Denial-of-Service (DoS) attack. These attacks achieve their goal by sending at a victim a stream of packets that swamps his network or processing capacity denying access to regular clients.

Distributed Denial of Service (DDoS) attacks add the many-to-one dimension to the DoS problem, making the prevention and mitigation of such attacks more difficult and the impact proportionally severe **[D. Mohamed, 2010]**.

The major type of DDoS attacks is TCP SYN flooding attack, which aims to exhaust the server SYN queue with spoofed packets, in order to make the server denying any new connection.

The SYN Cookie is an effective mechanism that saves large number of legitimate connections from being dropped when the server is under attack. However, the SYN Cookie mechanism does not differentiate between legitimate and spoofed packets, and makes a large amount of calculations that unnecessarily consume the CPU time. Hence, the spoofed packets reserve locations in the server SYN queue before reaching the SYN queue limit, leading to early activation of SYN Cookie **[J. Kurose et. al., 2008]**.

HCF is a countermeasure for filtering spoofed packets, but it is not a perfect and final solution in defending against the TCP SYN flooding attack **[C. Jin et. al., 2003]**. Therefore, a combination of SYN Cookie with Hop Count Filtering (HCF) is proposed to enhance the detection of TCP SYN flood attack.

The rest of the paper is organized as follows. Section II surveys previous research related to TCP SYN Flood attacks and their detection mechanisms. The proposed enhancement scheme is explained in section III. Analysis of the enhanced scheme is demonstrated in Section IV. Evaluation of the enhancement scheme is done in Section VI. Section VII concludes the paper and gives further work.

BACKGROUND

A. Ip Address Spoofing

Packets sent using the IP protocol include the IP address of the sending host. The recipient directs replies to the sender using this source address. However, the correctness of this address is not verified by the protocol. The IP protocol specifies no method for validating the authenticity of the packet's source. This implies that an attacker could forge the source address to be any he desires. Sending IP packets with forged source addresses is known as packet spoofing and is used by attackers for several purposes. These include obscuring the true source of the attack, implicating another site as the attack origin, pretending to be a trusted host, hijacking or intercepting network traffic, or causing replies to target another system.

IP spoofing has often been exploited by DDoS attack, and it has become a common feature of the many DDOS attack tools **[V. Praveena et. al., 2008]**. Because none of these are desirable, it is useful to determine if a packet has a spoofed source address.

The IP spoofing plays an important role in network attacks, in particular the DDoS attack, for a number of reasons [**D. Mohamed, 2010**, Z. Duany et. al., 2006]:

I. The IP spoofing makes it hard to distinguish attack packets with spoofed source addresses from legitimate packets.
II. The IP spoofing makes the detection of the flooding source very difficult, since it completely hides the IP address of the flooding source.
III. The common types of DDoS attack, such as the TCP SYN flooding attack and amplification attack, are not possible without the IP spoofing **[D. Mohamed, 2010, Q. Gu, et. al., 2007]**.

B. TCP SYN Flood Attacks

The Transmission Control Protocol (TCP) includes a full handshake between sender and receiver, before data packets are sent. The initiating system sends a SYN request shown in Figure 1 a. The receiving system sends an ACK (acknowledgement) with its own SYN request. The sending system then sends back its own

ACK and communication can begin between the two systems. If the receiving system is sent a SYN packet but does not receive an ACK to the SYN it sends back to the sender, the receiver will resend a new ACK + SYN after some time has passed shown in Figure 1 b. The processor and memory resources at the receiving system are reserved for this TCP SYN request until a timeout occurs.

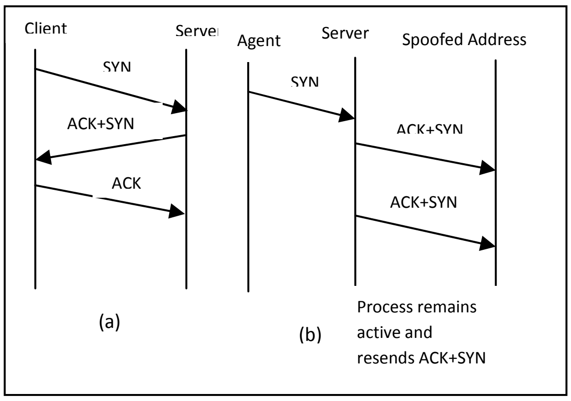

Figure 1: (a) TCP Synchronization (b) TCP-SYN attack

In a DDoS TCP SYN attack, the attacker instructs the zombies to send such bogus TCP SYN requests to a victim server in order to tie up the server's resources, and hence prevent the server from responding to legitimate requests.

The TCP SYN attack exploits the three-way handshake between the sending system and the receiving system by sending large volumes of TCP SYN packets to the victim system with spoofed source IP addresses, so the victim system responds to a non-requesting system with the ACK+SYN. When a large volume of SYN requests are being processed by a server and none of the ACK+SYN responses are returned, the server begins to run out of processor and memory resources. Eventually, if the volume of TCP SYN attack requests is large and they continue over time, the victim system will run out of resources and be unable to respond to any legitimate users **[B. Swain, 2009]**. Over 90% of DDoS attacks use TCP protocol and the TCP SYN flooding attack is the most common one among them **[D. Mohamed, 2010]**.

C. Hop Count Filtering (HCF)

HCF is a stand-alone scheme that works at the victim-end to defend against the DDoS by detecting and filtering spoofed packets. HCF mechanism does not require any network support.

Although an attacker can forge any field in the IP header, he cannot falsify the number of hops an IP packet takes to reach its destination, which is solely determined by the Internet routing infrastructure.

The hop count information is indirectly reflected in the TTL field of the IP header, since each intermediate router decrements the TTL value by one before forwarding it to the next hop. The difference between the initial TTL (at the source) and the final TTL value (at the destination) is the hop count between the source and the destination. By examining the TTL field of each arriving packet, the destination can infer its initial TTL value, and hence the hop counts from the source.

The rationale behind hop count filtering is that most spoofed IP packets, when arriving at victims, do not carry hop count values that are consistent with legitimate IP packets from the sources that have been spoofed.

Hop Count Filtering (HCF) builds an accurate HCI (IP to hop count) mapping table, while using a moderate amount of storage, by clustering address prefixes, based on hop count.

Two running states, alert and action, within HCF use this mapping to inspect the IP header of each IP packet. Under normal condition, HCF resides in alert state, watching for abnormal TTL behaviors without discarding packets. Upon detection of an attack, HCF switches to action state, in which the HCF discards those IP packets with mismatching hop counts. HCF can recognize close to 90% of spoofed IP packets **[C. Jin et. al., 2003**, **B. Swain, 2009**, **Mohan, 2010]**.

D. SYN Cookie

There is an effective defense, called SYN Cookie, now deployed in most major operating systems. In normal operation, a client sends a SYN packet and the server responds with a SYN/ACK packet, the server will then hold state information in the SYN queue while waiting for client ACK packet.

A simple SYN flood (using suitable software) will generate SYN packets which would consume all available SYN queue memory as the server must maintain state for all half-open connections. And since this SYN queue is finite the server will no longer accept new TCP connections and thus fail or deny service to the user. This is highly leveraged attack since a very small amount of bandwidth and CPU can exhaust the resources on a large number of servers.

SYN Cookies work as follows:

- When the server receives a SYN packet, it doesn't know if the packet is coming from a legitimate user or is part of SYN flood attack. So the server doesn't create a half-open connection for this SYN. Instead, the server creates an initial TCP sequence number that it is a complex function (hash function) of source and destination IP address and port numbers of the SYN packet, as well as a secret number only known to the server. (The server uses the same secret number for a large number of connections.) This carefully crafted initial sequence number is the so-called "Cookie". The server then sends a SYN/ACK packet with this special initial sequence number. Importantly, the server doesn't remember the Cookie or any other state information corresponding to the SYN.
- If the client is legitimate, then it will return an ACK packet. The server, up on receiving this ACK, needs to verify that the ACK corresponds to some SYN sent earlier. It is done with the Cookie. Specifically, for legitimate ACK, the value in the ACK field is equal to the sequence number in the SYN/ACK+1. The server will then run the same function using the same fields in the ACK segment and the secret number. If the result of the function plus one is the same as ACK number, the server concludes that the ACK corresponds to an earlier SYN segment and is hence valid. The server then creates a fully open connection along with a socket.
- On the other hand, if the client doesn't return an ACK packet, then the original SYN has done no harm at the server, since the server hasn't allocated any resources to it. SYN Cookies effectively eliminate the threat of a SYN flood attack **[J. Kurose, 2008]**.

The use of SYN Cookies allows a server to avoid dropping connections when the server SYN queue fills up. And does not break any protocol specifications, and therefore should be compatible with all TCP implementations. There are, however, as observed some overhead in the calculations, but these overhead need only apply when the server is under attack, and the connection would have otherwise been denied.

THE COMBINATION SCHEME

It is desired to combine the effect of the SYN Cookie with the hop count filtering mechanism to reduce both the amount of spoofed packets that reserve locations in the server SYN queue, and the undesired SYN Cookies calculations that happen when the SYN Cookie is used alone. The proposed idea works as follows:

The HCF mechanism is activated from the beginning to detect and filter the spoofed packets and prevent them from reserving locations in the server SYN queue. Consequently the number of spoofed packets in the SYN queue is reduced. Now while the HCF is activated and there is a flux of spoofed packets intervening legitimate packets, the server SYN queue will be filled up and starts denying new connections. To save the connection from being denied, the SYN Cookie mechanism is activated when server SYN queue is reached to a predetermined limit. In other words, the HCF followed by SYN Cookie mechanisms (after reaching the SYN queue limit) is adopted to save legitimate clients. Hence the number of calculated SYN cookie, after the SYN Cookie activation, is less than using SYN Cookie alone, because the SYN Cookie is applied mostly to legitimate clients, where most of the spoofed packets are filtered by HCF.

ANALYSIS OF THECOMBINATION SCHEME

To illustrate the performance of the proposed enhancement, this section discusses the protection of the server in four cases: case1 server without defense mechanism, case2 server with SYN Cookie defense mechanism, case3 server with Hop count filtering defense mechanism, and case4 server with combination of SYN Cookie and HCF mechanisms. For simplicity, we assume the length of the server SYN queue, which is used to store client connections' information, is 16 locations; each location of the server SYN queue is dedicated for one client, as illustrated in Figure 2.

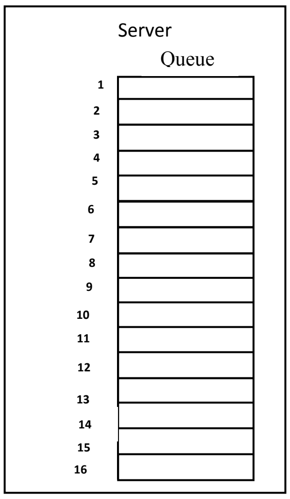

Figure 2: the sever SYN queue

In this discussion, it is assumed that in every unit of time (for example a second) either one legitimate TCP SYN packet or five attacker TCP SYN packets arrive to the server and reserve locations in the server SYN queue. Obviously, the purpose of the five TCP SYN Attack packets (arrive in one second) is to flood and exhausts the SYN queue. Also we will take a sample, for all the cases mentioned above, in that the TCP SYN packets arriving to the server are in the sequence of two legitimate clients followed by two attackers (ten packets), followed by two legitimate clients and so on. According to these assumptions we will test the performance of each case.

Case1: Server with No Defense Mechanism

In this case, the problem of the SYN flood DDoS attack is clearly illustrated, where the server SYN queue is quickly exhausted by huge number of spoofed packets that reserve locations in the SYN queue, causing the legitimate clients' packets to be dropped quickly by the server as shown in Figure 4. Now the spoofed packet's ratio in the server SYN queue is 0.75, while only 0.25 of the queue is for the legitimate clients, and end of the server SYN queue is reached quickly. This is the purpose of the SYN flood DDoS attack that must be detected and prevented.

As shown in Figure 3 the first and second clients (that sends 1 packet/second) reserve the first two locations in the server SYN queue, then the first and second attackers (that sends 5 packet/second) reserve ten locations in the server SYN queue. After that two clients send two packets and reserve two locations in the SYN queue. Now the SYN queue is filled and began to drop the new connections, when the SYN queue capacity exceeded.

In this case, the time required to full up the server SYN queue is 6.4 sec. two seconds for the first two legitimate clients and two seconds for the two attackers (each with 5 packets) and two seconds for the two legitimate clients and 0.4 seconds for the last two attacker's packets. At this point, the server SYN queue contains few legitimate packets (4 packets), while the rest of the server SYN queue is filled with attacker packets (12 packets). Thus the system must filter the packets and allow only the legitimate packets reserve locations in the server SYN queue.

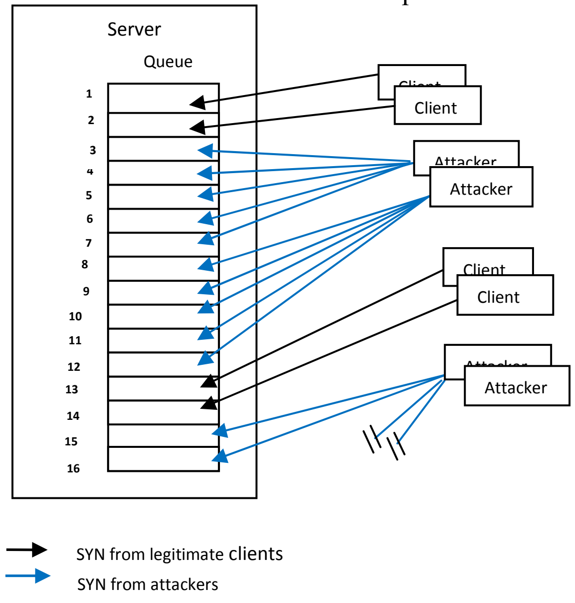

Figure 3: The server without any defense mechanism.

Case2: Server with SYN Cookie Defense Mechanism

The SYN Cookie is a practical defense mechanism. For instance, servers' operating systems enable SYN cookies only when a certain limit (of the server SYN queue) is exceeded, allowing the servers to normally operate without the disadvantages of SYN cookies computation time (if it is activated all the times) due to its encryption function. But the SYN Cookie also allowing the servers to be strongly protected when an attack occurs.

The SYN Cookie, is activated when the server SYN queue is allocated and reaches certain limit, say location 11 (the red location in Figure 4). At this limit the operating system starts to use SYN Cookie defense mechanism.

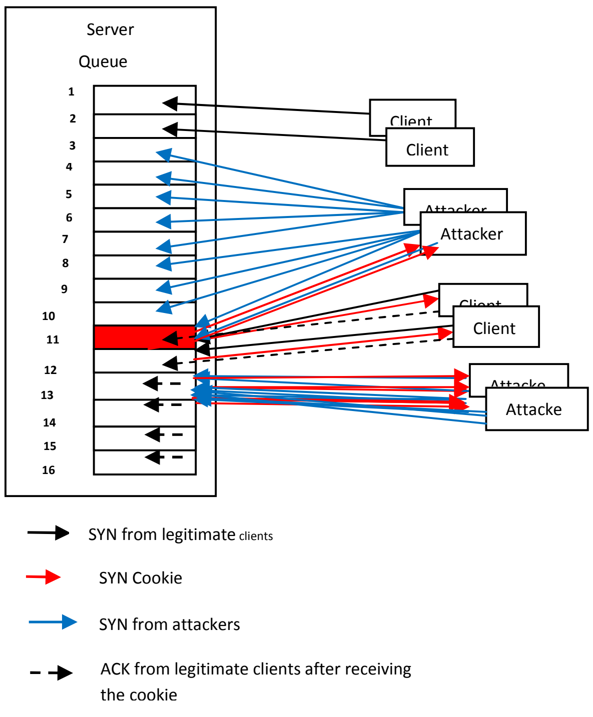

Figure 4: the server when the SYN Cookie is used alone

As shown in Figure 4, the first and second clients (that sends 1packet/second) reserve the first two locations in the server SYN queue (during 2 seconds), then the first and second attackers (that sends 5packet/second) reserve eight locations in the server SYN queue (during 1.6 seconds). The 9th and 10th packets are reached when the SYN Cookie is activated, Therefore, the SYN Cookie is applied to these two packets, this mean that SYN Cookie is applied to two spoofed packets (this is unnecessary overhead). Then two clients send two packets and two cookies are calculated to them (without reserving locations in the server SYN queue), then these two legitimate clients respond with ACK packet to finalize the 3-way handshaking. This means that the locations after the SYN queue limit are continuously in emptying state. Depending on the speed of the client's response and the CPU speed in calculating the SYN Cookie, the connections are saved; therefore, it is required to eliminate the number of spoofed packets that cause additional unnecessary CPU calculations of SYN Cookie. In this case, the time required to reach the SYN queue limit is 3.6 sec.

In summary the SYN Cookie defense mechanism has the following drawbacks:

- The attackers take more chances to reserve locations in the SYN queue before the SYN Cookie is activated, leading to reach the SYN queue limit (the red location in Figure 5) quickly. Therefore, the SYN Cookie, in this case, is activated earlier than cases 3 and 4 (described later), which contain filtering mechanism to drop spoofed packets before they can reserve location in the SYN queue.
- The SYN Cookie, after reaching the server SYN queue limit, is calculated to large number of packets. Most of them are spoofed packets. This is unnecessary calculation and time consuming overhead while the consumed time overhead can be used by the server to process more legitimate clients.

But SYN Cookie has advantage that it saves legitimate connections from being dropped by the server after reaching the server SYN queue limit; since the server sends SYN Cookies to the SYN packets' senders, without reserving locations for them in the server SYN queue.

Now there are 10 packets reserve locations in the server SYN queue before reaching the SYN queue limit, two of them are legitimate while the others are spoofed. While after reaching the server SYN queue limit all the packets are legitimate, hence, the minimum number of the legitimate packets here is 8, and the SYN Cookie after the SYN queue limit is applied to large numbers of spoofed packets. Therefore, using a filtering mechanism (like HCF mechanism) can enhance the defense against TCP SYN flood DDoS attack that uses the SYN Cookie mechanism.

Case3: Server with HCF Defense Mechanism

The HCF detects and filters spoofed packets using a table of IP addresses with corresponding hop counts. As mentioned before the HCF mechanism can detect 90% of spoofed packets, this means that for the two attackers that send 10 packets (in 2 seconds) 1 packet can reserve location in the server SYN queue in 2 seconds.

As illustrated in Figure 5, the new connections are denied after 21 sec (1 second for each legitimate packet and 2 second for the two attackers to reserve one location). And the number of spoofed packets (the blue arrows) is significantly reduced compared to the two previous cases (case1 and 2); since the filtering mechanism filters most of the spoofed packets (the green lines in Figure 5) and reduces their chances to reserve locations in the server SYN queue. So the number of spoofed packets that reserve locations in the server SYN queue is 5 packets while the number of legitimate packets is 11 packets.

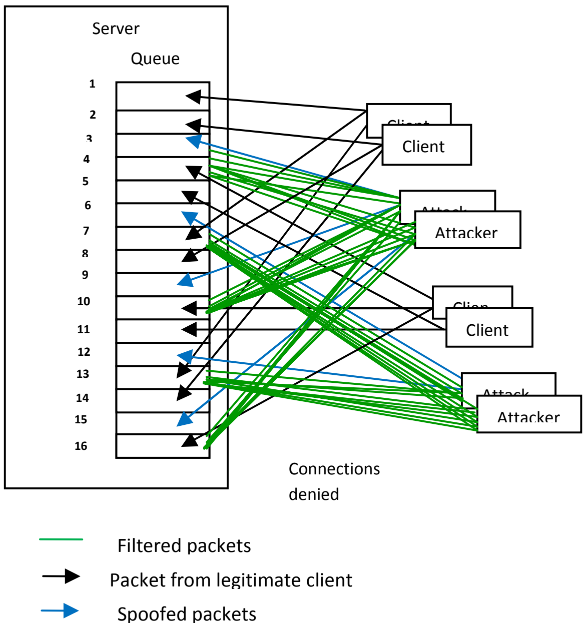

Figure 5: the server when using the HCF mechanism alone

However, this mechanism has a drawback that when the server's SYN queue is filled, the new connections are denied, while these connections are saved when using the SYN Cookie mechanism. Combining this mechanism with the SYN Cookie can help in saving the legitimate connections after a certain limit from being dropped, in addition to the benefits of the HCF filtering mechanism which has low processing time in accessing a look up table to detect the spoofed packet.

Case4: Server with Combined SYN Cookie and HCF (SYN Cookie/HCF)

As shown in Figure 6 two legitimate clients reserve locations in the server SYN queue, then two attackers send ten packets (during 2 seconds) and because the HCF is working, only one packet can reserve location in the server SYN queue.

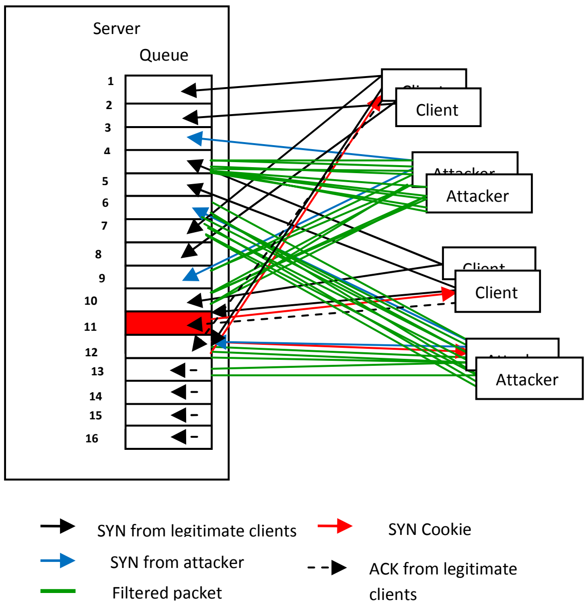

Figure 6: the server with combined SYN Cookie and HCF mechanisms

Figure 6 also shows that most of the packets, before reaching the server SYN queue limit (the red location), are legitimate packets. This is because most of the spoofed packets are filtered (the green lines) by the HCF mechanism; therefore, the server SYN queue limit is reached after 13 sec, later than case2 which uses only the SYN Cookie. Now, the number of legitimate packets, before reaching the server SYN queue limit, is 7 packets while the spoofed packets are 3 packets. Here, the minimum number of the legitimate packets is 13 packets (because the SYN Cookie activity guarantees that all the packets after the server SYN queue limit are legitimate packets).

After reaching the server SYN queue limit, it is obviously that the number of calculated SYN Cookie in SYN Cookie/HCF is significantly less than using SYN Cookie alone (because of the HCF activity that filters the spoofed packets). And the benefit of this mechanism is that the SYN Cookie prevents the spoofed packets from reserving locations when it is activated. Hence, the SYN Cookie/HCF is better than HCF alone in saving the new connections from being denied.

The reason of using HCF in detecting spoofed packets instead of activating the SYN Cookie from the beginning is due to the time-costing calculations in the SYN Cookie mechanism. Where the time required to process HCF, which is a look up table of IP addresses, is negligible compared to SYN Cookie processing time which requires calculations of encryption functions.

EVALUATION

We can see from Table 1 that the proposed SYN /HCF has the largest number of legitimate packets.

Table 1: the summary of the concluded performance of the four cases

	Without any defense	With SYN Cookie defense	With HCF defense	With SYN Cookie/HCF
Number of legitimate packets	4	8	11	13
Availability of Legitimate Clients	0.25	0.5	0.6875	0.8125

The two following tables, 2 and 3, compare the combined SYN Cookie/HCF mechanism with SYN Cookie and HCF respectively.

From Table 2, it is observed how the proposed SYN Cookie/HCF mechanism enhances the performance, and increase the defense against the SYN flood DDoS attack, since the time of the SYN Cookie activation is later than the case of SYN Cookie alone and the ratio of the spoofed packets that can reserve locations in the server SYN queue is minimized.

Table 2: comparison of the SYN Cookie mechanism with combined mechanism

	SYN Cookie	SYN Cookie/HCF
Time of reaching the server SYN queue limit	3.6	13
Spoofed packet ratio in the server SYN queue	0.5	0.1875

Table 3: comparison of the HCF mechanism with combined mechanism

	HCF	SYN Cookie/HCF
Number of legitimate packets	11	13
Spoofed packet ratio in the server SYN queue	0.3125	0.1875

Also from Table 3 it is clear that the number of the legitimate packets in the proposed SYN Cookie/HCF mechanism is higher than the case of HCF alone and the ratio of the spoofed packets that can reserve locations in the server SYN queue is minimized.

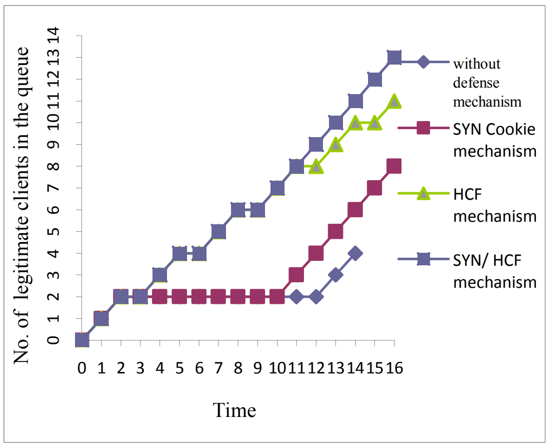

Figure 7: number of legitimate packets in the server queue for the four cases

Figure 7, shows four curves of the four cases. All the curves start with two legitimate clients in the first two seconds. The blue curve represents the server without applying any defense mechanism. It can be observed that the number of legitimate clients in the queue remains two until the second 13 and 14 then the queue is filled and any new connections will be denied. The red curve represents the number of legitimate clients when applying the SYN Cookie mechanism. The effect of this mechanism appeared at second 11, when the SYN Cookie is activated. From this point the number of legitimate clients is increased by one every second, since the SYN Cookie dedicate the queue for legitimate clients only after its activation. The green curve represents the number of legitimate clients when applying HCF mechanism. In this curve as observed the number of legitimate clients in the queue is in the sequence of two clients every three seconds. Finally, the purple curve represents the combination mechanism (HCF/SYN), where the difference is appeared at second 11, as the SYN Cookie mechanism is activated. From this point the number of legitimate clients in the queue is increased by one every second.

CONCLUSION

SYN Cookie mechanism does not differentiate between legitimate and spoofed packets, and makes a large amount of calculations that unnecessarily consume the CPU time. This wasted CPU time can be used by the server to process extra SYN packets. Where, the spoofed packets reserve locations in the server SYN queue before reaching the SYN queue limit, leading to early activation of SYN Cookie.

HCF is a countermeasure for filtering spoofed packets, but it is not a perfect and final solution in defending against the TCP SYN flooding attack.

From the proposed enhancement we conclude that: filtering the packets, using HCF, before computing the SYN Cookies for them enhances the overall system performance. Since the HCF reduces the probability of the spoofed packets to get a chance to reserve locations in the SYN queue. Clearly, it takes more time to reach the SYN queue limit as a result of dropping spoofed packets by HCF. In addition, when reaching the SYN queue limit the SYN Cookie is applied to less number of packets.

To increase the robustness of the defense against the TCP SYN flooding attack, it is mandatory and preferable to apply certain defense mechanism for preventing spoofed packets at the source network, where the spoofed packets are generated.

REFERENCES

B. R. Swain, “Mitigation of DDoS Attack using a Probabilistic Approach & End System based Strategy”, M.Sc. thesis, at National Institute of Technology, India, May 2009.

C. Jin, H. Wang, K. G. Shin “Hop-Count Filtering: An Effective Defense against Spoofed DDoS Traffic”, ACM, Washington, USA, October 27–30, 2003.

D. Mohamed, “Defense against Distributed Denial of Service Attacks in Computer Networks”, Ph.D thesis, at Graduate School of Information Sciences Tohoku University, March 2010.

J. F. Kurose, K. W. Ross, “Computer Networking a Top-Down Approach”, Fourth Edition, Pearson Education international, 2008.

Mohan H.S, A R. Reddy, “An Effective Defense against Distributed Denial of Service in Grid”, First International Conference on Integrated Intelligent Computing IEEE 2010.

Q. Gu, P. Liu, “Denial of Service Attacks”, June 2007, http://s2.ist.psu.edu/paper/DDoS-Chap-Gu-June-07.pdf .

V. Praveena, N. Kiruthika, “New Mitigating Technique to Overcome DDOS Attack”, World Academy of Science, Engineering and Technology 45 2008.

Z. Duany, X. Yuan, J. Chandrashekar, “Controlling IP Spoofng based DDoS Attacks Through Inter-Domain Packet Filters”, Proc. IEEE INFOCOM 2006.