Teaching Stateless and Stateful Firewall Packet Filtering: A Hands-on Approach

Zouheir Trabelsi, *UAE University*

Abstract - The need to use a practice and application oriented approach in information security education is paramount. A security education curriculum that does not give the students the opportunity to experiment in practice with security techniques cannot prepare them to be able to protect efficiently the confidentiality, integrity, and availability of computer systems and assets.

In this paper, first we discuss security issues with stateless basic packet filtering, and the concepts of stateful TCP, UDP and ICMP packet filtering. Then, we describe a comprehensive hands-on lab exercise implementation about how to identify whether a given firewall performs stateless or stateful packet filtering. The learning objective of the exercise is for students to better anatomize the concept of stateless and stateful firewall packet filtering through examples and experiments in an isolated network laboratory. The impact of offering the hands-on lab exercise on the students' grading performance is discussed. An anonymous questionnaire was administered to measure students' satisfaction level and to collect their feedback regarding the discussed hands-on lab exercise.

Index terms – Firewall, packet filtering, stateful firewall, stateless firewall, security policy.

I. INTRODUCTION

Nowadays, with the increase of information security programs, a number of laboratory experiments and laboratory-based courses have been developed for information security education [1-6]. In addition, since firewalls are an import topic in network security, most security education programs include courses that cover firewalls.

There is no doubt that hands-on lab exercises are a very effective way of learning the practical aspects of firewalls. A number of educational firewall systems have been previously developed. For example, the goal of the systems described by Garrido and Bandyopadhyay [7] and also Ye and Sandrasegaran [8] is to provide the students with a statistically supported understanding of a firewall's effectiveness. The goal of the system described by Williams and Yu [9] is to help students better understand the functions of a firewall and how to configure it, and to let students get hands-on experience.

Faculty of Information Technology, UAE University
Al-Ain, UAE
Email: Trabelsi@uaeu.ac.ae

In order to enhance firewall education, we describe, in this paper, a comprehensive hands-on lab exercise implementation about how to identify whether a given firewall performs stateless or stateful packet filtering. This paper assumes that the user has basic knowledge about TCP/IP protocols [10], TCP three-way handshake process, common Internet services, and packet filtering rules [11].

The rest of the paper is organized as follows: Section 2 discusses security issues with stateless firewall packet filtering. Sections 3, 4, and 5 discuss stateful TCP, UDP and ICMP packet filtering concepts, respectively. Section 6 describes a hands-on lab exercise implementation about how to verify a given firewall performs stateless or stateless packet filtering. Section 7 discusses the impact of offering the hands-on lab exercise on the students' performance; as well as students' satisfaction. Finally, Section 8 concludes the paper.

II. SECURITY ISSUES WITH STATELESS PACKET FILTERING

Firewalls control the access into and from the network based on a set of filtering rules, which reflect and enforce the organization's security policy [11]. Filtering rules related to TCP and UDP bi-directional services (such as http, ftp, and telnet) have to allow both traffic directions to cross the firewall. A TCP or UDP session has a client which is the computer that initiates the session, and a server which is the computer hosting the service. For example, the filtering rule shown in Table 1 allows bi-directional http traffic between the Web clients with IP addresses 192.168.1.1/24 and the Web servers with IP addresses 192.168.2.1/24 to across the firewall.

A TCP or UDP session is characterized by four attributes, namely the IP address of the client, the IP address of the server, the client port (known as source port), and the server port (known as destination port). Usually, the server port number allows to identify the nature of the offered service. For example, a Web session and Telnet session are almost always on TCP ports 80 and 23, respectively. However, the client port is usually chosen dynamically at run time by the operating system of the client host, and it is bigger than 1023. Therefore, the client port number is essentially unpredictable. It is important to notice that since the client port is unpredictable, the firewall should allow any session flow with any source port to cross the firewall. Consequently, in basic packet filtering, this would introduce

a very serious security vulnerability that allows malicious hosts to flood the target servers with unwanted traffic that may cause a Denial of Service (DoS) attack situation. To better anatomize this security problem, let's assume the following two security policies:

- *Security policy (SP #1):* We want to allow our internal hosts at IP addresses 192.168.1.1/24 to access outside Web servers (listening on TCP port 80) at IP addresses 192.168.2.1/24.
- *Security policy (SP #2):* In addition, we want to deny external hosts (192.168.2.1/24) from establishing TCP connections with the internal hosts (192.168.1.1/24).

The first security policy (SP #1) allows the internal hosts to establish Web connections with any external Web server. However, the second security policy (SP #2) allows preventing external hosts from establishing TCP connections on the internal servers, and consequently protects the internal servers from TCP SYN flooding attack, a kind of DoS attacks. Also, in case of the internal hosts are infected with remote controlled program based viruses, such as Trojan horses, the second security policy prevents malicious users from remotely connecting to the infected internal hosts. Remote controlled program based viruses are a very serious threat since they allow malicious users to fully control remote victim hosts.

The packets exchanged between the Web clients and servers will look like this:

Client-to-server packets:
Source IP = 192.168.1.1/24, Destination IP = 192.168.2.1/24, Source port = Y, Destination port = 80.

Where, 192.168.1.1/24 is the possible IP addresses of the Web clients, 192.168.2.1/24 is the possible IP addresses of the Web servers, and Y is an arbitrary port number selected by the Web client.

On the other hand, return traffic from the Web servers to the Web clients (Server-to-client), swaps IP addresses and port numbers, and looks like this:
Source IP = 192.168.2.1/24, Destination IP = 192.168.1.1/24, Source port = 80, Destination port = Y.

In a TCP session, if we consider only the SYN and ACK flags, there are only four types of exchanged TCP packets, as shown in Figure 1:

- Client-to-server packet: TCP packet with the flag SYN set, and the flag ACK unset.
- Server-to-client packet: TCP packet with the flags SYN and ACK set.
- Client-to-server packet: TCP packet with the flag SYN unset, and the flag ACK set.
- Server-to-client packet: TCP packet with the flag SYN unset, and the flag ACK set.

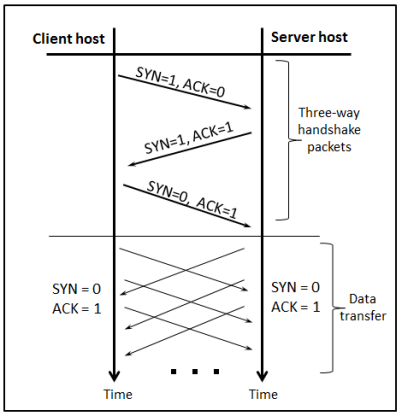

Figure 1. The types of packets exchanged in a TCP connection

Consequently, to allow bi-directional service traffic, the firewall should allow the above four types of TCP packets to pass through. For example, tables 2 and 3 list the filtering rules for the above two security policies SP#1 and SP#2, respectively.

Regrettably, an attacker may exploit rules 2 and 4 of Table 2 to conduct DoS attacks, since the two rules match packets based on their source ports. Remember that the source port is under the packet sender's control. The attacker on any host at spoofed IP addresses 192.168.2.1/24 can build fake packets with source port 80, destination any host at IP addresses 192.168.1.1/24, and a destination port of his choice. Fake packets crafted this way will cross the firewall because they match either Rule 2, if their flags SYN and ACK are set, or Rule 4, if their flag SYN is unset and their flag ACK is set. For example, Table 4 shows an example of a TCP packet that is rejected by the firewall since the packet attempts to establish a Web connection with an internal host. The packet is rejected by Rule 5 shown in Table 3.

However, Table 5 shows an example of malicious TCP packet that is allowed by the firewall to pass. The malicious TCP packet pretends that the TCP connection with source port 9000, has been already established, since its flag SYN is unset, and its flag ACK is set. Based on the above filtering rule list, the firewall will allow the malicious packet to pass. Consequently, flooding a target internal host with such malicious TCP packets may create a DoS attack situation at the target host.

Therefore, this simple example demonstrates the limitations of basic packet filtering. In fact, the major weakness of the filtering rules of tables 2 and 3 is that they allow malicious users to flood internal hosts with malicious TCP packets, such as shown in Table 5, which may create a DoS attack situation at the internal hosts. In basic packet filtering, this DoS attack may occur easily. Since, firewalls do not use mechanisms that allow to decide whether or not a given

TCP packet belongs to an already established session. In fact, stateless firewalls do not keep state of the ongoing TCP connection sessions, and do not remember what source port numbers the sessions' clients selected.

III. Stateful TCP Packet Filtering

To address the above security issue in basic packet filtering, firewalls keep track of established TCP connections. Practically, firewalls keep an entry, in a cache, for each open TCP connection. An entry of a TCP connection includes the client and server IP addresses, and the client and server port numbers. The client port number information was not fully known when the firewall administrator wrote the rules. However, when the connection is being set up, both port numbers are known, since they are listed in the packet's TCP header. All the packets that belong to an existing TCP connection, in both directions, are allowed to cross the firewall. This type of firewall is called stateful firewall.

The entries of the state cache of established TCP connections are created using a simple mechanism. That is, when the first packet (SYN packet) of a new TCP connection reaches the firewall, the firewall matches it against the set of filtering rules. If there is a filtering rule that allows the packet across, the firewall inserts a new entry into the cache, and the TCP connection state is set to the SYN_RCVD state. Once the two other remaining packets of the three-way handshake process are received, the TCP connection state transits to the ESTABLISHED state. Therefore, the first packet (SYN packet), of a TCP connection effectively opens a hole in the firewall, and the cache mechanism allows the return traffic to go through this hole.

After the TCP connection has been established, the decision to whether or not to allow subsequent TCP packets is based on the contents of the state cache. That is, when a subsequent TCP packet, with the flag SYN unset and the flag ACK set, reaches the firewall, the firewall checks whether an entry for the TCP connection it belongs to already exists in the cache. If the connection is listed in the cache, the packet is allowed through immediately. If no such connection exists, then the packet is rejected. Table 6 shows an example of a SYN packet of a new TCP connection.

When the above first packet is seen by the firewall, the firewall matches it against the set of filtering rules. Since Rule 1 of Table 2 allows the packet across, the firewall inserts a new entry into the state cache, and the TCP connection state is SYN_RCVD, as shown in Table 7.

Once the three-way handshake process is completed, the TCP connection state transits to the ESTABLISHED state, as shown in Table 8. When the TCP connection is terminated, the firewall removes the cache entry, thereby blocking the connection. Typically, the firewall also has a timeout value; if a TCP connection becomes inactive for too long, the firewall evicts the entry from the cache and blocks the connection.

IV. Stateful UDP Packet Filtering

The tracking of UDP session state is a complicated process, since UDP is a connectionless transport protocol, and unlike TCP, has no sequence numbers or flags (such as the six TCP flags: SYN, ACK, FIN, PSH, URG, and FIN). The only items on which a tracking process can use are the IP addresses and port numbers of the client and server involved in an UDP session.

In addition, UDP has no mechanism that announces a session's end. Consequently, UDP session's state table entries have to be cleared after reaching a predefined timeout value. Otherwise, malicious user may exploit this limitation in UDP protocol to fill in the UDP session state table with fake sessions, resulting in creating a DoS attack situation.

On the other hand, UDP protocol relies entirely on ICMP as its error handler. Therefore, ICMP protocol is an important part of an UDP session to be considered when tracking its overall state. For example, in an UDP session, the client or server host may not have sufficient buffer space to process the receiving packets. Consequently, the host may become unable to keep up with the speed at which it is receiving packets. In such a situation, the receiving host can send an ICMP source quench message (Type = 4, Code =0) which requests that the sender host decreases the rate of the sent packets. However, if the firewall blocks the ICMP source quench message because it is not part of the normal UDP session, the host that is sending packets too quickly does not know that an issue has come up, and it continues to send at the same speed, resulting in lost packets, or creating a DoS attack situation at the receiving host (Figure 2). Therefore, a stateful firewall that tracks UDP session state must consider such related ICMP traffic when deciding what traffic should be returned to protected hosts.

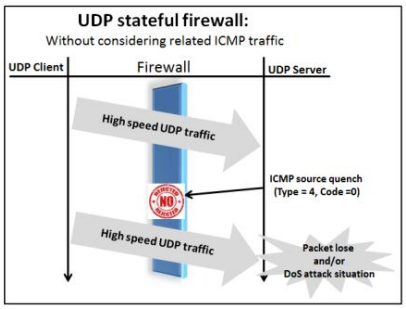

Figure 2. The potential effect of not considering related ICMP traffic in UDP sessions

V. Stateful ICMP Packet Filtering

ICMP is an error reporting and diagnostic protocol and is considered a required part of any IP implementation [10]. There are two types of ICMP packets, namely error-reporting and control packets. ICMP error-reporting packets are used to return error messages, and involve one-way communications, such as the ICMP source quench message. However, ICMP control packets are used by hosts to send request messages and receive back corresponding reply messages, such as the ICMP echo and reply messages (known as the Ping command). Hence, ICMP error messages involve two-ways communications, or request/reply-type messages. ICMP, like UDP, isn't a stateful protocol. However, like UDP, it also has attributes that allow its connections to be tracked. The ICMP attributes are usually the Type, Code, Identifier and Sequence number fields in the ICMP header.

The tracking of ICMP traffic that involves one-way communication is complicated, since ICMP error messages are precipitated by request by other protocols (TCP, UDP). Because of this multiprotocol issue, figuring ICMP messages into the state of an existing UDP or TCP session can be confusing and difficult to manage.

However, ICMP sessions that involve two-ways communications are less complicated to track, since for each ICMP response message, there should be an ICMP request message that has been sent before. That is, an ICMP session is tracked based on the source/destination addresses, Type, Code, Identifier and Sequence number of the request and reply messages. In an ICMP session, the Identifier, Sequence number and Data fields should be returned to the sender unaltered. The Identifier and Sequence number may be used by the echo request sender to aid in matching the replies with the echo requests. The Identifier might be used like a port in TCP or UDP to identify a session, and the Sequence number might be incremented on each echo request sent. The echoing node returns these same values in the echo reply. This tracking method is about the only way ICMP can enter into state table.

For example, after receiving the ICMP echo request packet shown in Table 9, the stateful firewall creates a new entry in its ICMP session's cache as shown in Table 10. Therefore, the ICMP echo reply packet of Table 11 is accepted by the stateful firewall, since it includes the same attributes values as the ICMP echo request packet. However, the fake ICMP echo reply packet of Table 12 is rejected, since there has been no ICMP echo request message including the same attribute values.

Another issue with ICMP is that, like UDP, it is connectionless; therefore, it must base the retention of a state table entry on a predetermined timeout because ICMP also does not have a specific mechanism to end its communication sessions.

VI. Experiment

This experiment describes steps to identify whether the Cisco ASA 5520 Adaptive Security Appliance (firewall) [12] offers stateful or stateless TCP and ICMP packet filtering. The experiment does not cover UDP traffic, since the tracking of UDP session state is a complicated process, and is not straightforward to implement. The experiment's steps can be used to test any other firewall device or software.

A. Network Architecture

Figure 3 shows the network architecture used in the experiment. We assume that a host (Host#1) with IP address 192.168.2.20 and a host (Host#2) with IP address 192.168.3.30 are connected to the GigabitEthernet0/0 and GigabitEthernet0/1 interfaces of the Cisco ASA 5520, respectively. We assume also that:

- Host#1 is a Web client;
- Host#2 is a Web server. *LiteServer* [13] is the Web server software;
- Both hosts use CommView sniffer [14] to capture the exchanged network traffic.

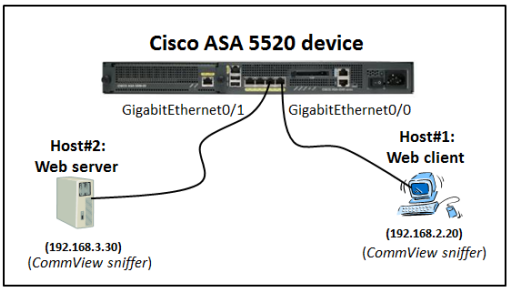

Figure 3. Network architecture

B. Experiment Steps

The experiment includes two parts.

1. Part 1: Stateful TCP packet filtering testing

The following are the steps of the experiment:

1. First, using the GUI interface of the Cisco ASA 5520, two filtering rules are implemented to allow standard web traffic (TCP/80) between the Web client host (Host#1) and the Web server host (Host#2).
2. Then, from Host#1, a Web browser is used to connect to the Web server at Host#2;
3. At Host#1, CommView sniffer is used to capture the three-way handshake TCP packets of the Web session,

as shown in Figure 4. Table 13 shows the values of the main fields, characterizing the Web session, of the captured three-way handshake packets.

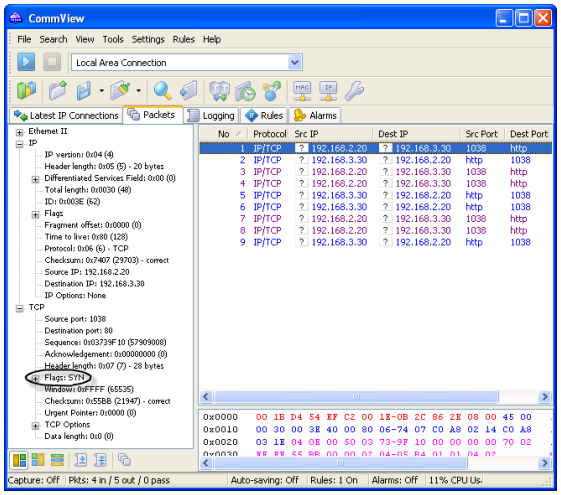

Figure 4. Three-way handshake TCP packets of a Web session

4. Then, CommView Visual Packet Builder [14] is used to send from Host #1 to Host #2 a fake TCP packet pretending that a TCP connection on port 80 is already established (SYN = 0 and ACK = 1). The fake TCP packet includes a source port different from the source port of the current active Web session, as shown in Table 14. Figure 5 shows the fields of the above generated fake TCP packet built using CommView Visual Packet Builder.

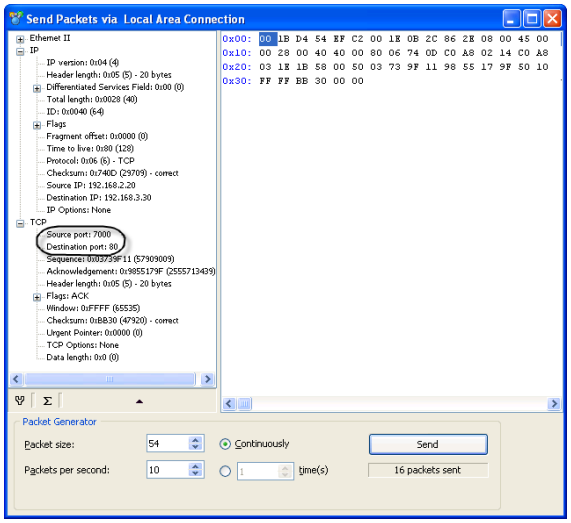

Figure 5. The fields of the fake TCP packet

5. At Host #2, CommView sniffer did not capture the generated fake TCP packet. This is because the fake TCP packet has been blocked by the Cisco ASA 5520. Consequently, the Cisco ASA 5520 is a stateful firewall for TCP related traffic, since it denies TCP packets that do not belong to established TCP sessions. It is important to indicate that if CommView sniffer was able to capture the fake TCP packet at the Web server host, then the Cisco ASA 5520 would be a stateless firewall.

2. Part 2: Stateful ICMP packet filtering testing

The following are the steps of the experiment:

1. First, to allow Host#1 to ping Host#2, two filtering rules are implemented using the GUI interface of the Cisco ASA 5520.
2. Then, Host#1 pings Host#2.
3. At Host#1, CommView sniffer is used to capture the exchanged ICMP packets. Table 15 shows the values of the main fields, characterizing the generated ICMP traffic by the Ping command.
4. Then, CommView Visual Packet Builder is used to send from Host #2 to Host #1 a fake ICMP echo reply packet, pretending that an ICMP echo request packet has been received before from Host#1. The fake ICMP echo reply packet includes different Identifier and Sequence number, as shown in Table 16.
5. At Host #1, CommView sniffer succeeded to capture the fake ICMP echo reply packet. Consequently, the Cisco ASA 5520 is a stateless firewall for ICMP related traffic, since it did not deny the fake ICMP echo reply packet. It is important to indicate that if CommView sniffer did not capture the fake ICMP packet at Host#1, then the Cisco ASA 5520 would be a stateful firewall for ICMP related traffic.

VII. Student's Performance and Satisfaction

From fall 2006 to spring 2008 (a two year period), students enrolled in the Network Border Control course (SECB358) course, at the Faculty of Information Technology, UAE University, were not offered hands-on lab exercises on stateless and stateful firewall packet filtering. Only the conceptual part of the topic has been described in the class. However, from fall 2008 to spring 2011 (a three year period), the students were offered the hands-on lab exercise described in this paper. Over the five year period, each semester the students were given two quizzes and a Midterm exam exercise about stateless and stateful packet filtering.

A. Quiz Example

As an example, the following is the contents of a quiz that has been given to students during Spring 2011:

We would like to test whether a firewall performs stateless or stateful inspection for TCP and ICMP traffic. We assume that:

- *The firewall filters traffic exchanged between two hosts (Host #1 and Host #2), as shown in Figure 6.*
- *The firewall allows Host #1 to access any Web server running on Host #2.*
- *The firewall allows Host #1 to ping Host #2.*
- *The firewall does not allow Host #2 to ping Host #1.*
- *The default security policy is "Deny all".*

We assume also that 8 packets have generated and exchanged, as shown in Figure 6. By analysing the accepted and denied packets, (1) tell if the firewall performs stateless or stateful packet inspection for TCP traffic, and (2) tell if the firewall performs stateless or stateful packet inspection for ICMP traffic.

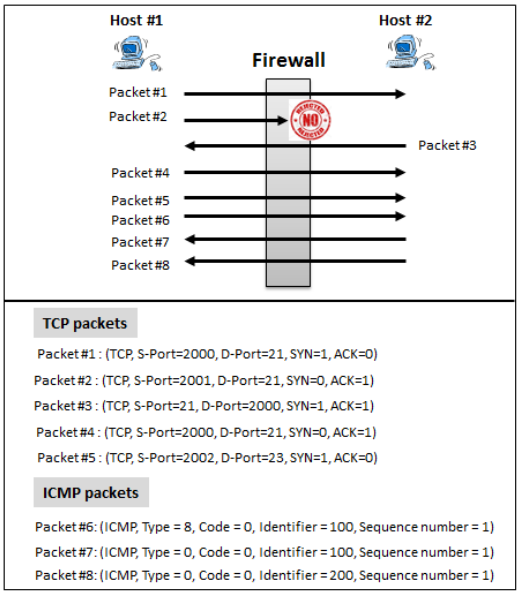

Figure 6. List of packets accepted and denied by the firewall

B. Students' Grading Performance

Figure 7 shows the students' total average grades per semester for the two quizzes and the mid-term exam exercise. It is clear that from fall 2008, the students' total average grade has started improving. This is mainly due to the fact that the hands-on lab exercise allowed students to better anatomize the concept of stateless and stateful packet filtering learned from the lecture. Through the five-year study, students who have participated in the lab exercises performed better on quizzes and mid-terms than those students who did not participate in the lab exercises.

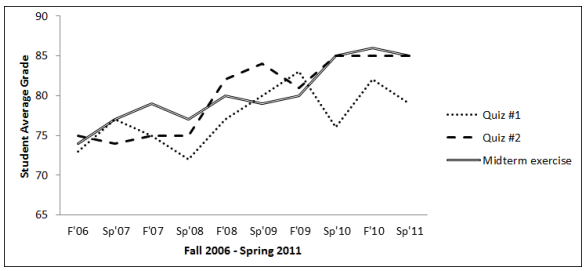

Figure 7. Student total average grades in the quizzes and the Midterm exam exercise

C. Students' Satisfaction

An anonymous questionnaire was administered to 110 students, who participated in the lab exercise, to measure their satisfaction level and collect their feedback regarding the discussed hands-on lab exercise. The results of the questionnaire showed that more than 87% of all students who answered the questionnaire believed the lab exercise to be useful and helped them better understand the underlying theoretical concepts associated with stateless and stateful packet filtering. The questionnaire also revealed that 90% of the students were interested in similar exercises in other network security classes, and 88% would strongly recommend the lab exercise to other students.

VIII. Conclusion

The importance of experimental learning has long been recognized in the learning theory literature. Students with practical skills will be better prepared to work as security administrators with better chances of landing jobs than students without these skills.

In this paper, we described a comprehensive hands-on lab exercise implementation about how to identify whether a given firewall performs stateless or stateful packet filtering. The learning objective of the exercise is for students to better anatomize the concept of stateless and stateful firewall packet filtering through examples and experiments in an isolated network laboratory. The lab exercise contributed to improve the students' grading performance. Students' satisfaction and feedback regarding the hands-on lab exercise were very positive.

IX. Acknowledgment

The author acknowledge the support of Emirates Foundation through Research Grant No. (2009/161).

X. References

[1] Dongqing Yuan, and Jiling Zhong. A lab implementation of TCP SYN flood attack and defense. SIGITE '08 Proceedings of the 9th ACM SIGITE Conference on

Information Technology Education (SIGITE '08), pp. 57-58, Cincinnati, Ohio, USA, October 16-18, 2008.

[2] Zouheir Trabelsi. Hands-on Lab Exercises Implementation of DoS and MiM Attacks using ARP Cache Poisoning. Proceedings of the 2011 Information Security Curriculum Development Conference, (InfoSecCD 2011), pp. 74-83, Kennesaw, GA, USA, September 30 - October 01, 2011.

[3] Zouheir Trabelsi. Switch's CAM Table Poisoning Attack: Hands-on Lab Exercises for Network Security Education. Proceedings of the Fourteenth Australasian Computing Education Conference (ACE2012), pp. 113-120, Melbourne, Australia, January 30 - February 3, 2012.

[4] Mike O'Leary. A Laboratory Based Capstone Course in Computer Security for Undergraduates. Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'06), Houston, Texas, USA, March 3–5, 2006.

[5] Paul J. Wagner, and Jason M. Wudi. Designing and Implementing a Cyberwar Laboratory Exercise for a Computer Security Course. ACM SIGCSE Bulletin, v.36 n. 1, March 2004.

[6] Xiaohong Yuan, David mattews, Omari Wright, Jinsheng Xu, and Huiming Yu. Laboratory exercises for wireless network attacks and defenses. Proceedings of the 14th Colloquium for Information Systems Security Education, pp. 115-123, Baltimore, Maryland, USA, June 7-9, 2010.

[7] J. Garrido and T Bandyopadhyay. Simulation Model Development in Information Security Education. Proceedings of the 2009 Information Security Curriculum Development Conference (InfoSecCD '09), Kennesaw, GA, USA, September 25-26, 2009.

[8] M. Ye and K. Sandrasegaran. Teaching about Firewall Concepts using the iNetwork Simulator. Proceedings of the 7th International Conference Information Technology Based Higher Education and Training (ITHET '06), Ultimo, NSW, USA, July 10-13, 2006.

[9] Kenneth Williams and Huiming Yu. An Interactive Firewall Simulator for Information Assurance Education. Proceedings of the World Congress on Engineering (WCE 2011), Vol I, London, U.K, July 6-8, 2011.

[10] Kevin R. Fall and W. Richard Stevens. TCP/IP Illustrated. Volume 1: The Protocols (2nd Edition), Addison-Wesley Professional, 2011.

[11] R. Tibbs, and E. Oakes. Firewalls and VPNs: Principles and Practices. Prentice Hall, 2006.

[12] http://www.cisco.com

[13] LiterServe software, http://www.cmfperception.com

[14] CommView tool, http://www.tamos.com.

Table 1. A filtering rule to allow HTTP traffic

Direction	Source IP	Destination IP	Protocol	Source port	Destination port	Action
Client-to-Server	*192.168.1.1/24*	*192.168.2.1/24*	TCP	Any	HTTP	Allow

Table 2. Firewall filtering rules for security policies SP#1

Rule	Direction	Source IP	Destination IP	Protocol	Source port	Destination port	SYN	ACK	Action
R1	Client-to-Server	*192.168.1.1/24*	*192.168.2.1/24*	TCP	Any	80	1	0	Allow
R2	Server-to-Client	*192.168.2.1/24*	*192.168.1.1/24*	TCP	80	Any	1	1	Allow
R3	Client-to-Server	*192.168.1.1/24*	*192.168.2.1/24*	TCP	Any	80	0	1	Allow
R4	Server-to-Client	*192.168.2.1/24*	*192.168.1.1/24*	TCP	80	Any	0	1	Allow

Table 3. Firewall filtering rules for security policies SP#2

Rule	Direction	Source IP	Destination IP	Protocol	Source port	Destination port	SYN	ACK	Action
R5	Server-to-Client	*192.168.2.1/24*	*192.168.1.1/24*	TCP	Any	Any	1	0	Deny

Table 4. Example of rejected TCP connection establishment request

Packet	Direction	Source IP	Destination IP	Protocol	Source port	Destination port	SYN	ACK
Packet #1	Server-to-Client	*192.168.2.20*	*192.168.1.10*	TCP	6000	80	1	0

Table 5. An example of malicious TCP packet

Packet	Direction	Source IP	Destination IP	Protocol	Source port	Destination port	SYN	ACK
Packet #2	Server-to-Client	*192.168.2.20*	*192.168.1.10*	TCP	80	9000	0	1

Table 6. An example of a first packet of a TCP connection (SYN packet)

Packet	Direction	Source IP	Destination IP	Protocol	Source port	Destination port	SYN	ACK
Packet #1	Client-to-Server	*192.168.1.5*	*192.168.2.10*	TCP	1200	80	1	0

Table 7. The state cache of established TCP connections after receiving the first packet

TCP connection	Client IP	Server IP	Client port	Server port	Connection State
Connection #1	*192.168.1.5*	*192.168.2.10*	1200	80	SYN_RCVD

Table 8. TCP connection state after completing the three-way handshake process

TCP connection	Client IP	Server IP	Client port	Server port	Connection State
Connection #1	*192.168.1.5*	*192.168.2.10*	1200	80	ESTABLISHED

Table 9. An ICMP echo request packet

Packet	Source IP	Destination IP	Type	Code	Identifier	Sequence number
Packet #1	*192.168.1.5*	*192.168.2.10*	8	0	200	1

Table 10. The firewall's cache of ICMP sessions

ICMP session	Source IP	Destination IP	Type	Code	Identifier	Sequence number	Session State
Session #1	*192.168.1.5*	*192.168.2.10*	1200	80	200	1	Request

Table 11. Allowed ICMP echo reply packet by the firewall

Packet	Source IP	Destination IP	Type	Code	Identifier	Sequence number
Packet #2	*192.168.2.10*	*192.168.1.5*	0	0	200	1

Table 12. Denied ICMP echo reply packet by the firewall

Packet	Source IP	Destination IP	Type	Code	Identifier	Sequence number
Packet #3	*192.168.2.10*	*192.168.1.5*	0	0	300	1

Table 13. The field values of the three-way handshake TCP packets of the Web session

Packet number as displayed in CommView sniffer	Source IP	Destination IP	Source port	Destination port	SYN	ACK
1	192.168.2.20	192.168.3.30	1038	80	1	0
2	192.168.3.30	192.168.2.20	80	1038	1	1
3	192.168.2.20	192.168.3.30	1038	80	0	1

Table 14. Fake TCP packet

Source IP	Destination IP	Source port	Destination port	SYN	ACK
192.168.2.20	192.168.3.30	**7000**	80	0	**1**

Table 15. The values of the main fields of the exchanged ICMP echo request and reply packets

Packet number as displayed in CommView	Source IP	Destination IP	Type	Code	Identifier	Sequence Number
1	192.168.2.20	192.168.3.30	8	0	512	9472
2	192.168.3.30	192.168.2.20	0	0	512	9472

Table 16. Fake ICMP echo reply packet

Source IP	Destination IP	Type	Code	Identifier	Sequence Number
192.168.3.30	192.168.2.20	0	0	512	8000