International Journal of Computer Science and Information Security (IJCSIS),
Vol. 15, No. 1, January 2017

Intrusion Detection System for SYN Flood Attack: Methods and Implementation

Abdulaziz Almehmadi
University of Tabuk
Tabuk, Saudi Arabia
aalmehmadi@ut.edu.sa

Abstract—**Since Distributed Denial of Service (DDoS) attacks are difficult to be detected as distinguishing if packets are malicious or normal is challenging, new methods of detection are proposed. The suggested methods are based on analyzing the traffic and monitoring its results on servers. It is also based on the packets behavior and attempts. In this paper, new methods of detecting SYN flood is discussed and given. Several methods have been proposed to detect SYN flood attack; however, none of them is accurate enough. One of the detection mechanisms that we propose involves looking at the Backlog queue since it is the main cause why new connections are denied. Another way of detection is by counting the number of suspected SYN packets on traffic, and whenever it exceeds a specific number, an alarm is triggered showing that there is a potential of SYN Flood attack. Methods of detecting such attack should be resolved automatically when it happens because of the direct impact this attack causes.**

Keywords- Intrusion Detection System, DDoS, SYN Flood.

1. Introduction

INTERNET connection has become an essential part of our daily life, and it is important to a huge number of companies and organizations. From that point, the security of networks has become a very important issue due to the risks these networks face during their connection to the internet. Many people are using the internet on a daily basis; thousands of them are trying to intrude on major networks all the time. Those intruders are trying to damage network resources and/or bring a network down. To defeat them, the Intrusion Detection System (IDS) was born, in order to discover any attempt to intrude on a network by using different methods and algorithms based on the kind of the attack.

Intrusion Detection System is either hardware or software that analyzes the interior traffic and detect if it is normal or malicious traffic. If the IDS detects malicious traffic, it triggers an alarm that there is an attack for a prevention method to take place. In this paper, we focus on attacks related to distributed denial of service.

Denial of service attack (DoS) is a way to attack networks by flooding its resources by sending millions of packets in a short interval. These packets are not to be processed quickly, so a huge number of them eventually exceeds the capacity of the servers. As a result, the attack causes slow services and lack of access to that server. There are typical DoS attacks such as Fraggle, SYN Flood, Ping flood, and Ping of death.

Another form of DoS is Distributed Denial of Service attacks (DDoS); however, DDoS it is done through a number of machines rather than just using one. These different machines are often called "Zombies" because attackers are using them to perform the attack with or without their owners' approval. The examples of DDoS attacks are SYN Flood, Smurf, Trinoo, TFN, TFN2K, Trinity, and Mastream attacks.

Due to the impact DoS attacks cause, both on the services a company provide, the resources and reputation, there exists a need to a method of detection that can handle the millions of SYN flood packets and prevention them from causing the any service disruption.

The rest of this paper is organized as follow. In section 2, we take a look at TCP Packet's header explaining the most important parts that we benefit from in this paper followed by section 3 which provides a general overview on the SYN flood attack. In section 4, we discuss the methods of these attacks along with their effects. Related work section follows in section 5. Then we propose our own solution along with the methods of detection in section 6. The next section is the implementation followed by section 8 that shows our results from the experiment that was conducted. Finally, the conclusion and future work are given at the end of this paper.

2. TCP

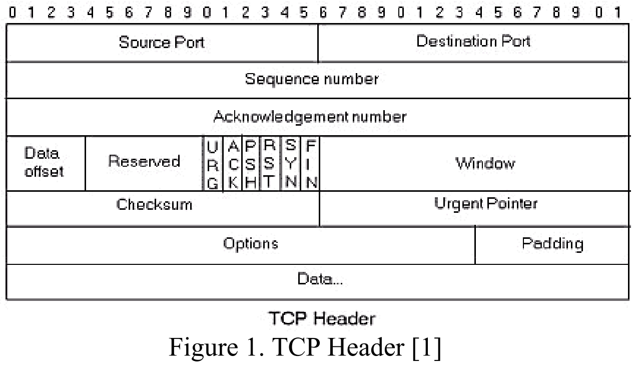

Figure 1. TCP Header [1]

Figure 1 shows the TCP header [RFC 793]. The main three parts that are used in the proposed method are:

- Sequence Number.
- Acknowledgement Number.
- Flag section.

Sequence Number:

In this paper, it is beneficial to know the sequence number of each packet. These numbers are important to be known for intrusion detection system to be effective. We store the sequence number for each packet for later use.

Acknowledgement Number:

From the Acknowledgement Number, we investigate if a client has sent the last ACK in the three-way handshake to establish a connection or not. If it is not received, we consider it as a black-listed IP for further investigations. If it is received, we consider it safe.

Flag section:

Flag section is the most important part we focus on when detecting SYN flood attack. If we are unaware of the kind of packet we receive, we will not be able to determine its purpose. Basically, we use this part to identify SYN, ACK, FIN and RST packets from the rest of the packet.

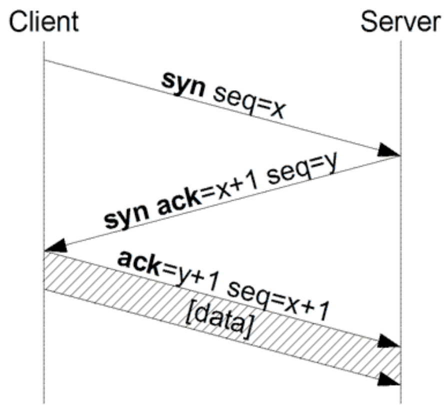

Figure 2. TCP Three-way handshake [1]

3. SYN Flood:

During the last five years of 1990s, the world has known a new method of Denial of Service attack. The first attack targeted an ISP's mail servers and caused well-publicized outages. The attack when compare with other DoS attacks show severe consequences. Users were not able to connect to the targeted server as the effected system could not handle all other requests. The attack is known as a SYN attack or SYN flood attack.

3.1 Definition of SYN Flood:

When wanting to access a particular server, there has to be the TCP 3 way handshake between source and destination. The following is the process that is shown in Figure 2:

1-The client requests a connection by sending a SYN (*SYNchronize*) packet to the server.
2-The server *acknowledges* this request by sending SYN-ACK packet back to the client.
3-The client responds with an ACK packet, and the connection is established [1].

When a client sends the first packet (SYN) and receives SYN-ACK without sending the ACK packet to establish a connection as shown in Figures 3 or sending a SYN packet with spoofed IP as shown in Figure 4, destination server waits for a specific amount of time before eliminating the SYN packet from its memory. This vulnerability of TCP 3 way handshake has been exploited by sending millions of SYN packets in a specific interval so the memory that keeps SYN packets becomes full. Therefore, the victim server becomes unreachable and goes into BLOCK mode. Moreover, each further request is discarded due to the lack of memory that the flooding caused. SYN flood attack causes what is known as Denial of Service or DoS attack.

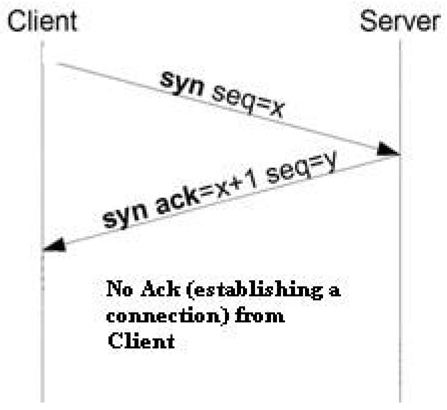

Figure 3. SYN Flood Attack with no ACK

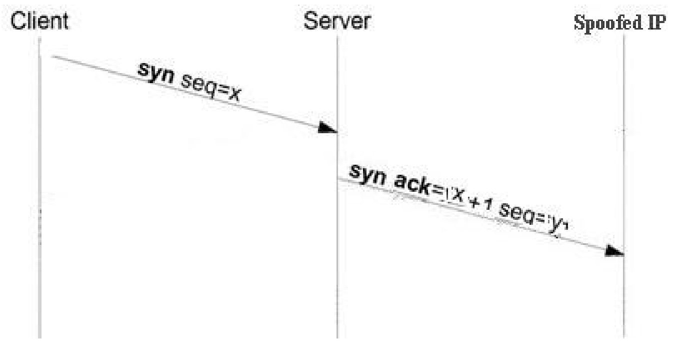

Figure 4. SYN Flood Attack with spoofed IP

4. SYN Flood Attack Methods vs. IDS:

There are many and different methods to perform SYN Flood attacks; most of them started simple. By the time, however, their vulnerabilities have been patched to become powerful. Whenever attacks methods are detected, attackers find new ways to trick the detection systems. Therefore, SYN Flood attacks methods are becoming much powerful and difficult to detect. One of the attacks methods that has been patched from being detected from IDSs is the detection based on source addresses. The attack methods were patched by spoofing the source IP address randomly in each SYN packet. Thus, the detecting methods based on the source addresses are not detected anymore. Another way to not being detected is by applying the DDoS, which involves a number of Zombies to perform an attack rather than just using the attacker's machine. Some machines are not powerful enough to perform such attack, so by using powerful ones, attackers are able to send a huge number of SYN packets in a small period of time. As a way of defense from such methods, the idea of increasing the Backlog queues was provided. However, the idea of increasing Backlog queue is not effective enough due to the maximum size of it. Moreover, attackers might exploit a specific vulnerability in TCP packets by rather than sending a huge number of SYN packets, they send only one SYN packet that its size force the Backlog queue to discard any further requests to establish a connection. Each time security specialists detect SYN Flood attacks, attackers create new ways to hide it. One of the major problems in SYN Flood attack is that these malicious packets are hard to be filtered or ignored because they are important in any normal network traffic.

Effects:

SYN Flood attack does not damage machines or cause loses of information. It causes a server to not being able to establish new connections. Therefore, it does not cause, for example, a company to lose important information; however, it affects its reputation and that affects trust in the company. When dealing with the issue of causing a server not to establish new connections even for few minutes, we deal with companies' loss of millions of dollars.

5. Related work:

There are several SYN flood detection mechanisms which are based on the relationships between the TCP control packets during the TCP three-way handshake process. We discuss a few of such methods in this section.

SYN cookies method is a technique proposed by Daniel J Bernstein [8] to protect against the SYN flood attacks. In this technique the idea is not to store any information about the incomplete connection in the queue when the first SYN packet is received, instead a special sequence number is sent with SYN-ACK packet's TCP header. The initial sequence number is a combination of current time, maximum segment size value specifying how many bytes of data can be encoded in a packet and cryptographic hash function computed with the server and client IP addresses, server and client port numbers, time and a secret value known to the server. The description is shown in Figure 5.

T	MSS	H

T= counter incremented every minute
MSS= maximum segment size
H= hash (source and destination IP address, source and destination port, T, secret)

Figure 5. SYN Cookies [8]

Once the server sends the SYN-ACK packet the client sends back the ACK packet with the sequence number one greater than the initial sequence number. Then the server subtracts one from the sequence number of the ACK packet to determine if it is a valid SYN cookie or not. Then it computes the SYN cookie again based on the MSS and time value present in the received packet. If the computed and received SYN cookie match, the packet is considered valid or else it is discarded.

Limitations of this method are the computational overhead required to process incoming SYN and ACK packets and no retransmission of unacknowledged SYN-ACK packets. Also parameters such as size of the receiving window and MSS which helps in efficient communication are lost.

The other method which was proposed by Wang et al [3] involves counting the number of incoming SYN and FIN/RST packets in each detection period and no parametric Cumulative Sum based method to detect the change of the normalized difference between SYN and FIN/RST packets. The simplicity of the method was its statelessness and low computation overhead. Instead of monitoring the packets at front end (like proxy or firewall) or the victim's server itself, Wang et al detected the SYN flood attacks at the leaf routers which connect the end hosts to the internet. Benefit deploying the FDS near the first mile leaf router is the proximity to the flooding sources and which saves most of the time of IP trace back.

Sun *et al.* [4] proposed a robust technique to detect the SYN flooding attacks. Under the normal circumstances, there is a one to one match between valid SYN packets and valid FIN packets. They made use of counting Bloom Filter [4] to store the *4-tuple* (Source and Destination IP, Source and Destination Port) of counted SYN packets to determine the valid FIN or RST packets for the corresponding SYN packet. A FIN or RST packet is considered valid only if its 4-tuple is in the Bloom Filter. This scheme is deployed at the edge routers of ISP and only one direction of traffic need to be monitored.

D-SAT is an algorithm to detect SYN flood attack in two stages. It is proposed by Seung-won et al [7] and uses two statistical methods to detect the attack. In the first step, the whole traffic is captured and only SYN packets, which has the SYN flag is set, are stored. Then the number of the SYN packets is compared to the all TCP packets. In the normal state, it is impossible to have a number of SYN packets larger than TCP packets. Therefore, if the ratio of SYN packets is increased by a huge number then they move to the second stage. The second stage is basically a comparison between different servers in the network. If SYN packets have been increased in other servers as well, then this is considered a normal state. Otherwise, it is considered a SYN flood attack. Their experiments found that the attack is detected in a short time. Although D-SAT looks like our method as our method analyzes the traffic and it has two stages, our algorithm is more efficient. It is efficient because we do not store all TCP packets and compare them to SYN packets. We reduce the processing load by storing only packets that make us distinguish between malicious SYN packets and normal SYN packets. As a challenge to their approach let us take this scenario "A user is downloading a huge file." How SYN packets are to be compared with the all TCP packets?

6. Solution:

In this section, we propose different methods to detect SYN flood attack. These methods are the result of our research and studying of the SYN packets' behavior and characteristics. We implement our methods in order to achieve better approach of detecting SYN flood attack with no false positives. This section is organized as follow. In part 5.1, our methods are provided. In part 5.2, our methods are discussed.

6.1 Proposed Methods:

We examined the effects of SYN flood attack and the way it is performed. Also, we studied SYN packets' characteristics along with the TCP protocol. As a result, we propose two new methods to detect the SYN flood attack.

The first method is called the Smart-Counting and it is a way of counting only SYN and ACK packets that are the first packets in the three-way handshake. These packets have characteristics that cannot be found in any of the TCP packets. Based on these characteristics, we determine if the captured packets are SYN packets or not. We used Wireshark library to capture all of the packets. Then we applied a number of filters to only display SYN packets. These filters are as follow: the first filter is (tcp.flags.SYN == 1) which only shows TCP packets that their SYN flag is set, but that is not enough, so we needed another filter which is (ip.dst == victim's IP address). This filter shows us only packets that are sent to the victim server. To detect SYN flood attack, we expect a number of SYN packets sent to the victim server. The third filter is (tcp.options.mss_val > 1000). In TCP protocol there is not any packet that has a value of Maximum Segment Size (MSS) set other than the first two packets in the TCP three-way handshake [2]. The MSS is important to set the maximum IP datagram size without using fragmentation. The fourth and last filter is (tcp.flags.ack == 0) which shows only packets that has the ACK flag not set. By applying all of these filters together, we essentially have only the first SYN packets in the three-way handshake, and then we store these packets for further processing. The entire filter will be as follows:

(tcp.flags.SYN == 1) && (ip.dst == victim's IP address) && (tcp.options.mss_val > 1000) && (tcp.flags.ack == 0)

In the following Table 1, we summarize the filters with their description.

Table 1. Filters for SYN and Description.

Filter	Description
(tcp.flags.SYN == 1)	SYN flag is set (exist in the packet)
(ip.dst ==Victim's IP address)	The destination IP address is the server's IP address that is being attacked.
(tcp.options.mss_val> 1000).	MSS exist.
(tcp.flags.ack == 0)	ACK flag is not set (Does not exist in the packet)

Storing packets in files and counting them is not a valid method to detect SYN flood attack. Therefore, we use a timer for each file, so in a specific amount of time, we have a new file that has a number of SYN packets. As a result, we get the number of SYN packets in a specific amount of time. This method monitors the traffic and detect if the rate of SYN packets to establish a new connection is normal or abnormal.

ACK packets should be considered in counting because they are the packets that determine if the connection has been established or not. Therefore, they should be counted as well. The same technique for counting SYN packets applies with changing the filters. The filters that guarantee displaying only ACK packets are as follow. The filter (tcp.flags.SYN == 0) which shows that any TCP packet that has the SYN flag not set are going to be shown. Also, (tcp.flags.ack == 1) which guarantees showing only TCP packets that their ACK flag is set. The same filter that shows the destination ip address is applied because ACK packets are suppose to be sent to the victim server. Moreover, (tcp.seq == 1)) && (tcp.ack == 1) assure that the ACK packets are only the first ACK packets in the TCP connection and they are not acknowledgments other usage. We also used (!http) filter to not show http packets. Finally, (tcp.nxtseq >1) is used which shows the TCP ACK packets that shows the next sequence number that is larger than 1. That means it will show the entire TCP segment of reassembled PDUs that we do not need; however, by using (!) to ignore the reassembled PDUs we get !(tcp.nxtseq >1) that show only ACK packets that are not part of transmitting files. By applying all of the filters together, we get only ACK packets in the three-way handshake that establish connections. The entire filter will be as follows:

(tcp.flags.SYN == 0) && (tcp.flags.ack == 1) && (!http) && (tcp.seq == 1) && (tcp.ack == 1) && !(tcp.nxtseq >1) && (eth.dst == MAC)

In the following Table 2, we summarize the filters with their description.

Table 2. Filters for ACK and Description.

Filter	Description
(tcp.flags.SYN == 0)	SYN flag is not set (Does not exist in the packet)
(eth.dst == MAC address)	The destination MAC address is the server's MAC address.
!http	No HTTP packets
(tcp.flags.ack == 1)	ACK flag is set (exist in the packet)
(tcp.seq == 1) && (tcp.ack == 1)	Guarantee showing Only the first TCP packets in the single connection
!(tcp.nxtseq >1)	The packet is not a part of a reassembled PDU

In Figure 6, we show the Smart- Counting algorithm that we propose and it is as follows:

1- Capture packets by using Wireshark library.
2- Store captured packets in queue 1.
3- Check if the time that is suggested is exceeded then go to step 4, if not go back to step 1.
4- Take the queue 1 and apply two filters:
   a. First filter will store the SYN packets in queue 2.
   b. Second filter will store the ACK packets in queue 3.
5- Subtract the number of SYN packets in queue 2 from the number of ACK packets in queue 3 (the result is the number of malicious SYN packets).
6- If the percentage of the malicious SYN packets exceeds %80 from the whole number of SYN packets go to step 7, if not go to step 1.
7- Set the SYN flood attack detected method to "True." For further processing to prevent the attack.

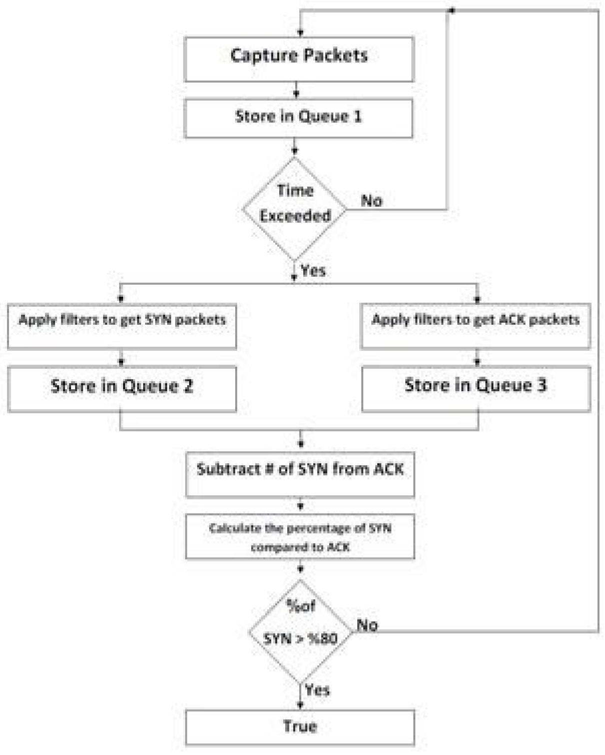
Capture Packets
Store in Queue 1
Time Exceeded
No
Yes
Apply filters to get SYN packets
Apply filters to get ACK packets
Store in Queue 2
Store in Queue 3
Subtract # of SYN from ACK
Calculate the percentage of SYN compared to ACK
%of SYN > %80
No
Yes
True

Figure 6. Smart-Counting Algorithm

The second method is called the Backlog queue Evaluator. Backlog queue is the space in the memory where every SYN packet is being kept until its connection is established or the time for storing it expires. Backlog queue is not a static place in the memory. Each program that supports establishing connections has its own Backlog queue that exists when running a server or a program. In SYN flood attack, Backlog queue gets filled by receiving a huge number of SYN packets with not intention to establish a connection, so it becomes up to the expiration time to discard these packets. Therefore, the huge number of sent SYN packets keeps the Backlog queue full. Consequently, no one will be able to establish a connection with the targeted server. As a countermeasure, increasing the Backlog queue size was proposed; however, this idea did not keep SYN flood attack from causing DoS attack. Backlog queue Evaluator is a new method to access Backlog queue and measure its status. As shown in Figure 7, if SYN packets in Backlog queue are less than 80%, then it is in a normal status; however, whenever it reaches 80%, then we detect that a SYN flood attack is conducted.

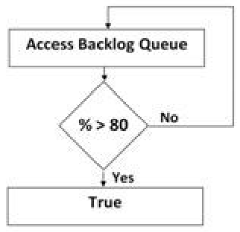

Figure 7. Backlog queue Evaluator Algorithm

By combining both methods together as shown in Figure 8, whenever both methods result in "True," we be sure that SYN flood attack is performed. Therefore, an alarm is triggered showing that SYN flood Attack has been detected.

6.2 Discussed methods:

Basically, the algorithm relies on counting SYN and ACK flagged TCP packets. However, there are some difficulties behind the scene. The first issue is having too many ACK flagged packets in our transmission. So we created an algorithm to identify the three-way handshaking SYN and ACK flagged packets. The algorithm relies on flag and sequence number at the same time. What we have seen in WireShark is that the sequence number starts from 0 and for ACK it starts from 1. However, we cannot use it as a base algorithm to identify the pure SYN packets and ACK packets during the 3-way handshake messages.

7. Implementation:

The implementation was been done in C++. The reason is that during the test we found that because Wireshark has too many features, it stops working when launching SYN flood attack. Therefore, we implemented the algorithm from scratch. The implementation contains 3 different Engines:

1. Capture Engine
2. Decrypt Engine
3. Signature Engine

Capture Engine is the main and most important part of this work. We connected to network adapters and capture all traffics. The library that were used is Winsock2. We modified winsock2 to listen to all traffics that is received. Every time it captures a single traffic, it sends it into the IDS. This engine runs as a separate thread to capture the packet and send it to the Decrypt Engine. Figure 9 shows the class diagram of Capture Engine.

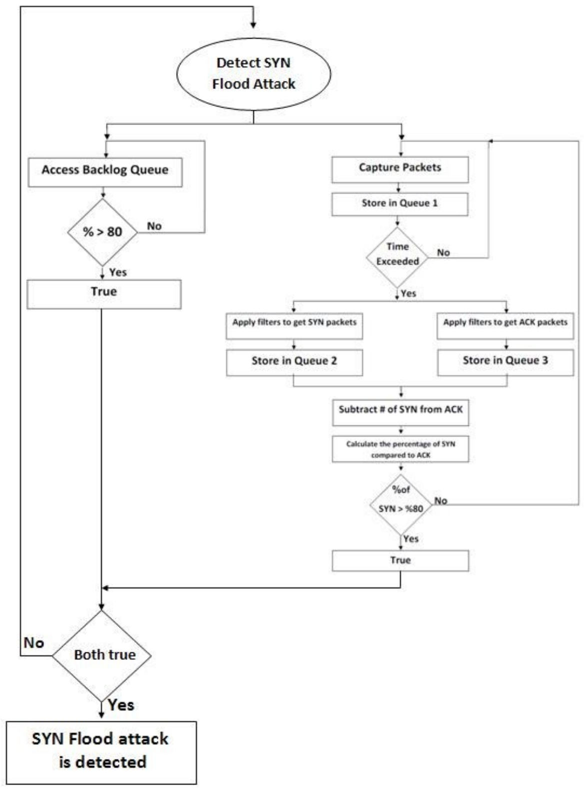

Figure 8. SYN Flood Attack Detection Algorithm

In order to decrypt the RAW data into understandable packet we have to have a structure to decode the headers and based on those data, we could decide if this packet is needed or not. The following structures and classes are designed to help the engine to decrypt the RAW data into IP and TCP packet. If the engine gets a corrupted RAW data, the IDS drops it at very early stage so the IDS can increase its performance with a large number of packets. Figure 10 shows the 4 classes and structures that help decrypt engine to get the meaningful data out of RAW data.
Inside the decrypt engine, we implemented the methods proposed in this paper to detect the SYN flood attacks. As we discussed the algorithm in previous sections, we count the number of the SYN flagged packets and match them with number of ACK flagged packets. If the numbers are increasing in a fast rate, we raise the alarm.

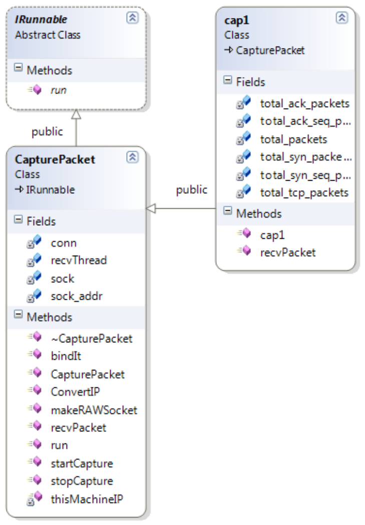

Figure 9. Capture Engine.

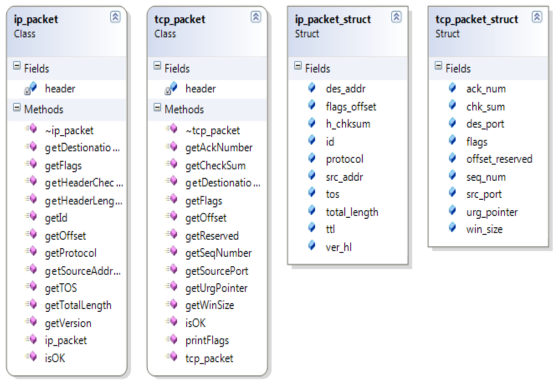

Figure 10. Decrypt Engine.

8. Experiments:

During the tests, we found that some SYN flood programs are not following the routine attacks. And with previous implementation we could not get good results. So we extended our IDS into a flexible SYN flood intrusion detection system. We used antivirus and worm detection mechanism and create the Signature based Engine. With this approach, if our previous method failed to recognize the attacks, our second engine will check the signature of the attack and compare it with the library that it has. If it matches, then it counts the number of the attack. If it passes the maximum number of malicious packets, it raises the alarm. Figure 11 shows the class diagram for the Signature Engine.

Dealing with thousands of packets per milliseconds requires a massive computation. The IDS contains 2 separate files. The first one is an executable file that uses all previous classes and help of structures to decode and identify the attacks. And the Second file contains all the signature of Syn flood attacks. This file designed so that users also are able to add new signatures into database in a very easy, understandable and text fashion way. Finally, based on our experiment, the IDS was able to identify the SYN flood attack.

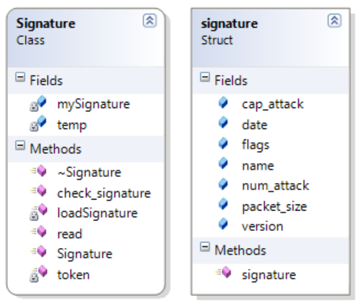

Figure 11. Signature Engine.

9. Conclusion:

In this paper we present the problem which is SYN flood attack and present a number of the latest methods that are proposed by other researches. Then, we proposed two unique solutions in order to detect it at early steps so that the server performance will not go down. The implementation part shows how we were able to detect SYN flood. For future work, implementing others' researches methods in the proposed IDS might increase the efficiency and reduce the false positives.

REFERENCES

[1] Information Sciences Institute University of Southern California, "RFC793 - Transmission Control Protocol," RFC 793, September1981

[2] Frederick K. K. "Studying Normal Traffic, Part Three: TCP Headers," security focus 2001

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

[4] Sun, C. Fan, J. and Liu, B. "A robust scheme to detect SYN flooding attacks," in International Conference on Communications and Networking in China (ChinaCom), Shanghai, China, August 22-24 2007

[5] Bloom, B. H. "Space/time trade-offs in hash coding with allowable errors," Communications of the ACM, vol. 13, no. 7, pp. 422-426, 1970

[6] Changhua Sun, Chengchen Hu, Yi Tang and Bin Liu, "More accurate and fast SYN flood detection, "

[7] Seung-won S., Ki-young K., Jong-soo J. "d-sat: detecting syn flooding attack by two-stage statistical approach"Applications and the Internet, 2005. Proceedings. The 2005 Symposium on 31 Jan.-4 Feb. 2005 Page(s):430 – 436

[8] Smith, C., Matrawy, " Comparison of operating system implementations of SYN flood defenses (Cookies)" Communications, 2008 24th Biennial Symposium on 24-26 June 2008 Page(s):243 - 246