***LAB – 3 ASSIGNMENT***

***COURSE CODE (ET 2595): Network and System Security***

***Name:***

***Person nr:***

***Task-1:***

* I decided to reset Server A and Server B to their original default settings to make sure they were clean for upcoming tasks. This was done to undo any changes made to the servers in the past.
* By restoring the servers to their original state, we are giving Lab 03 a fresh start for any future activities. We've temporarily set Promiscuous Mode to allow everyone to participate, ensuring a safe environment.
* This precautionary measure is taken because there are signs that Server B might have aggressive intentions, like trying to attack Server A or conducting reconnaissance using Metasploit.
* To address this potential threat, we're implementing the Snort IDS System on Server A. This application serves as a vigilant guardian, continuously monitoring our system to detect any potential attacks or unusual activities. By having Snort in place, we enhance the safety and security of our system, providing robust defenses against potential threats and ensuring a protected environment for our operations.

A computer screen shot of a network

Description automatically generated

A computer screen shot of a network

Description automatically generated

To set up Snort on our machine, we'll begin by executing the command "apt install snort." Once Snort is installed, it's crucial to configure it appropriately for our needs. Specifically, we need to specify the IP address 192.168.70.5 associated with the interface. This step ensures that Snort can effectively monitor and identify potential security threats by analyzing the traffic on that particular interface.

A close-up of a text

Description automatically generated

The visual representation of this test, including the captured packets between Servers A and B, is provided in the section below the reference figures. This approach allows us to gain insights into the communication flow and detect any anomalies or potential security issues between the servers.

A screenshot of a computer

Description automatically generated

After gathering the data packets generated during our ping test, I carefully analyzed them with Wireshark. My goal was to spot ICMP echo and reply packets resulting from the ping test. Upon collecting and scrutinizing the packets, our analysis proved successful, as depicted in the accompanying image.

This thorough examination of the collected packets using Wireshark allowed us to pinpoint and distinguish the ICMP echo and reply to packets that were generated during the ping test. The success of our analysis is clearly illustrated in the image on the right.

A screenshot of a computer

Description automatically generated

***Task-2:***

Once the connection is established successfully, it becomes crucial to make adjustments to the Snort file. To achieve this, add a "#" symbol at the end of the line corresponding to the rules for ICMP notification in the "icmp-info.rules" file. This action comments out the rule and prevents Snort from sending ICMP notifications.

To disable the ICMP notifications, it's essential to follow this procedure. The modification mentioned in the rule description involves adding a new local rule to the "local.rules" file. This new local rule will be included in the list of local rules.

The additional regulations were incorporated as follows:

A computer screen shot of a computer screen

Description automatically generated

On Server A, the newly implemented Snort rule has been successful in detecting ICMP packets when Server B sends ping messages to it. This marks a significant achievement that aligns to what has been described in the reference.

***Task 3***

I began the Metasploit software by executing the command "sudo msfconsole." Subsequently, I proceeded to configure the environment using the following commands:

1. \*\*use auxiliary/scanner/portscan/syn:\*\* This command activates the port scanning module, allowing for the identification of open ports on the target system.
2. \*\*set RHOSTS 192.168.70.5:\*\* By setting the remote host to 192.168.70.5, I directed the actions within Metasploit towards Server A, as indicated in the reference.
3. \*\*set INTERFACE enp0s8:\*\* Configuring the interface to use the IP address 192.168.70.6 ensures proper network connectivity with Server A.
4. \*\*set PORTS 1-500:\*\* To streamline the scanning process, I specified the scanning of ports ranging from 1 to 500. This targeted approach enhances efficiency in identifying potential vulnerabilities.

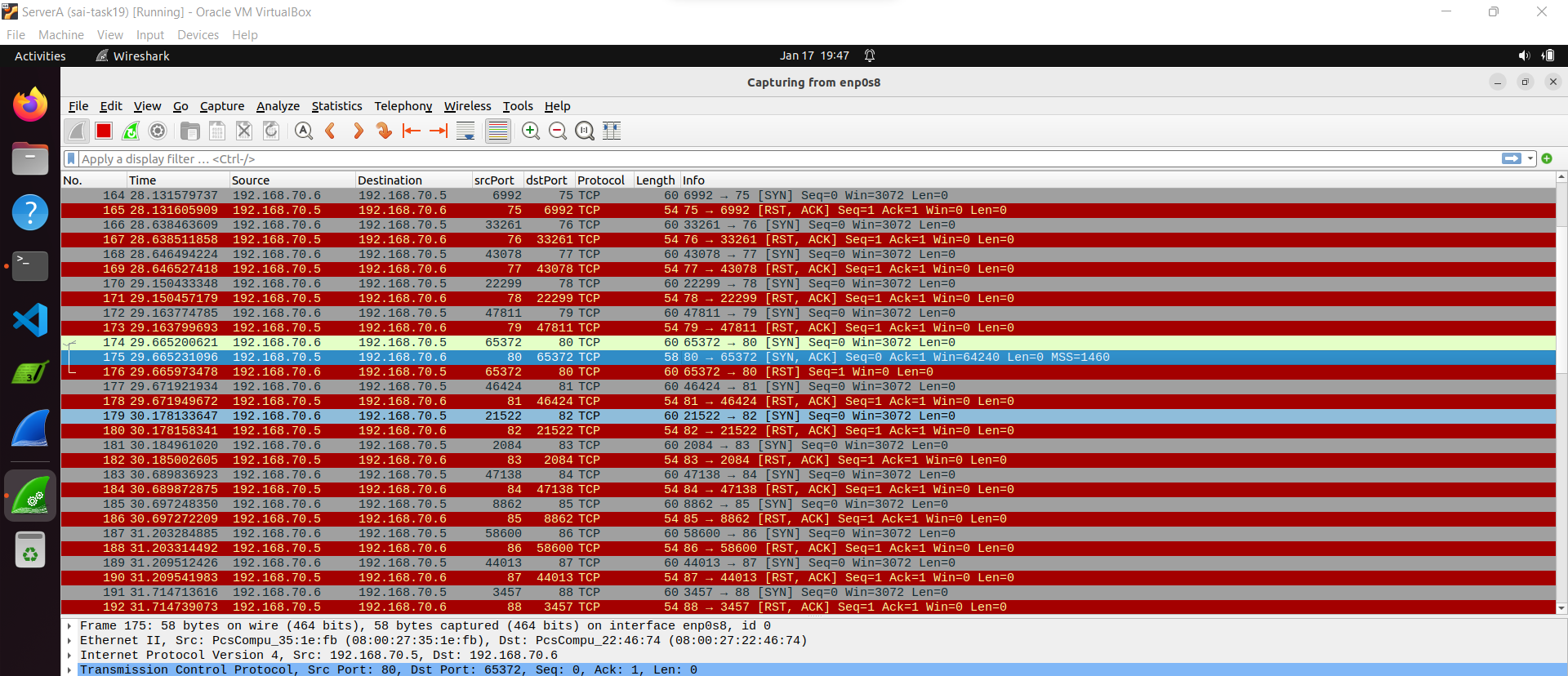
These commands collectively establish a structured environment within Metasploit, aligning with the provided reference. This setup facilitates effective port scanning and exploration of Server A for potential security weaknesses.

A screenshot of a computer

Description automatically generated

The picture indicates that only two ports are accessible in the range of 1 to 500 – specifically, PORT 80 and PORT 22, 443. Below is a list of the scanned ports, and the associated numbers for the open ports are presented alongside the outcomes.

In the image below, you can observe the Wireshark terminal displaying network traffic data captured by Server A. Server A has two accessible ports: 80, 22, and 443. For communication on port 80, Server B (65372) sends SYN packets. Server A responds with [SYN, ACK] packets to confirm acceptance. This step is essential in identifying open ports during a scan. Server B receives an RST reset response if a port is closed, prompting a third handshake initiation.



**Telnet traffic capture for Port 23 by Wireshark**

A close up of a text

Description automatically generatedA screenshot of a computer

Description automatically generated

**Telnet traffic capture for Port 22 by Wireshark**

Server B tried to connect to port 22 instead of port 23 by entering the command "telnet 192.168.70.5 22" in its terminal. It was important to specify the port number in the command. Fortunately, an open port 22 was found on the local network, enabling the successful establishment of the connection. In the figure below, Wireshark shows the communication between Servers A and B in the console. The [SYN, ACK] response from Server A confirms the successful communication with Server B over port 22.

**A screenshot of a computer

Description automatically generated**

**SSH traffic capture by Wireshark:**

A close-up of a text

Description automatically generated

A screenshot of a computer

Description automatically generated

The Snort rule provided is designed to detect port scanning activities. Here's a breakdown of the rule:

***A close-up of a document

Description automatically generated***

For the network administrator to identify port scanning activities on the network, the packet must include a set SYN flag and no fragmented bits. You can refer to the citation below for more information on this rule.

A screenshot of a computer

Description automatically generated

***Task 4***

To identify a Denial of Service (DoS) attack in this scenario, a Snort rule was crafted. Initially, Server B's Metasploit was directed to "*auxiliary/dos/tcp/synflood*" and set up with the external host and interface. The rule is explicitly focused on port 80, as indicated in Figure.

The Snort rule reads as follows:



In Task 3, we learned that Snort focuses on detecting port scanning packets, excluding benign traffic. Now, let's delve into additional aspects not previously discussed:

1. \*\*threshold:type both:\*\* The "type" specified in the threshold is set to "both." This implies that the threshold applies to both the source and destination, ensuring a comprehensive assessment of network traffic.
2. \*\*threshold:track by dst:\*\* The threshold is configured to "track by destination." This means that the counting mechanism is specifically monitoring events based on the destination address, providing a targeted approach.
3. \*\*threshold:count 10:\*\* This parameter dictates the number of events considered by the threshold. In this case, the threshold is triggered when there are 10 occurrences of the specified event, contributing to the identification of potential DoS attacks.
4. \*\*threshold:seconds 10:\*\* The time aspect of the threshold is set to "seconds 10." This signifies the duration over which the count is accumulated. In this rule, if there are 10 occurrences of the event within a time window of 10 seconds, the alert is triggered, aiding in the timely detection of potential DoS attacks.

A screenshot of a computer

Description automatically generated

Upon implementing the aforementioned Snort rule, the Snort system underwent a re-execution. In this process, it successfully identified and flagged the DoS SYN packets. My attempts to test benign traffic, however, revealed that Snort did not detect the benign traffic but specifically identified and reported only the SYN packets. The reference for this observation is provided below.

A screenshot of a computer

Description automatically generated

Wireshark captured the network packets during a SYN flood orchestrated by Metasploit, specifically targeting Port 80 on the IP address 192.168.70.5. The figure below provides a visual representation of the observed malicious traffic..

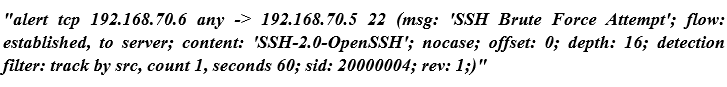
A screenshot of a computer

Description automatically generated

***Task 5***

For this, a Snort rule was crafted to identify rogue SSH connections. Initially, Server B's Metasploit was directed to "auxiliary/scanner/ssh/ssh login," with the remote host set. The reference for this is shown in Figure.

The added Snort rule is as follows:



Now, let's explore the elements that haven't been discussed before:

1. flow:established: This ensures that the rule applies only to established TCP connections, focusing on traffic flow in specific directions.
2. flow:to server: The rule is set to trigger on Server B requests from the IP address 192.168.70.6.
3. content: "SSH-2.0-OpenSSH": The rule detects packets containing the string "SSH-2.0-OpenSSH."
4. nocase: This command ignores the case of the string, making the rule case-insensitive.
5. offset:0: By setting the offset to 0, the Snort rule starts matching the pattern from the beginning of the payload.
6. depth:16:\*\* With a depth of 16 (considering the string size is 15), the rule matches patterns up to 16 bytes of payload (string).
7. detection filter:track by src: Detection is based on the source IP address.
8. detection filter:count 1: The rule triggers after one match within 60 seconds.
9. detection filter:seconds 60: This indicates the time window over which the count is accumulated.
10. rev:1: The revision number is a unique identifier for the Snort rule, distinguishing different versions.

Following the creation of the mentioned Snort rule, the local.rules file was updated. Subsequently, Metasploit was utilized with a login information file to initiate a brute force attack. The reference for this process is provided below.

A screenshot of a computer screen

Description automatically generated

The Snort system successfully identified and alerted on all unauthorized SSH connections during the attempted brute force. It's noteworthy that the benign traffic attempting to connect did not trigger any detection from Snort. The reference for this observation is also outlined below.

A screenshot of a computer

Description automatically generated