# TCP-SYN Flooding Attack in Wireless Networks

Mitko Bogdanoski
Military Academy "General Mihailo Apostolski"
Skopje, R. Macedonia
mitko.bogdanoski@ugd.edu.mk

Tomislav Shuminovski, Aleksandar Risteski
Faculty of Electrical Engineering and Information Technologies
Skopje, R. Macedonia
{tomish, acerist}@feit.ukim.edu.mk

***Abstract*— This paper concerns the TCP (Transmission Control Protocol) vulnerabilities which gives space for a DoS (Denial of Service) attacks called TCP-SYN flooding which is well-known to the community for several years. The paper shows this attack in wireless as well as wired networks using perl synflood script, Wireshark network analyzer server, Windows 2008 server, and OPNET simulation environment. Using these tools an effects of this attack are shown. Finally, some effective practical mitigation techniques against SYN flooding attack for Linux and Windows systems are explained.**

***Index Terms* - DoS, Flooding, TCP-SYN, Wireshark, OPNET Modeler**

## I. Introduction

Nowadays, there are many attacks intended to deprive legitimate users from accessing any kind of network resources and functions. Any act which denies legitimate use of a service can be classed as a Denial of Service (DoS) attack. A DoS attacks are major security threats to the services provided through the Internet resulting in large scale revenue losses. One specific kind of DoS attack which is large-scale cooperative attack, typically launched from a large number of compromised hosts, is Distributed Denial-of-Service (DDoS). DDoS attacks are bringing about growing threats to businesses and Internet providers around the world. While many methods have been proposed to counter such attacks, they are either not efficient or not effective enough.

Moreover, the analysis shows that the DDoS attacks which use TCP and TCP-SYN flooding are the most prevalent among them ([1], [2]). However, flooding DDoS attacks are distinct from other attacks, for example, those that execute malicious code on their victim. These attacks floods the victim with a large volume of traffic and continuous data stream disables the victim from providing services to the legitimate users. These types of attacks are the mass of all attacking packets directed at the victim, which poses the threat, rather than the contents of the packets themselves. In that context, the Flooding DoS attacks are classified as resource depletion form of attacks. Moreover, these types of attacks pose the greatest problem in today's network infrastructures. Subverting the use of protocols, such as TCP or UDP, enables an attacker to disrupt on-line services by generating a traffic overload to block links or cause routers near the victim to crash. Considering the TCP-SYN flooding attack, it should be mentioned that it is a DoS method affecting hosts that run TCP server processes. Although this paper is considering the effects of this attack on wireless networks, however considering the affected layer from this attack and the way of conducting the attack there is no main difference with the wired networks. Considering the TCP-SYN flooding attack, as a well-known DoS method affecting hosts that run TCP server processes (the three-way handshake mechanism of TCP connection), nowadays, despite the original one, a lot of variations of it are still seen. Although there are many effective techniques against TCP-SYN flooding attack exist, and even RFC4987 is covering some common mitigation techniques against this attack yet there is no single mechanism (schemes) for effective defense.

This paper is organized as follows. Section 2 illustrates short reviews on TCP-SYN flooding attacks. Moreover, Section 3 gives related works in this area. Section 4 describes practical demonstration of the TCP-SYN flood attack using perl script synflood and Section 5 withdraws some simulation results and analysis of the effects of TCP-SYN and DDOS TCP-SYN flooding attack. In Section 6 some practical example to protect against this type of attack are explained. Finally, Section 7 concludes our work.

## II. TCP-SYN Flooding Attacks

The TCP is connection oriented and reliable, in-sequence delivery transport protocol. It provides full duplex stream of data octets and it is the main protocol for the Internet. Most nowadays services on Internet relay on TCP. For example mail (SMTP, port 25), old insecure virtual terminal service (telnet, port 23), file transport protocol (FTP, port 21) and most important in this case also is the hyper text transfer protocol (HTTP, 80) better known as the world wide web services (WWW). Almost everything uses TCP someway to do their communications over the network - at least the interactive ones.

In TCP-SYN flooding attack, the "SYN" stands for the Synchronize flag in TCP headers. The SYN flag gets set when a system first sends a packet in a TCP connection, and indicates that the receiving system should store the sequence number included in this packet. In this kind of flooding attack, the focus is given on the Flags, six different bits that may be sent to represent different conditions, such as initial sequence number (SYN), that the acknowledgement field is valid (ACK), reset the connection (RST), or close the connection (FIN).

During the TCP-SYN flooding attack [1], the attacking system sends TCP-SYN request with spoofed source IP address to the victim host. These SYN requests appear to be legitimate. The spoofed address refers to a client system that does not exist. Hence, the final ACK message will never be sent to the victim server system. This results into increased number of half-open connections at the victim side. A backlog queue is used to store these half-open connections. These half-open connections bind the resources of the server. Hence, no new connections (legitimate) can be made, resulting in DoS or DDoS. The victim server is unable to respond to the requests for Domain Name System (DNS) service coming from legitimate users (this attack is illustrated in Fig. 2).

Generally in the literature, there are three types of TCP-SYN flooding attacks, which are going out in the nowadays Internet networks: Direct Attack, Spoofing Attack and Distributed Direct Attack (see Fig.: 1-3).

If attackers rapidly send SYN segments without spoofing their IP source address, this is assign as a direct attack (Fig. 1). This method of attack is very easy to perform because it does not involve directly injecting or spoofing packets below the user level of the attacker's operating system. It can be performed by simply using many TCP connect() calls, for instance. To be effective, however, attackers must prevent their operating system from responding to the SYN-ACKS in any way, because any ACKs, RSTs, or ICMP messages will allow the listener to move the TCB out of SYN-RECEIVED. When detected, this type of attack is very easy to defend against, because a simple firewall rule to block packets with the attacker's source IP address is all that is needed. This defense behavior can be automated, and such functions are available in off-the-shelf reactive firewalls.

Furthermore, TCP-SYN spoofing attacks uses IP address spoofing, which might be considered more complex than the method used in a direct attack, in that instead of merely manipulating local firewall rules, the attacker also needs to be able to form and inject raw IP packets with valid IP and TCP headers. Moreover, the IP address spoofing techniques can be categorized into different types according to what spoofed source addresses are used in the attacking packets.

The three common IP spoofing types are: random spoofing, subnet spoofing and fixed spoofing [4].

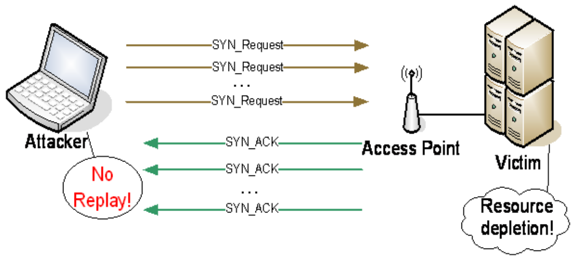


Figure 1. TCP-SYN Flooding: Direct attack

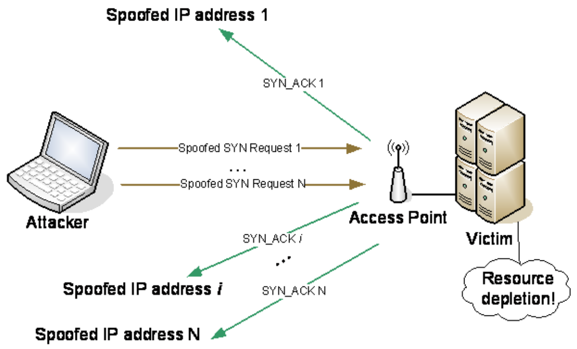


Figure 2. TCP-SYN Flooding: Spoofing attack

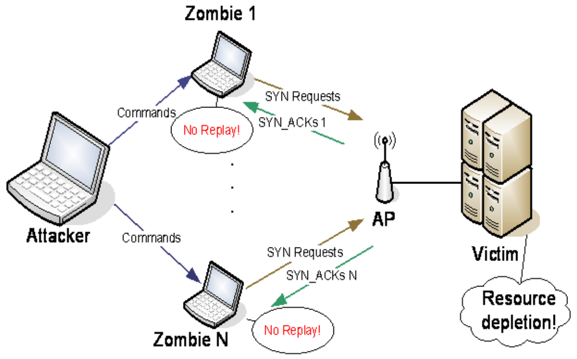


Figure 3. TCP-SYN Flooding: Distributed direct attack

In random spoofing, the attacker randomly generates 32-bit numbers for use as source addresses of the attacking packets. In subnet spoofing, the addresses are generated from the address space corresponding to the subnet in which the agent machine resides. For example, a machine which is part of the 194.149.134.0/24 network may spoof any IP address in the range from 194.149.134.0 to 194.149.134.255. Another type of IP spoofing, called fixed spoofing, chooses source addresses from a given list. In this case, the attacker typically wants to perform a reflector attack or impose a blame for attack on several specific machines.

Moreover, for spoofing attacks, a primary consideration is address selection. If the attack is to succeed, the machines at the spoofed source addresses must not respond to the SYN-ACKs that are sent to them in any way. A very simple attacker might spoof only a single source address that it knows will not respond to the SYN-ACKs, either because no machine physically exists at the IP address presently, or because of some other property of the address or network configuration. Another option is to spoof many different source addresses, under the assumption that some percentage of the spoofed addresses will be unrespondent to the SYN-ACKs. This option is accomplished either by cycling through a list of source addresses that are known to be desirable for the purpose, or by generating addresses inside a subnet with similar properties. If only a single source address is repetitively spoofed, this address is easy for the victim to detect and filter. In most cases a larger list of source addresses is used to make defense more difficult. In this case, the best defense is to block the spoofed packets as close to their source as possible.

Generally, the defense against spoofed flooding traffic, especially with subnet spoofing, is really difficult, but in the literature can be found several schemes for defense. One of them is [5] which is based on a storage-efficient data structure and a change-point detection method. Through trace-driven simulations, it is shown that this method is accurate and efficient to detect the SYN flooding attacks, due to the fact that it achieves shorter detection time and small storage space. Moreover, in [6] a novel defense mechanism that makes use of the edge routers that connect end hosts to the Internet to store and detect the outgoing SYN, ACK or incoming SYN/ACK segment is proposed. That is accomplished by maintaining a mapping table of the outgoing SYN segments and incoming SYN/ACK segments and establishing the destination and source IP address database. The results of simulation given in [6] are showing that the approach can yield accurate DDoS flooding attack alarms at early stage.

On the other hand, assuming the attacker is based in a "stub" location in the network (rather than within a transit Autonomous System (AS), for instance), restrictive network ingress filtering [7] by stub ISPs and egress filtering within the attacker's network will shut down spoofing attacks—if these mechanisms can be deployed in the right places. Because these ingress/egress filtering defenses may interfere with some legitimate traffic, such as the Mobile IP triangle routing mode of operation, they might be seen as undesirable, and are not universally deployed. Moreover, the IP Security (IPsec) also provides an excellent defense against spoofed packets, but this protocol generally cannot be required, because its deployment is currently limited, and it is usually impossible for the listener to ask the initiator's ISPs to perform address filtering or to ask the initiator to use IPsec, defending against spoofing attacks that use multiple addresses requires more complex solutions.

The real limitation of single-attacker spoofing-based attacks is that if the packets can somehow be traced back to their true source, the attacker can be easily shut down. Although the tracing process typically involves some amount of time and coordination between ISPs, it is not impossible. A distributed version of the SYN flooding attack, in which the attacker takes advantage of numerous zombie machines/processes throughout the Internet, is much more difficult to stop. In the case shown in Fig. 3, the zombies use direct attacks, but in order to increase the effectiveness even further, each zombie could use a spoofing attack and multiple spoofed IP addresses. Currently, distributed attacks are feasible because there are several "botnets" or "zombie armies" of thousands of compromised machines that are used by criminals for DoS attacks. Because zombie machines are constantly added or removed from the armies and can change their IP addresses or connectivity, it is quite challenging to block those types of TCP-SYN flooding attacks.

## III. RELATED WORK

In the literature there are many methods and frameworks which are proposed in order to detect the TCP-SYN flooding attacks. The authors of [1] detected the SYN flooding attacks at leaf routers that connect end hosts to the Internet, which utilizes the normalized difference between the number of SYNs packets and the number of FIN (RST) packets in a time interval. If the rate of SYNs packets is much higher than that of FIN (RST) packets by a non-parametric cumulative sum algorithm, the router recognizes that some attacking traffic is mixed into the current traffic. Similar work is presented in [8], where the fast and effective method for detecting TCP-SYN flooding attacks is given. Moreover, a linear prediction analysis is proposed as a new paradigm for DoS TCP-SYN flooding attack detection. The proposed mechanism makes use of the exponential backoff property of TCP used during timeouts. By modeling the difference of SYN and SYN&ACK packets, it is shown that this approach is able to detect an attack within short delays. Again this method is used at leaf routers and firewalls to detect the attack without the need of maintaining any state. However, considering the fact that the sources of attack can be distributed in different networks, there is a lack of analysis for the traffic near the sources and also the detection of the source of SYN flooding attack in TCP based low intensity attacks is missing.

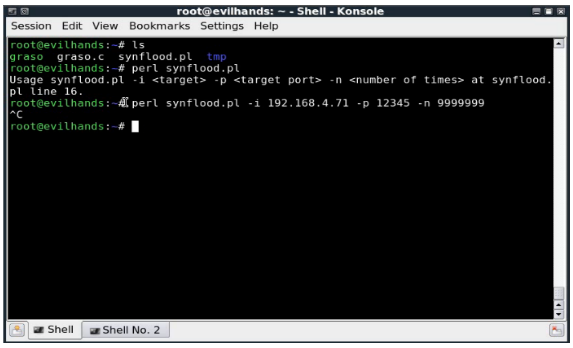
Moreover, a quite similar (with the previous two papers) approach was used in [9], which also considers a non-parametric cumulative sum algorithm; however the authors apply it to measure the number of only SYN packets, and by using an exponential weighted moving average for obtaining a recent estimate of the mean rate after the change of SYN packets. In [10] three counters algorithms for SYN flooding defense attacks are given. The three schemes include detection and mitigation. The detection scheme utilizes the inherent TCP valid SYN–FIN pairs behavior, hence is capable of detecting various SYN flooding attacks with high accuracy and short response time. The mitigation scheme works in high reliable manner for victim to detect the SYN packets of SYN flooding attack. Although the given schemes are stateless and required low computation overhead, making itself immune to SYN flooding attacks, the attackers may retransmit every SYN packet more than one time to destroy the mitigation function. It is necessary to make it more robust and adaptive.

In [11], the authors built a standard model generated by observations from the characteristic between the SYN packet and the SYN+ACK response packet from the server by a program for the activity of the server. The authors of [12] proposed a method to detect the flooding agents by considering all the possible kinds of IP spoofing, which is based on the SYN/SYN-ACK protocol pair with the consideration of packet header information. The Counting Bloom Filter is used to classify all the incoming SYN-ACK packets to the sub network into two streams, and a nonparametric cumulative sum algorithm is applied to make the detection decision by the two normalized differences, with one difference between the number of SYN packets and the number of the first SYN-ACK packets and another difference between the number of the firs SYN-ACK packets and the number of the retransmission SYN-ACK. Moreover, in [13] a simple and efficient method to detect and defend against TCP-SYN flooding attacks under different IP spoofing types is proposed. The method makes use of a storage-efficient data structure and a change-point detection method for distinguishing complete three-way TCP handshakes from incomplete ones. The presented experiments in [13] consistently show that their method is both efficient and effective in defending against TCP-based flooding attacks under different IP spoofing types. However there is a lack of autoimmunization of processes within the shame for setting the parameters. Additionally, the method is not evaluated in a reasonably large real network.

Moreover, there are also some other related studies such as SYN cookies, SYN filtering mechanisms [14], SYN cache, SYN proxy (firewall), SYN kill, D-SAT [15] and DiDDeM ([16] and [17]), and more related studies is in [5], [6], [18] and [19]. In the [5] and [6] an early stage detecting method (ESDM) is proposed. The ESDM is a simple but effective method to detect SYN flooding attacks at the early stage. In the ESDM the TCP-SYN traffic is forecasted by autoregressive integrated moving average model, and non-parametric cumulative sum algorithm is used to find the SYN flooding attacks according to the forecasted traffic. The ESDM achieves shorter detection time and small storage space. However, these exiting methods or defense mechanisms that against SYN flooding attack are effective only at the later stages, when attacking signatures are obvious [6].

## IV. PRACTICAL DEMONSTRATION OF THE TCP-SYN FLOOD ATTACK

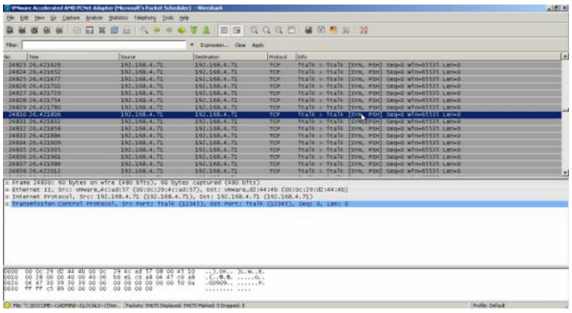
For a practical demonstration of TCP-SYN attack we will use the perl script synflood (Fig. 4), and in order to make an analysis of traffic during the attack we will use the Wireshark tool.



root@evilhands: ~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell
Shell No. 2

```
root@evilhands:~# ls
graso  graso.c  synflood.pl  tmp
root@evilhands:~# perl synflood.pl
Usage synflood.pl -i <target> -p <target port> -n <number of times> at synflood.
pl line 16.
root@evilhands:~# perl synflood.pl -i 192.168.4.71 -p 12345 -n 9999999
^C
root@evilhands:~# 
```

Figure 4. TCP-SYN flooding attack using perl script synflood



VMware Accelerated AMD PCNet Adapter (Microsoft's Packet Scheduler) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

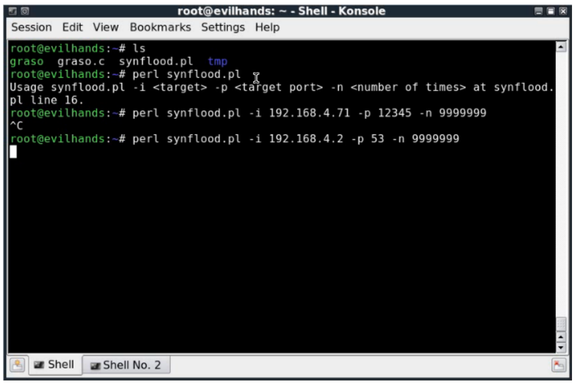
```
No.   Time       Source        Destination   Protocol Info
24823 26.421626  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24824 26.421652  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24825 26.421677  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24826 26.421702  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24827 26.421729  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24828 26.421754  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24829 26.421780  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24830 26.421806  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24831 26.421832  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24832 26.421858  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24833 26.421884  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24834 26.421909  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24835 26.421935  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24836 26.421961  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24837 26.421986  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0
24838 26.422012  192.168.4.71  192.168.4.71  TCP  italk > italk [SYN, PSH] Seq=0 Win=65535 Len=0

⊞ Frame 24830: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
⊞ Ethernet II, Src: Vmware_4c:ad:57 (00:0c:29:4c:ad:57), Dst: Vmware_d2:44:4b (00:0c:29:d2:44:4b)
⊞ Internet Protocol, Src: 192.168.4.71 (192.168.4.71), Dst: 192.168.4.71 (192.168.4.71)
⊞ Transmission Control Protocol, Src Port: italk (12345), Dst Port: italk (12345), Seq: 0, Len: 0

0000  00 0c 29 d2 44 4b 00 0c  29 4c ad 57 08 00 45 10   ..).DK.. )L.W..E.
0010  00 28 00 00 40 00 40 06  b0 e1 c0 a8 04 47 c0 a8   .(..@.@. .....G..
0020  04 47 30 39 30 39 00 00  00 00 00 00 00 00 50 0a   .G0909.. ......P.
0030  ff ff c5 89 00 00 00 00  00 00 00 00               ........ ....

File: "C:\DOCUME~1\ADMINI~1\LOCALS~1\Tem...  Packets: 54670 Displayed: 54670 Marked: 0 Dropped: 0   Profile: Default
```

Figure 5. Traffic analysis using Wireshark network analyzer

Using the perl script the Wireshark network analyzer server is attacked. The attack starts and stops for a short time after Wireshark collects a sufficient number of packets for traffic analysis (Fig. 5).

During the traffic analysis in Wireshark it can be noticed that the address of the attacker is the same as that of the client which means that the client address is spoofed, and it looks like the client sends TCP packets to itself i.e., it looks like it burdens itself with huge amount packages.

To demonstrate the consequences of this attack we are login into our Windows Server 2008 that has installed DNS role and we are attacking it with synflood perl script. Although before the attack it is notable that the processor is not burdened at all, and the burden on network traffic is very low, however in the moment when the attack is executed on port 53 (DNS port) (Fig. 6), small increasing on CPU burden on 2008 server can be noticed, but very evident is the increase of the burden of network bandwidth, which increases over 7Mbps (Fig. 7). We must take into consideration that the attack was committed by only one machine. It is obvious that in case of attack by 2 or 3 machines or in case of organized DDoS attack the server consequences would be worse, even catastrophic, which would cause to be unable to answer on the DNS requests.



root@evilhands: ~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell
Shell No. 2

```
root@evilhands:~# ls
graso  graso.c  synflood.pl  tmp
root@evilhands:~# perl synflood.pl
Usage synflood.pl -i <target> -p <target port> -n <number of times> at synflood.
pl line 16.
root@evilhands:~# perl synflood.pl -i 192.168.4.71 -p 12345 -n 9999999
^C
root@evilhands:~# perl synflood.pl -i 192.168.4.2 -p 53 -n 9999999
```

Figure 6. Attacking DNS server using perl script synflood

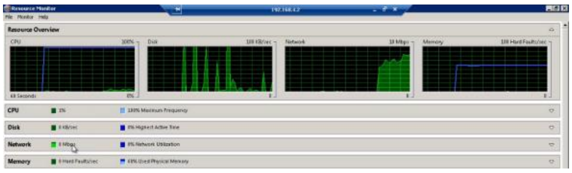


Figure 7. Resource monitoring after attack

Immediately after stopping the attack the burden of the CPU load and network bandwidth is reduces to its primary values. It is necessary to know that realization of this type of attack requires a great processing time in order to process large amount of packets that are sent to the client, which makes understandable the usage of DDoS attacks from multiple machines in order to increase processing time for packets generation as well as increasing of the network bandwidth to send those packages to the client-victim.

## V. Simulation Results and Analysis of the Effects of TCP-SYN and DDOS TCP-SYN Flooding Attack

In this section we are going to present our simulation experiment on TCP-SYN flooding attack. We have subdivided this section into two subsections. We start this section giving the configuration of our whole test bed. Then we present results we got from specific scenarios.

### A. *The Scenario Configuration*

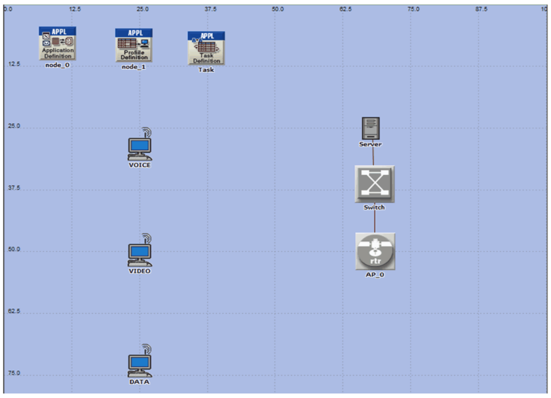


Figure 8. OPNET Scenario

In order to make a detailed analysis we are considering a scenario with three WLAN nodes, where one of them is used for simulating data traffic, another for simulating the video and the third for voice transmission. The scenario takes place on 100x100m workspace and it is presented in Fig. 8. For serving different requirements (data, video, voice) from different nodes we are using a server that is configured to support all the above services. The wireless communication is realized via a wireless access point.

In this paper three different scenarios are reviewed: during the first scenario there is no attack, during the second scenario TCP-SYN flood attack is conducted, and during the third scenario we are considering the situation when the network is attacked by DDoS TCP-SYN flood attack. In all these scenarios as a basis we use the scenario without attack, and what changes is the intensity of the attack, which depends on the number of attackers. The time of simulation is set to 20 minutes. We should also mention that examined environment presented in this paper is 802.11e capable.

### B. *Simulation Results and Analysis*

The average values from the obtained simulation results are presented. As results *global statistic* of the simulated scenario is shown. These statistics are scoped to the simulation as a whole, in contrast to local statistics, which are scoped to a particular queue or processor. In other words, multiple processes, as well as pipeline stages, all at different locations in the model's system, can contribute to the same shared statistic. This is done by referring to the statistic by name and obtaining a *statistic handle*.

Although from the simulations we obtained a lot of results which only confirmed our expectation, however in this paper we will present only few of them. Considering the fact that in all scenarios there are three legitimate clients with different requirements from the server, hereinaftersome of the obtained results considering each legitimate node individually are explained.

#### 1) *Node: Voice*

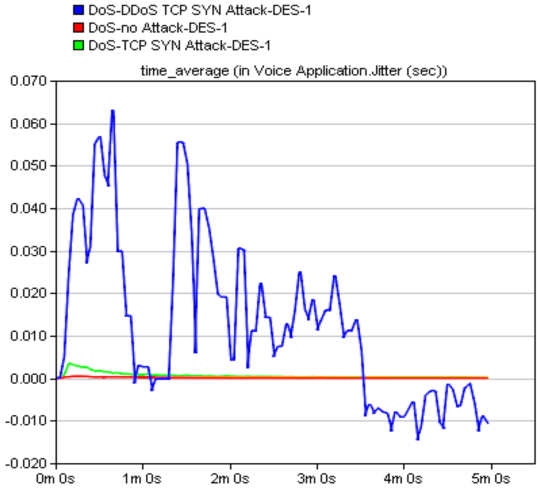


Figure 9. Voice Application Jitter

In Fig. 9 the average voice jitters which occur during voice transmission in the situation when no attack is performed and when the wireless network is under the TCP-SYN and DDoS TCP-SYN Attack is presented. Even the jitter value when there is no attack conducted is 0, this value during attacks are dramatically increased. Special case is the situation when the network/server is attacked by DDoS TCP-SYN Attack which is causing higher voice jitter or higher voice packet delay variation.

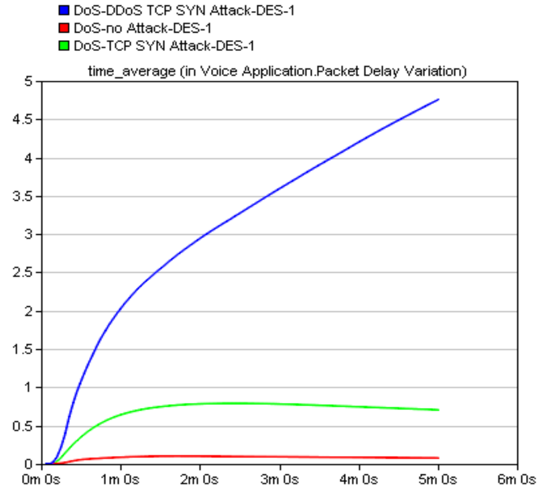


Figure 10. Voice Application Packet Delay Variation

The packet delay variation on voice application is shown in Fig. 10. This parameter shows the variance among end to end delays for voice packets received by this node. End to end delay for a voice packet is measured from the time it is created to the time it is received. Again, it is noticeable that this delay is 10 time higher when TCP-SYN flood attack is committed. The situation of DDoS attack is even worse. As it can be seen the delay is constantly increasing during this attack and it is even incomparable higher than the values from other two scenarios.

### 2) *Node: Video*

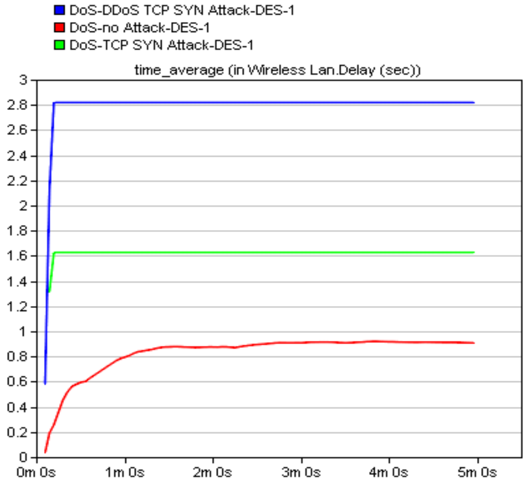


Figure 11. Video Application WLAN Delay

The wireless LAN delay (Fig. 11) represents the end-to-end delay of all the data packets that are successfully received by the WLAN MAC and forwarded to the higher layer. This delay includes queuing and medium access delays at the source MAC, reception of all the fragments individually, and the relay of the frame via AP, if the source and destination MACs are non-AP MACs of the same infrastructure BSS. Increased number of attackers causes higher delay.

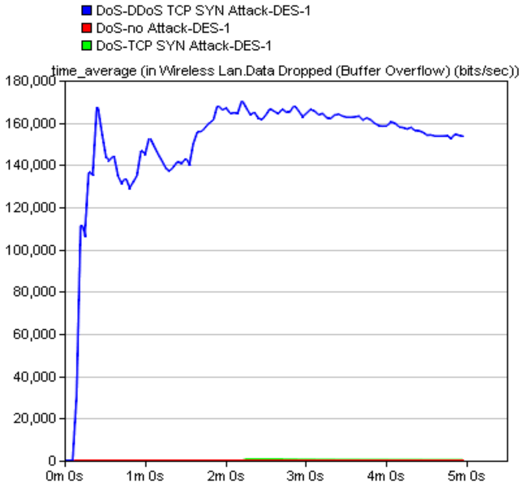


Figure 12. Video Application WLAN data dropped (buffer overflow)

Fig. 12 shows higher layer video data traffic dropped (in bits/sec) by the WLAN MAC due to full higher layer data buffer. In this case during the TCP-SYN flood attack by one attacker the value of this parameter is almost the same with the scenario without attack. Opposite of this, in the case of DDoS attack the number of dropped packets is very high which is understandable if we consider that the buffers are filled by multiple malicious nodes.

### 3) *Node: Data*

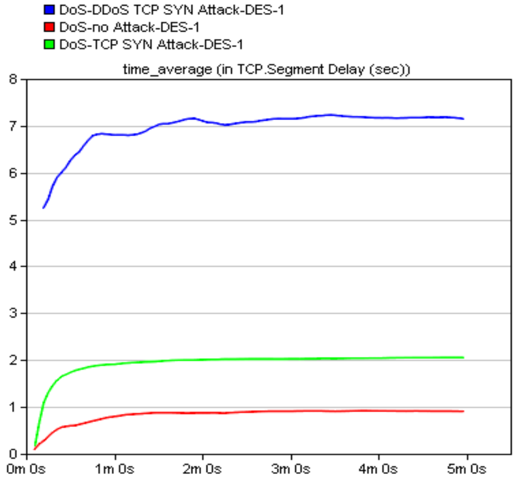


Figure 13. Data TCP segment delay

Fig. 13 shows the delay (in seconds) of segments received by the TCP layer in this node, for all connections. It is measured from the time a TCP segment is sent from the source TCP layer to the time it is received by the TCP layer in the destination node. Using this parameter we collect "TCP Delay" statistics for delays experienced by complete application packets submitted to TCP. This delay during the TCP-SYN flood attack is two time higher than the situation without attack, but again the worst case is  the scenario with conducted DDoS attack, where the value of TCP segment delay is seven time higher than the scenario without attack.

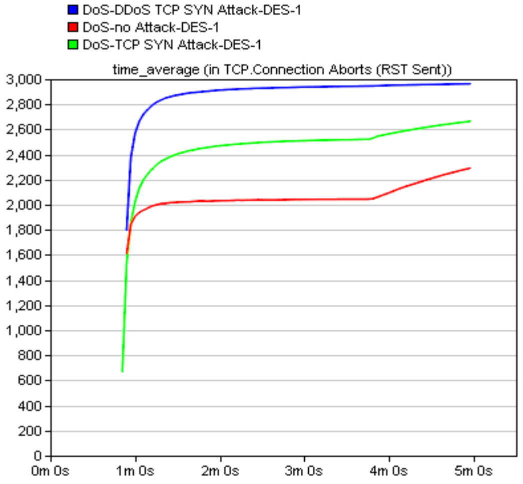


Figure 14. Data TCP connection aborts (RST sent)

The parameter TCP connection aborts (RTS Sent) shows the total number of TCP connections aborted because the maximum number of retransmissions has been reached (Fig. 14). Statistic is updated each time the connection is abort is initiated by the TCP process at this node. The method used by OPNET to calculate this parameter is incrementing each time a TCP connection is aborted at this node. There is not difference from the previous explanations. Against the worst case is the situation when the network is attacked by DDoS attack. Moreover, Fig. 15 shows the TCP load parameter or the total number of packets submitted to the TCP layer by the application layer in this node, for all connections.

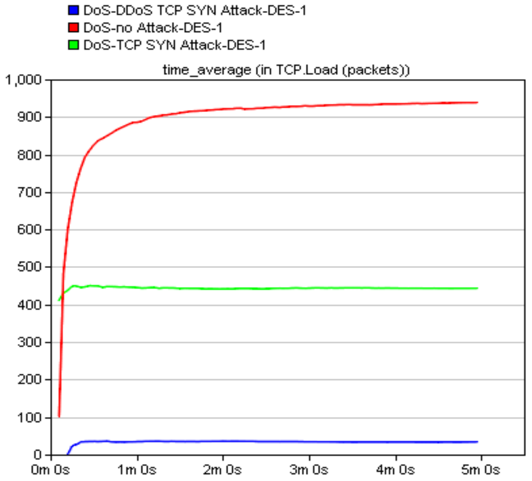


Figure 15. Data TCP Load

In case of TCP-SYN flood attack this total number of packets is reduced on half compared with scenario without attack, but situation is much better than the situation when DDoS attack is conducted, in which case the number of these packets is dramatically reduced.

## VI. Protection Against TCP-SYN Flood Attack

The universal answer on the question how to protect against TCP-SYN attack is that this type of attack is a very problematic issue.

In the case of TCP-SYN attack, the client does not return a response to a SYN / ACK packet, and does not return an ACK packet to the server to open connections between the client and server. Because of that the server remains with half-opened connections which are stored in memory, and until the timeout expiration the server tries to send a new SYN / ACK packet to the client.

The base problem is the large quantities of generated "half-opened" connections which server keeps in memory until the expiration of their timeout, which can ultimately lead to server crash, but nowadays the most common consequence is the inability/sluggishness of opening a new connection due to the occupancy of the limit for the half-Opened connections.

Although one of the indicators can be increased CPU time and memory, today it will not be visible on the new generation of servers, or it will be marginally noticeable. In the case of web server what will surely be noticeable, at least at first glance, is extremely slow opening of the web pages without any "obvious" reasons such as overload of the servers' resources or bandwidth link.

Netstat command can easily detect whether there is a TCP-SYN attack. On Linux systems, half-opened connections will be marked as SYN_RECV, while on Windows systems they will be marked as SYN_RECEIVED.

Primer (Linux) [20]:

```
>netstat -tuna | grep :80 | grep SYN_RECV
tcp 0 0 192.168.2.1:80 15.55.82.20:1309          SYN_RECV
tcp  0  0  192.168.2.1:80  51.1.5.7:1743          SYN_RECV
tcp  0  0  192.168.2.1:80  209.112.192.126:4988  SYN_RECV
tcp  0  0  192.168.2.1:80  53.12.51.1:1724        SYN_RECV
…
```

This command filters all opened connections per port 80 (HTTP), and status SYN_RECV (half-opened connection). If the output contains a lot of rows that contain SYN_RECV, it is likely that a TCP-SYN attack occurred. Consideration should be given that filtering IP address, from which the attack comes, probably will have no impact on reducing the attack because the attacker can easily spoof packets, respectively spoofs IP addresses from which an attack is coming. The IP addresses in most cases will be related to the actual source of the attack.

Although the ultimate solution to prevent TCP-SYN flood attack is very complex issue, it is possible to perform the basic actions that will reduce the impact of this attack.

In Chapter 3 we already mentioned several proposed effective mechanisms for detection and prevention against this type of attack. Even more, some effective solutions against this attack are considered in RFC 4987 [21]. Anyway, in this section we are considering a practical approach to mitigate the effects of SYN flood attack.

**LINUX:** Enabling SYN Cookies function within the Linux kernel will allow "ignoring" of the TCP-SYN flood attacks, allowing the server to stop the half-opened connections when they fill their limit. While the connection will be terminated, the server will send a SYN / ACK packet to the client, and if it receives an ACK response from the client it will reconstruct the previous half-opened connection, and enable communication with the client.

To enable SYN cookies inside the kernel the following actions should be taken:

```
sysctl -w net.ipv4.tcp_syncookies=1
```

In order to enable SYN cookies during every restart it is necessary to edit/etc/sysctl.conf, and enter into a new row:

```
net.ipv4.tcp_syncookies = 1
```

It is recommended to increase the SYN backlog queue with a default value of 1024 to 2048:

```
sysctl -w net.ipv4.tcp_max_syn_backlog=2048
```

To set options for each server restart is necessary to edit/etc/sysctl.conf, and enter into a new row:

```
net.ipv4.tcp_max_syn_backlog = 2048
```

In the case of TCP-SYN attacks in the logs the following should occur, but despite the attack server will still be able to normally open a new connection toward the customers.

```
[1116377.589736] possible SYN flooding on port 80. Sending cookies.
[1116439.567828] possible SYN flooding on port 80. Sending cookies.
[1116500.631623] possible SYN flooding on port 80. Sending cookies.
```

**WINDOWS:** In order to configure TCP-SYN flood protection for Windows 2000 and 2003 servers several registry entries should be set (Table 1).

First step is to enable SYN attack protection. SynAttackProtect defines whether it is included TCP-SYN flood protection, and if the value is 0 then it is disabled, which means that it should be set to the value 1 for better or value 2 for best protection against TCP-SYN flood attack [22].

TABLE I. CONFIGURING REGISTRY ENTRIES

| Registry Path | NAME | VALUE |
|---|---|---|
| HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TcpIp\Parameter | *SynAttackProtect* | 2 |
| | *TcpMaxPortsExhausted* | 5 |
| | *TcpMaxHalfOpen* | 500 |
| | *TcpMaxHalfOpenRetried* | 400 |
| | *TcpMaxConnectResponseRetransmissions* | 2 |
| | *TcpMaxDataRetransmissions* | 2 |
| | *EnablePMTUDiscovery* | 0 |
| | *KeepAliveTime* | 300000 |

A denial of service (DoS) attack against Windows servers is to send it a "name release" command. This will cause it to release its NetBIOS, preventing clients from accessing the machine. By setting *NoNameReleaseOnDemand* with registry path KEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters to DWORD "1" instead of the default "0", other machines will be prevented from causing the protected system's name from being released.

Using the *TcpMaxPortsExhausted, TCPMaxHalfOpen TCPMaxHalfOpenRetried* it is possible to define values, primarily limits of the possible number of half-opened connections. Within 2008/ Vista/7 Windows the protection from the TCP-SYN flood attacks is turned on for by default, and cannot be excluded. For Windows 2008 servers it is possible to apply an increase in CPU and memory load, because the server releases the half-opened connections depending on system load. Manually adjustment option as in Windows 2003 Server is not possible, but the server adapts the configuration to current needs / conditions.

## VII. CONCLUSION

In this paper one of the vulnerabilities of TCP protocol which leads to the TCP-SYN flooding DoS attack is shown. Furthermore a practical demonstration of the effects of this attack using perl script, synflood and Wireshark network analyzer server, as well as Windows Server 2008 is presented. To confirm the results from the practical demonstration and to see the effects of the TCP-SYN flood attack and DDoS TCP-SYN flood attack on other parameters on different types of traffic (voice, video, data) in 802.11e environment we used OPNET Modeler simulation tool. The results we obtained are as we expected to be. Namely, during the TCP-SYN flooding attack the situation with all network parameters became worse, but even this results are incomparable with the dramatically deteriorated results obtained during the simulation of the last scenario when DDoS TCP-SYN flood have been conducted. Finally, we have given some practical example for mitigating the effects of TCP-SYN flood DoS attack in Linux and Windows environment.

## REFERENCES

[1] H. Wang, D. Zhang, and K. G. Shin, “Detecting SYN flooding attacks”, *in Proceedings of Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM)*, volume 3, pages 1530-1539, June 23-27 2002

[2] D. Moore, G. M. Voelker, and S. Savage, “Inferring Internet Denial-of-Service Activity,” *in Proceedings of the 10th USENIX Security Symposium*, Aug. 2001, pp. 9–22.

[3] J. Postel, “Transmission Control Protocol” STD 7, September 1981. [RFC 793] http://tools.ietf.org/html/rfc793

[4] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms", *ACM SIGCOMM Computer Communications Review*, 34(2):39—54, April 2004

[5] S. Qibo, W. Shangguang, Y. Danfeng and Y. Fangchun, "An Early Stage Detecting Method against SYN Flooding Attacks", *China Communication*, Vol. 4, pp. 108-116, November 2009.

[6] G. Wei, Y. Gu and Y. Ling, "An Early Stage Detecting Method against SYN Flooding Attack," csa, pp.263-268, *International Symposium on Computer Science and its Applications*, 2008.

[7] P. Ferguson and D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing,” BCP 38, RFC 2827, May 2000. http://www.ietf.org/rfc/rfc2827.txt

[8] D. M. Divakaran, H. A. Murthy and T. A. Gonsalves, ``Detection of SYN Flooding Attacks Using Linear Prediction Analysis," *14th IEEE International Conference on Networks*, ICON 2006, pp. 218-223, Sep. 2006.

[9] V. A. Siris and P. Fotini, “Application of Anomaly Detect Algorithms for Detecting SYN Flooding Attack” *Elsevier Computer Communications*, 29: 1433-1442, 2006.

[10] S.Gavaskar, R.Surendiran and Dr.E.Ramaraj, "Three Counter Defense Mechanism for TCP-SYN Flooding Attacks", *International Journal of Computer Applications*, Volume 6–No.6, pp.12-15, September 2010.

[11] T. Nakashima and S. Oshima, “A detective method for SYN flood attacks”, *First International Conference on Innovative Computing, Information and Control*, 2006.

[12] D. Nashat,X. Jiang and S. Horiguchi, “Detecting SYN Flooding Agents under Any Type of IP Spoofing”, *IEEE International Conference on e-Business Engineering table of contents*, 2008.

[13] W. Chen and D.-Y. Yeung, "Defending Against TCP-SYN Flooding Attacks Under Different Types of IP Spoofing", *ICN/ICONS/MCL ’06*, IEEE Computer Society, pp. 38-44, April 2006.

[14] A. Yaar, A. Perrig and D. Song, "StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense*", IEEE Journal on Selected Areas in Communications*, Volume 24, no. 10, pp.: 1853-1863, October 2006.

[15] S.-W. Shin, K-Y. Kim and J.-S. Jang, “D-SAT: detecting SYN flooding attack by two-stage statistical approach”, *Applications and the Internet*, pp.:430 – 436, 2005.

[16] J. Haggerty, T. Berry, Q. Shi and M. Merabti, “DiDDeM: a system for early detection of TCP-SYN flood attacks”, *GLOBECOM*, 2004.

[17] J. Haggerty, Q. Shi and M. Merabti, "Early Detection and Prevention of Denial-of-Service Attacks: A Novel Mechanism With Propagated Traced-Back Attack Blocking", *IEEE Journal On Selected Areas In Communications*, Vol. 23, No. 10, pp.: 1994-2002, October 2005.

[18] T. Peng, C. Leckie and K. Rammamohanarao, “Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems”, *ACM Computing Surveys*, Vol. 39, Issue 1. 2007.

[19] B. Xiao, W. Chen, Y. He and E.H.-M. Sha, “An active detecting method against SYN flooding attack*”, Parallel and Distributed Systems*, 2005.

[20] A. Chin, Detecting and preventing SYN Flood attacks on web servers running Linux, *Linux Forum*, 21. January 2011

[21] W. M. Eddy, “TCP-SYN Flooding Attacks and Common Mitigations,” RFC 4987, August 2007. http://tools.ietf.org/html/rfc4987

[22] support.microsoft.com