I. J. Computer Network and Information Security, 2013, 8, 1-11
Published Online June 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2013.08.01

Analysis of the SYN Flood DoS Attack

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

Tomislav Shuminoski and Aleksandar Risteski
Faculty of Electrical Engineering and IT, Ss. Cyril and Methodius University, Skopje, R. Macedonia
{tomish, acerist}@feit.ukim.edu.mk

Abstract — The paper analyzes systems vulnerability targeted by TCP (Transmission Control Protocol) segments when SYN flag is ON, which gives space for a DoS (Denial of Service) attack called SYN flooding attack or more often referred as a SYN flood attack. The effects of this type of attack are analyzed and presented in OPNET simulation environment. Furthermore, the paper presents two anomaly detection algorithms as an effective mechanism against this type of attack. Finally, practical approaches against SYN flood attack for Linux and Windows environment which are followed by are shown.

Index Terms — DoS, Flooding, SYN flooding, OPNET Modeler, Anomaly detection

I. Introduction

Today, the work of most of the important and crucial services dependent on rapid development of the technologies and their operation is almost inconceivable without the Internet usage, so any disruption in the operation of the Internet can be very inconvenient. Considering the fact that the Internet was originally designed for openness and scalability without much concern for security, it's clear that the malicious users can exploit the design weaknesses of the Internet to wreak havoc in the operation of most of services.

According to the last analyses cybercrime and hactivism became the primary motivation behind cyber-attacks (Fig. 1) [1].

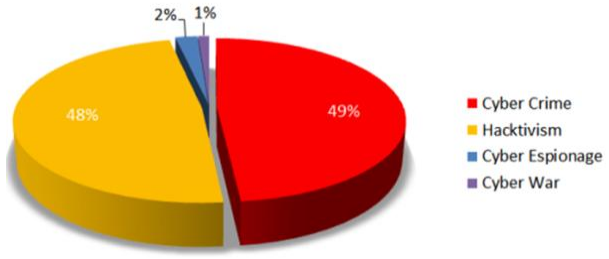


Figure 1: Motivations behind cyber-attacks in March, 2013 [1]

Amongst many other types of cyber-attacks, the DoS attacks are major security threats to the services provided through the Internet resulting in large scale revenue losses. This conclusion can be confirmed by statistic given in Fig. 2 [1].

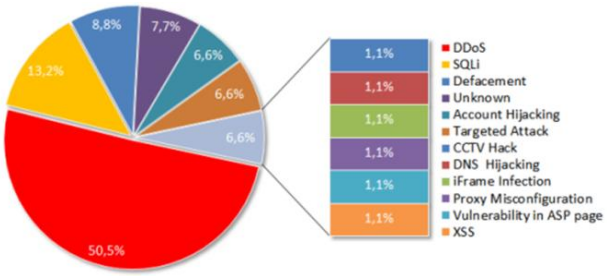


Figure 2: Distribution of attack techniques, March, 2013 [1]

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.

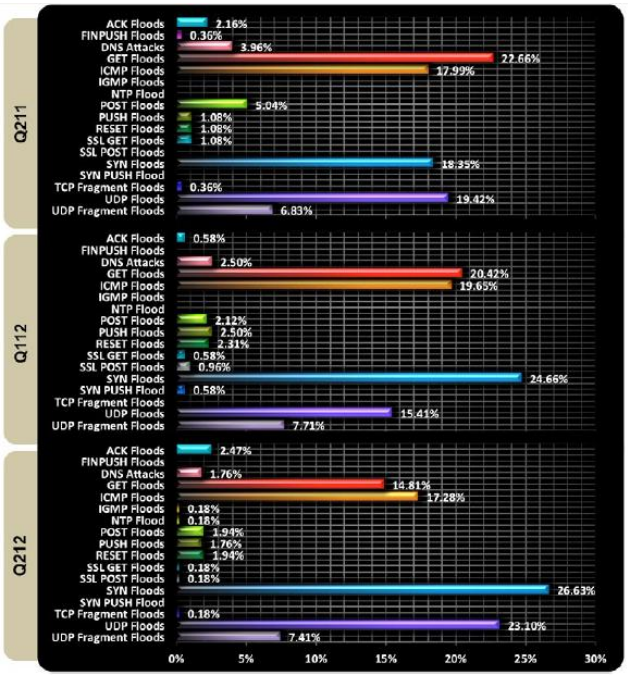


Figure 3: DDoS flooding attacks statistic [2]

Moreover, the analysis, as it is shown in Fig. 3, shows that the DDoS attacks which use TCP and SYN flood are the most prevalent among them [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. Flooding DDoS 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, these 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.

The rest of the paper is organized as follows. Section 2 gives related work in this area. Section 3 withdraws some simulation results and analysis of the effects of SYN and DDoS SYN flood attack. In Section 5 two effective anomaly detection algorithms are described and some practical examples to protect against this type of attack are explained. Finally, Section 6 concludes our work.

II. Background and Related Work

At the beginning the paper briefly describes operation of SYN flood attack (also known as a TCP-SYN flooding attack). Then some relevant previous studies regarding this topic are discussed.

A. SYN flood attack

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.

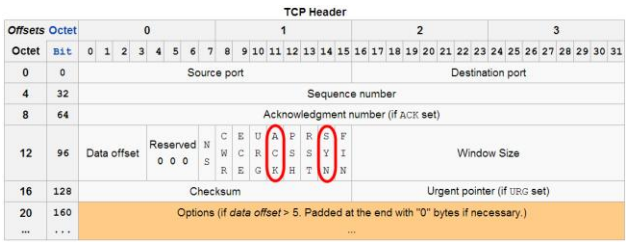


Figure 4: TCP Header

In SYN flood 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. Fig. 4 shows the TCP header, which is twenty bytes long without options. In this illustration each line represents four bytes. 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).

As it was mentioned before, in SYN flooding [3], the attacking system sends 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.

SYN flooding attack is a DoS method affecting hosts that run TCP server processes (Fig. 5).

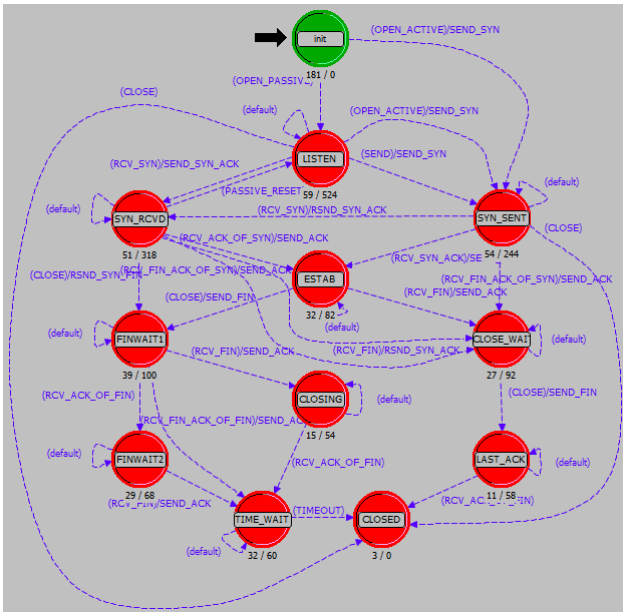


Figure 5: TCP State Transmission Diagram

Although this paper is analyzing 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.

The SYN flood attack is well-known DoS method which affects hosts that run TCP server processes (the three-way handshake mechanism of TCP connection). Nowadays, despite the original one, a lot of its variations can be seen. Although there are many effective techniques against SYN flood attack, and even RFC4987 is covering some common mitigation techniques against this attack [4], yet there is no single mechanism (schemes) for effective defense.

During the SYN flood attack [3], the attacking system sends 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.

Generally in the literature [3, 4], there are three types of SYN flood attacks, which are going out in the nowadays Internet networks: Direct Attack, Spoofing Attack and Distributed Direct Attack.

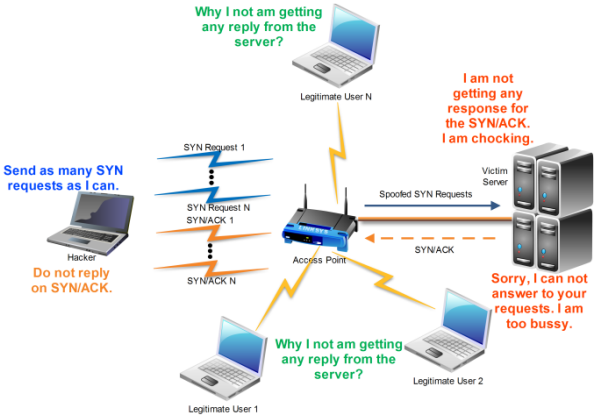


Figure 6: SYN Flooding: Direct attack

If attackers rapidly send SYN segments without spoofing their IP source address, this will cause direct attack (Fig. 6). This type of SYN flooding attack does not involve directly injecting or spoofing packets below the user level of the operating system of the attacker. One way to perform thid type of attack is by simply using many TCP connect() calls [5]. However, the attacker's operation system must not respond to the SYN-ACKs, because any ACKs, RSTs, or ICMP (Internet Control Message Protocol) messages will allow the listener to move the TCB (Transmission Control Block) out of SYN-RECEIVED. In order to prevent its operating system from responding to the SYN-ACKs, the attacker can set some firewall rules which can filter outgoing packets to the listener (allowing only SYNs out), or filter incoming packets so that any SYN-ACKs are discarded before reaching the local TCP processing code.

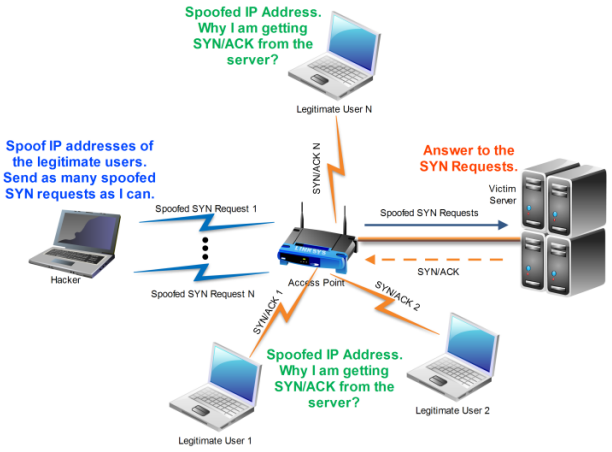


Figure 7: SYN Flooding: Spoofing attack

On the other hand, the SYN spoofing attack (shown in Fig. 7) uses IP address spoofing, which might be considered more complex than the method used in a direct attack. During this type of SYN flooding attack 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.

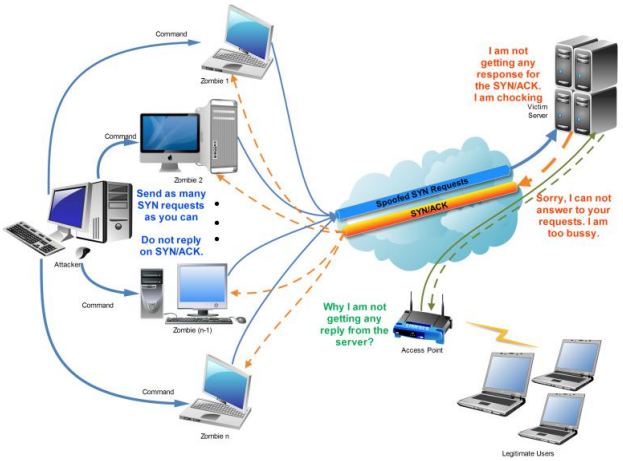


Figure 8: SYN Flooding: Distributed direct attack

A distributed version of the SYN flooding attack shown in Fig. 8 is the most dangerous amongst mentioned types of SYN flooding attacks. During this type of SYN flooding attack the attacker takes advantage of numerous zombie machines/processes throughout the Internet. In the case, 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 SYN flood attacks.

B. Related work

In the literature there are many methods and frameworks which are proposed in order to detect the SYN flood attacks. The authors of [3] 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 [6], where the fast and effective method for detecting SYN flood attacks is given. Moreover, a linear prediction analysis is proposed as a new paradigm for DoS SYN flood attack detection. The proposed mechanism makes use of the exponential backoff property of TCP used during timeouts. By modelling 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 [7], 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 [8] 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 [9], 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 [10] 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 first SYN-ACK packets and the number of the retransmission SYN-ACK. Moreover, in [10] a simple and efficient method to detect and defend against SYN flood 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 [11] 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 process automation within the sheme 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 [12], SYN cache, SYN proxy (firewall), SYN kill, D-SAT [13] and DiDDeM ([14] and [15]), and more related studies is in [16], [17], [18] and [19]. In the [16] and [17] 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 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 which oppose to the SYN flooding attack are effective only at the later stages, when attacking signatures are obvious [17].

III. Simulation Results and Analysis of the Effects of SYN and SYN Flood DDoS Attack

In this section the paper presents simulation experiment on SYN flooding attack. The section is subdivided into two subsections. The section starts giving the configuration of our whole test bed. Then it presents results obtained from specific scenarios.

A. The Scenario Configuration

In order to make a detailed analysis a scenario with three WLAN nodes is considered, 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. 9. For serving different requirements (data, video, voice) from different nodes we used server is configured to support all the above services. The wireless communication is realized via a wireless access point.

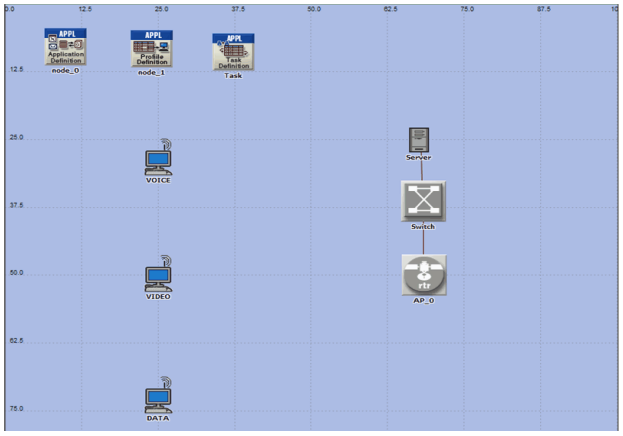


Figure 9. OPNET Scenario

In this paper three different scenarios are reviewed: during the first scenario there is no attack, during the second scenario SYN flood attack is conducted, and during the third scenario we are considering the situation when the network is attacked by DDoS SYN flood attack. In all these scenarios as a basis a scenario without attack is considered, 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. It should be also mentioned 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 a lot of results are obtained which only confirmed our expectation, however this paper will present only few of them. Considering the fact that in all scenarios there are three legitimate clients with different requirements from the server, hereinafter some of the obtained results considering each legitimate node individually are explained.

1) Node: Voice

In Fig. 10 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 SYN and DDoS 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 SYN Attack which is causing higher voice jitter or higher voice packet delay variation.

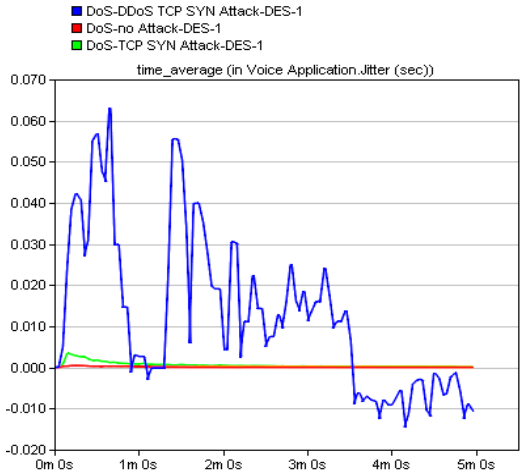


Figure 10. Voice Application Jitter

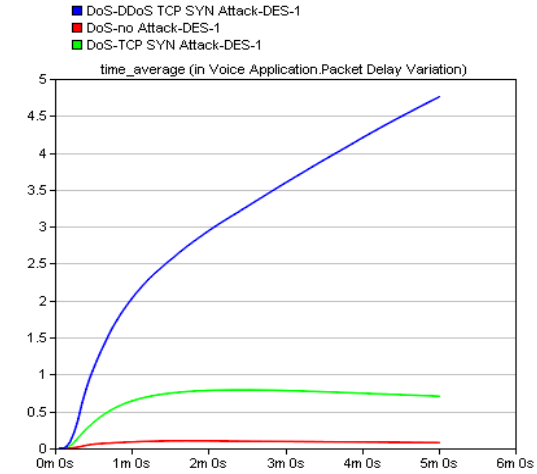


Figure 11. Voice Application Packet Delay Variation

The packet delay variation on voice application is shown in Fig. 11. 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 times higher when 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

The wireless LAN delay (Fig. 12) 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.

Fig. 13 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 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 it is considered that the buffers are filled by multiple malicious nodes.

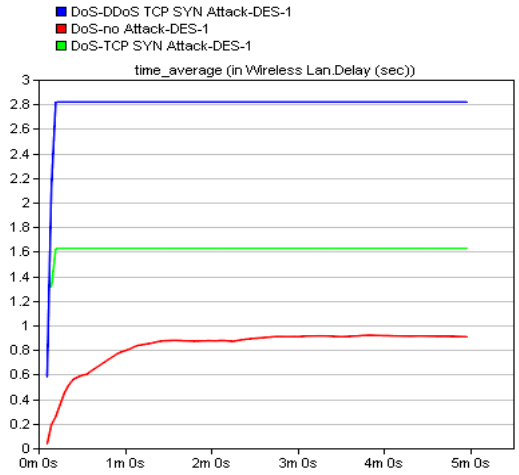


Figure 12. Video Application WLAN Delay

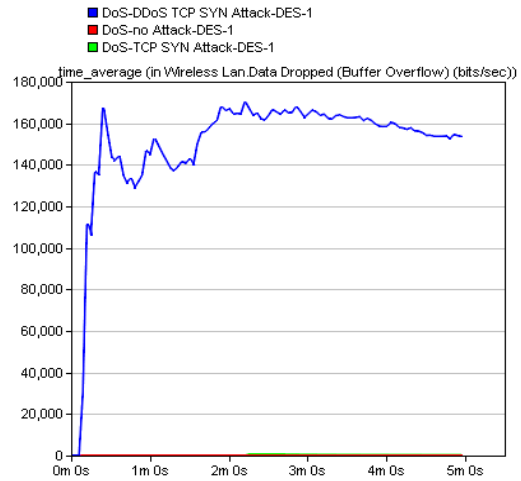


Figure 13. Video Application WLAN data dropped (buffer overflow)

The wireless LAN delay (Fig. 12) 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.

Fig. 13 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 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 it is considered that the buffers are filled by multiple malicious nodes.

3) Node: Data

Fig. 14 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 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.

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. 15). 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. 16 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.

In case of 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.

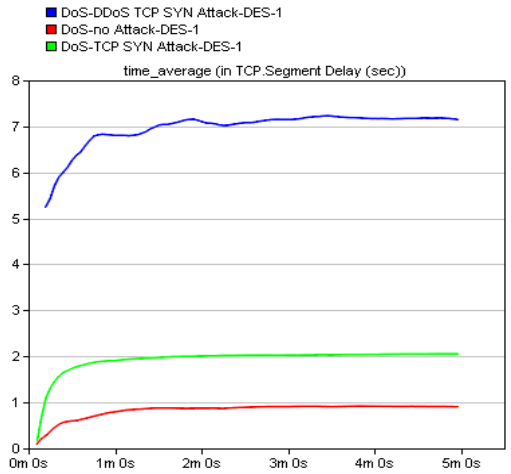


Figure 14. Data TCP segment delay

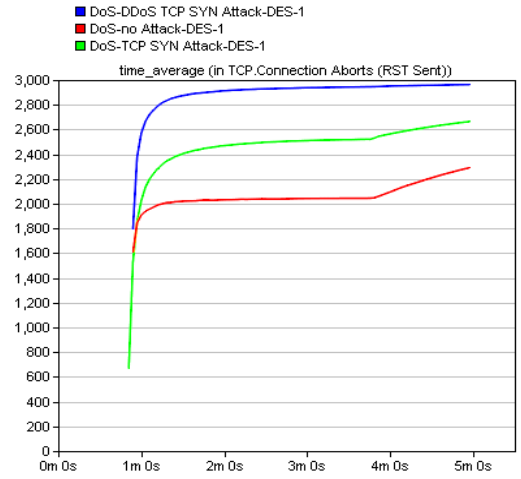


Figure 15. Data TCP connection aborts (RST sent)

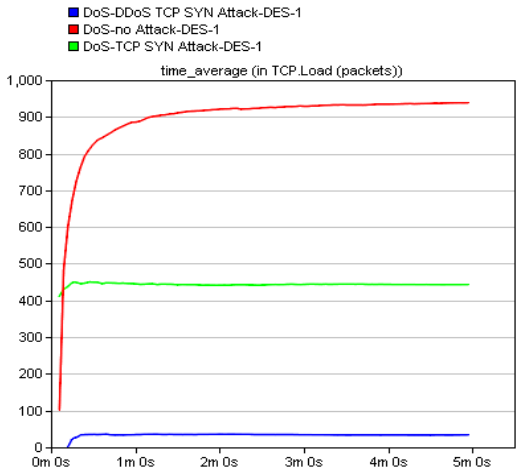


Figure 16. Data TCP Load

IV. Protection against SYN Flooding Attack

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

In the case of 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.

In this chapter we are considering two effective anomaly detection algorithms which can be used as effective tools for detection of the SYN-Flooding attack. Furthermore practical solutions to defend against this attack in Linux and Windows environment are presented.

A. Anomaly detection algorithms

Considering the fact that flooding attacks cause significantly increased SYN segment traffic as opposed to the period when there is no attack it is obvious that most appropriate countermeasures against SYN- Flood attack, and flooding attacks at all, will be based on anomaly detection algorithms. Here this section presents two anomaly detection algorithms. The first, an adaptive threshold algorithm, is a simple algorithm that detects anomalies based on violations of a threshold that is adaptively set based on recent traffic measurements, and the second is based on a used anomaly detection algorithm cumulative sum (CUSUM) algorithm.

Adaptive Threshold Algorithm: This is simple algorithm which measures network traffic (in our case SYN packets) and compare it with previously defined threshold. This threshold is adaptively set in certain period of time and is based on the estimated mean number of SYN packets. If measured traffic exceeds a particular threshold it will be defined as anomaly and alarm will be activated.

Let us suppose that the number of SYN packet in the *t*-th time interval is x_t, and measured mean rate prior to t is μ_{t-1}. In this case the alarm condition is as following:

If $x_t \geq (\alpha + 1)\mu_{t-1}$, then ALARM signaled at time t, where $\alpha > 0$ parameter indicates the percentage above mean value which is considered as a threshold. The mean μ_t can be computed over some past time window or using an exponentially weighted moving average (EWMA) of previous measurements

$$\mu_t = \beta\mu_{t-1} + (1-\beta)x_t \qquad (1)$$

where β is EWMA factor.

Flow chart in Fig. 17a shows how adaptive threshold algorithm can be used for detecting SYN flood attacks. On the arrival of packets the algorithm checks if the packets are TCP, and if this is True it checks if TCP packets are SYN or other type of TCP. If the SYN bit is on than it will update the record. Calculating the μ and α rate than it will check the condition for x_t either it is greater than $(\alpha + 1)\ \mu_{t-1}$ or not. If it is greater it will generate the alarm.

Cumulative Sum (CUSUM) Algorithm: This algorithm is based on change-points detection techniques based on hypothesis testing, i.e., the rising and falling edges of the anomalous period, which denotes the duration that the anomaly lasts, without prior knowledge on the distribution of the traffic flow. When the mean of a subset of samples becomes higher than a threshold, a change point is registered, and the level of change is

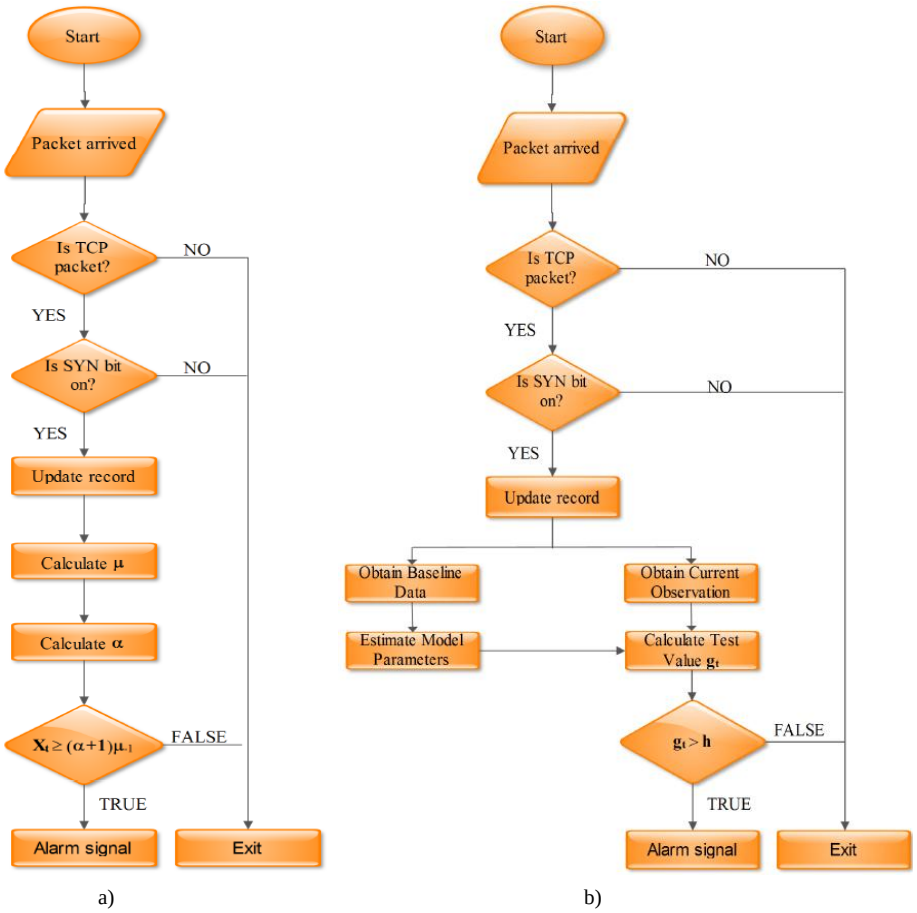
cumulated into an accumulator. The accumulator begins to discharge when the other change point is detected at the end of the anomaly period [7,20]. This algorithm was developed for independent and identically distributed random variables. According to the approach, there are two hypotheses, where the first corresponds to the statistical distribution prior to a change and the second to the distribution after a change. If s(y) is positive, the monitored random variable more likely conforms to the distribution after change than to the distribution before change. This is also a simple algorithm which measures network traffic (in our case SYN packets) and compare it with previously defined threshold (Fig. 17b).

This threshold is adaptively set in certain period of time and is based on the estimated mean number of SYN packets. If measured traffic exceeds a particular threshold it will be defined as anomaly and alarm will be activated.

In order to increase the power of hypothesis test, most of the anomaly detection algorithms a sequence of observations $\{y_t|t=a,\ldots,b\}$. If we suppose that the observations are independent, the log-likelihood ratio of the sequence is:

$$s(y_a, \ldots, y_b) = log \frac{\prod_{t=a}^{b} p\theta_1(y_t)}{\prod_{t=a}^{b} p\theta_0(y_t)} = \sum_{t=a}^{b} s(y_t) \qquad (2)$$

where parameter values before and after the change will be θ_0 and θ_1 respectively. The probability density function of Y with parameter θ *is* $p_\theta(y)$. If the result for the log-likelihood ratio $s(y_t)$ is positive, probably the random values conforms to the distribution after change than to the distribution before change. Considering this, it is possible to define a threshold h for $s(y_t)$ which can reject the null hypothesis H_0 (for the case $\theta=\theta_0$) and accept the alternative hypothesis H_1 (for the case $\theta=\theta_1$) at a given level of significance. This level is compatible to the probability of a false alarm.



a) b)

Figure 17. Flow charts of considered anomaly detection algorithms
a) Adaptive threshold algorithm, b) CUSUM algorithm

Here simple model of this algorithm is presented. The algorithm is based on fact that $S_t=s(y_1, \ldots, y_t)$ has a negative drift under normal conditions and a positive drift after change. Actually, decision function g_t of this algorithm compares the increase of S_t with respect to its minimum to a threshold h:

$$g_t = S_t - \min_{1 \le i \le t} S_i = max(0, s(y_t) + g_{t-1}) = [(g_{t-1} + s(y_t)]^+ \ge h; \quad g_0 = 0 \qquad (3)$$

where g_t and g_{t-1} are estimated differences in time-windows *t* and *t-1*. If g_t exceeds the given threshold value *h*, the alarm will be activated.

The threshold *h* allows trading off the mean detection delay and the mean time between false alarms. In the case when distribution of *Y* is unknown, $s(y_t)$ must be replaced by a statistic $u(y_t)$ with comparable properties. The expectation value of $u(y_t)$ depend on hypotheses. This value will be negative under null hypothesis H_0 and positive under alternative hypothesis H_1. This variant is often called non-parametric CUSUM algorithm.

Using statistics $u^+(y) = y - (\mu_0 + K)$ and $u^-(y) = (\mu_0 - K) - y$ where *K* is defined as a reference value we can detect positive and negative shifts respectively. Consequently following two decision function are defined:

$$g_t^+ = [g_{t-1}^+ + y_t - (\mu_0 + K)]^+ \geq h, \quad g_t^- = [g_{t-1}^- + y_t - (\mu_0 + K)]^+ \geq h \qquad (4)$$

where typical settings for reference value K are $\sigma/2$, 4σ and 5σ. σ is the standard deviation of Y_t.

The authors of [7] show that although a simple straightforward algorithm such as the adaptive threshold algorithm can have satisfactory performance for high intensity attacks, its performance deteriorates for low intensity attacks. They also show that an algorithm based on change point detection, such as the CUSUM algorithm, can exhibit robust performance over a range of different types of attacks, without being more complex.

B. Practical approaches against SYN-Flood attack

Netstat command can easily detect whether there is a 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) [21]:

>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 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 SYN flood attack is very complex issue, it is possible to perform the basic actions that will reduce the impact of this attack.

Section 3 discusses 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 [4]. Anyway, this section considers 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 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 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.

5.2.2. *WINDOWS*: In order to configure 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 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 SYN flood attack [22].

TABLE I. CONFIGURING REGISTRY ENTRIES

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

First step is to enable SYN attack protection. SynAttackProtect defines whether it is included 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 SYN flood attack [22].

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 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.

V. Conclusions

In this paper one of the vulnerabilities of TCP protocol which leads to the SYN flood DoS attack is shown. To see the effects of the SYN flood attack and DDoS SYN flood attack on other parameters on different types of traffic (voice, video, data) in 802.11e environment the OPNET Modeler simulation tool is used. The obtained results are expected. Namely, during the SYN flood 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 SYN flood have been conducted. Finally, effective anomaly detection algorithms against SYN Flood attacks are presented. Furthermore, the paper shows some practical example for mitigating the effects of SYN flood DoS attack in Linux and Windows environment.

References

[1] March 2013 Cyber Attacks Statistics, April 9, 2013, Available: http://hackmageddon.com/category/security/cyber-attacks-statistics/

[2] DDOS: coordinated attacks analysis, PenTest magazine, PenTest Extra 05/2012, Available: http://pentestmag.com/ddos-attacks-pt-extra-05_2012/

[3] 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, pp. 1530-1539, June 23-27, 2002.

[4] W. M. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," RFC 4987, August 2007. [Online]. Available: http://tools.ietf.org/html/rfc4987.

[5] W. Eddy, "Defenses Against TCP SYN Flooding Attacks", Cisco Internet Protocol Journal Volume 9, Number 4, December 2006, Available: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-4/syn_flooding_attacks.html

[6] 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.

[7] V. A. Siris and P. Fotini, "Application of Anomaly Detect Algorithms for Detecting SYN Flooding Attack" *Elsevier Computer Communications*, pp. 1433-1442, 2006.

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

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

[10] 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.

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

[12] 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.

[13] 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.

[14] J. Haggerty, T. Berry, Q. Shi and M. Merabti, "DiDDeM: a system for early detection of SYN flood attacks", *GLOBECOM*, 2004.

[15] 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.

[16] 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.

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

[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] Sheng-Ya Lin, Jyh-Charn Liu and Wei Zhao, "Adaptive CUSUM for Anomaly Detection and Its Application to Detect Shared Congestion", Technical Report– 1-2, Department of Computer Science, Texas A&M University, 2007.
[21] A. Chin, Detecting and preventing SYN Flood attacks on web servers running Linux, *Linux Forum*, 21. January 2011.
[22] support.microsoft.com.

Dr. Mitko Bogdanoski received his B.Sc. degree from the Military Academy, Skopje, Macedonia, and M.Sc. and Ph.D. degree from the Faculty of Electrical Engineering and Information Technologies, Ss Cyril and Methodius University, Skopje, Macedonia, in 2000, 2006 and 2012 respectively. He is an assistant professor at the Military Academy. He is an author of more than 30 International/National conference/journal publications. His research interests include wireless and mobile networks, MANET, WSN, cyber security, energy efficiency and communication theory.

MSc. Tomislav Shuminoski received the B.Sc. (2008) and M.Sc. (2010) degrees from Faculty of Electrical Engineering and Information Technologies, Ss Cyril and Methodius University, Skopje, Macedonia. He is a research and teaching assistant and currently is working toward his PhD degree at the Institute of Telecommunications, Faculty of Electrical Engineering and Information Technologies, Ss Cyril and Methodius University in Skopje, Macedonia. His research interests include quality-of-service provisioning in wireless and mobile multimedia networks, cross-layer design, wireless mesh networking, and communication theory. He is a member of IEEE since 2006.

Dr. Aleksandar Risteski received his B.Sc., M.Sc. and Ph.D. degrees in telecommunications at the Ss Cyril and Methodius University in Skopje, Macedonia in 1996, 2000 and 2004, respectively. He is currently a professor and a vice-dean for research and international cooperation at the same university, Faculty of Electrical Engineering and Information Technologies. In 2001, 2003 and 2004, he had several internships at IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, where he worked towards his Ph.D. degree. His research interests are in the fields of optical communications, secure communications and coding theory. He is an author of more than 50 journal and conference papers. Dr. Risteski was a project leader/participant in more than 30 research and industry-related projects, co-chairman of three international conferences and co-director of one NATO advanced research workshop.