ACO331 Project 1 – Jack Sharkey

Both computers were connected to my local network at home. I have written a python script to parse the text file, I have pasted the code at the end of this document.

Attacking Computer (Computer A):

This is the VM running on my laptop with bridged adapter setting.

IP Adress: 192.168.0.107

Scanned ports 1-5000 using: “nmap -Pn -p 1-5000 192.168.0.202”

Victim Computer (Computer B):

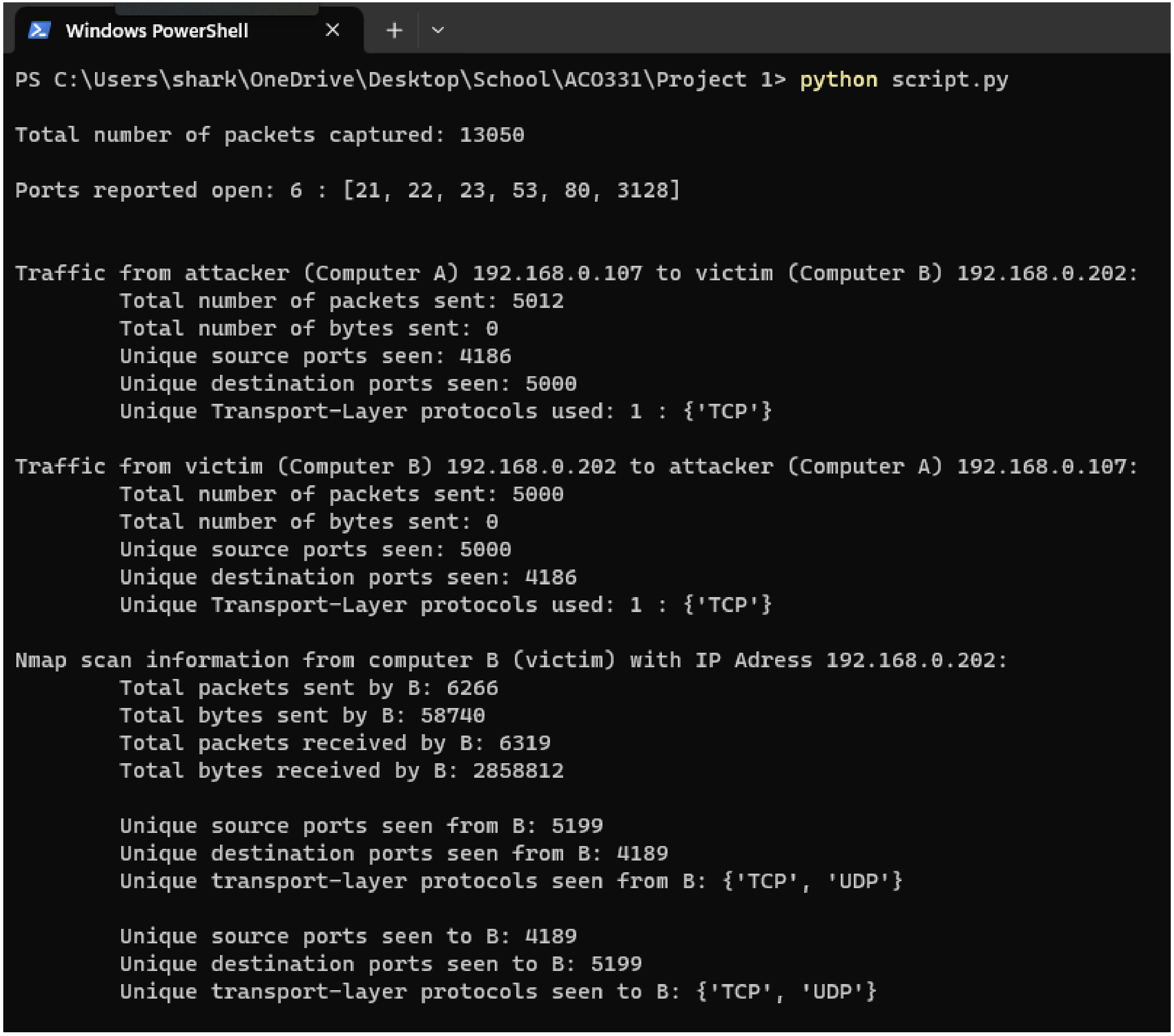
This is the VM running on my desktop with bridged adapter setting.

IP Adress: 192.168.0.202

Captured traffic using command: “sudo tcpdump -i any -n > nmap.scan3.txt”

Sites visited: Google.com, YouTube.com, Facebook.com, Reddit.com, Wikipedia.com, cnn.com, wsj.com, x.com, asu.edu

I wrote a python script to parse the text file and answer the questions. Here is the output:



```
PS C:\Users\shark\OneDrive\Desktop\School\ACO331\Project 1> python script.py

Total number of packets captured: 13050

Ports reported open: 6 : [21, 22, 23, 53, 80, 3128]


Traffic from attacker (Computer A) 192.168.0.107 to victim (Computer B) 192.168.0.202:
        Total number of packets sent: 5012
        Total number of bytes sent: 0
        Unique source ports seen: 4186
        Unique destination ports seen: 5000
        Unique Transport-Layer protocols used: 1 : {'TCP'}

Traffic from victim (Computer B) 192.168.0.202 to attacker (Computer A) 192.168.0.107:
        Total number of packets sent: 5000
        Total number of bytes sent: 0
        Unique source ports seen: 5000
        Unique destination ports seen: 4186
        Unique Transport-Layer protocols used: 1 : {'TCP'}

Nmap scan information from computer B (victim) with IP Adress 192.168.0.202:
        Total packets sent by B: 6266
        Total bytes sent by B: 58740
        Total packets received by B: 6319
        Total bytes received by B: 2858812

        Unique source ports seen from B: 5199
        Unique destination ports seen from B: 4189
        Unique transport-layer protocols seen from B: {'TCP', 'UDP'}

        Unique source ports seen to B: 4189
        Unique destination ports seen to B: 5199
        Unique transport-layer protocols seen to B: {'TCP', 'UDP'}
```

This is the script I wrote to parse the file:

```
def main():
    a_ip = '192.168.0.107'    # IP address of the attacker (Computer A)
    v_ip = '192.168.0.202'    # IP address of the victim (Computer B)
    file = 'nmap.scan3.txt'   # Nmap scan file
    getInformation(a_ip, v_ip, file)


def getInformation(attacker_ip, victim_ip, scan_file):
    # A to B Traffic
    A_to_B_packet_count = 0        # Count total packets sent
    A_to_B_byte_count = 0          # Count total bytes sent
    A_to_B_unq_src_port = set()    # Count total unique source ports sent
    A_to_B_unq_dst_port = set()    # Count total unique destination ports sent
    A_to_B_transport_protocols = set()  # Count total unique transport-layer protocols used


    # B to A Traffic
    B_to_A_packet_count = 0        # Counting total packets sent
    B_to_A_byte_count = 0          # Counting total bytes sent
    B_to_A_unq_src_port = set()    # Count total unique source ports sent
    B_to_A_unq_dst_port = set()    # Count total unique destination ports sent
    B_to_A_transport_protocols = set()  # Count total unique transport-layer protocols used


    opened_ports = set()           #Count number of open ports


    #B Traffic
    B_packet_sent_count = 0        #Count total number of packets sent from B
    B_byte_sent_count = 0          #Count total number of bytes sent from B
    B_packet_received_count= 0     #Count total number of packets received by B
    B_byte_received_count = 0      #Count total number of bytes received by B
    B_tp_uniq_from = set()         #Count unique transport-layer protocols sent
    B_tp_uniq_to = set()           #Count unique TLP received
    B_uniq_src_prt_sent = set()    #Count number of unique ports from B
    B_uniq_dst_prt_sent = set()    #Count number of unique ports from B
    B_uniq_src_prt_received = set() #Count number of unique ports to B
    B_uniq_dst_prt_received = set() #Count number of unique ports to B
    total_packet_count = 0          #Count total number of packets captured


    with open(scan_file, 'r') as file:
        for line in file:
            total_packet_count += 1  # Increment total packet count for each line
            parts = line.split()     # Split current line into accessible parts


            # If True there is a source and desitination IPv4 Adress in current line
            if len(parts) >= 5 and '.' in parts[2]:
                src_ip_parts = parts[2][::-1].split('.', 1)  # Split source IP address and port
number
                dst_ip_parts = parts[4][::-1].split('.', 1)  # Split destination IP address and port
number


                src_ip = src_ip_parts[1][::-1]                      # Extracting only the IP address
part from the end
                src_port = int(src_ip_parts[0][::-1].rstrip(':'))   # Extracting the port number and
converting to integer


                dst_ip = dst_ip_parts[1][::-1]                      # Extracting only the IP address
part from the end
                dst_port = int(dst_ip_parts[0][::-1].rstrip(':'))   # Extracting the port number and
converting to integer




                # If True, Packet was sent from A to B
                if src_ip == attacker_ip and dst_ip == victim_ip:
                    A_to_B_packet_count += 1               # Increment packet count
                    A_to_B_unq_src_port.add(src_port)       # Counting unique source ports
                    A_to_B_unq_dst_port.add(dst_port)       # Counting unique destination
ports
```

```
    #Add total bytes sent if the payload is not zero
    payload_length = int(parts[-1])
    if payload_length != 0:
        A_to_B_byte_count += payload_length



    #Check Transport-Layer Protocol
    if len(parts) >= 6 and parts[5] == 'Flags':
        A_to_B_transport_protocols.add('TCP')
    else:
        A_to_B_transport_protocols.add('UDP')

# If True, Packet was sent from B to A
elif src_ip == victim_ip and dst_ip == attacker_ip:

    #Check if sequence number is greater than 0, if so, the port is open
    if int(parts[8].rstrip(',')) > 0:
        opened_ports.add(src_port)

    B_to_A_packet_count += 1              # Increment packet count
    B_to_A_unq_src_port.add(src_port)        # Counting unique source ports
    B_to_A_unq_dst_port.add(dst_port)        # Counting unique destination ports

    #Add total bytes sent if the payload is not zero
    payload_length = int(parts[-1])
    if payload_length != 0:
        B_to_A_byte_count += payload_length
```

```
            #Check Transport-Layer Protocol
            if len(parts) >= 6 and parts[5] == 'Flags':
                B_to_A_transport_protocols.add('TCP')
            else:
                B_to_A_transport_protocols.add('UDP')

        # If True, Packet was sent from B
        if src_ip == victim_ip:
            B_packet_sent_count += 1              #Increment Number of packets sent by
B
            B_uniq_src_prt_sent.add(src_port)     #Add port to set to check if unique
            B_uniq_dst_prt_sent.add(dst_port)     #Add port to set to check if unique

            #Check Transport-Layer Protocol
            if len(parts) >= 6 and parts[5] == 'Flags':
                B_tp_uniq_from.add('TCP')

                #Extract Payload Length
                if parts[-1].isdigit():
                    B_byte_sent_count += int(parts[-1])
                else:
                    #Handle HTTP Packets
                    B_byte_sent_count += int(parts[20].strip(':'))
            else:
                B_tp_uniq_from.add('UDP')

                #Extract Payload Length
```

```
                        B_byte_sent_count += int(parts[-1].strip('()')) - 28 #Subtract UDP and IP header size for payload size


                #If True, Packet was received by B
                elif dst_ip == victim_ip:
                    B_packet_received_count += 1           #Increment number of packets received by B
                    B_uniq_src_prt_received.add(src_port)  #Add port to set to check if unique
                    B_uniq_dst_prt_received.add(dst_port)  #Add port to set to check if unique


                    #Check Transport-Layer Protocol
                    if len(parts) >= 6 and parts[5] == 'Flags':
                        B_tp_uniq_to.add('TCP')


                        #Extract Payload Length
                        if parts[-1].isdigit():
                            B_byte_received_count += int(parts[-1])
                        else:
                            B_byte_received_count += int(parts[20].strip(':'))
                    else:
                        B_tp_uniq_to.add('UDP')


                        #Extract Payload Length
                        B_byte_received_count += int(parts[-1].strip('()')) - 28



    # Output results


    # Total Number of packets captured
    print(f'\nTotal number of packets captured: {total_packet_count}\n')
    # Number of ports reported open by nmap
    print(f'Ports reported open: {len(opened_ports)} : {sorted(opened_ports)}\n')


    # A to B
    print(f'\nTraffic from attacker (Computer A) {attacker_ip} to victim (Computer B)
{victim_ip}:')
    #  Total data packets
    print(f"\tTotal number of packets sent: {A_to_B_packet_count}")


    #  Total bytes
    print(f"\tTotal number of bytes sent: {A_to_B_byte_count}")


    #  Total unique source ports
    print(f"\tUnique source ports seen: {len(A_to_B_unq_src_port)}")


    #  Total unique destination ports
    print(f"\tUnique destination ports seen: {len(A_to_B_unq_dst_port)}")


    #  Number unique Transport-Layer protocols used
    print(f"\tUnique Transport-Layer protocols used: {len(A_to_B_transport_protocols)} :
{A_to_B_transport_protocols}\n")


    # B to A
    print(f'Traffic from victim (Computer B) {victim_ip} to attacker (Computer A)
{attacker_ip}:')


    #  Total packets
    print(f"\tTotal number of packets sent: {B_to_A_packet_count}")
    #  Total bytes
    print(f"\tTotal number of bytes sent: {B_to_A_byte_count}")

    #  Total unique source ports
    print(f"\tUnique source ports seen: {len(B_to_A_unq_src_port)}")

    #  Total unique destination ports
    print(f"\tUnique destination ports seen: {len(B_to_A_unq_dst_port)}")

    #  Number unique Transport-Layer protocols used
    print(f"\tUnique Transport-Layer protocols used: {len(B_to_A_transport_protocols)} : {B_to_A_transport_protocols}\n")

    # B Traffic
    print(f'Nmap scan information from computer B (victim) with IP Adress {victim_ip}:')
    #  Total packets sent by B
    print(f'\tTotal packets sent by B: {B_packet_sent_count}')

    #  Total bytes sent by B
    print(f'\tTotal bytes sent by B: {B_byte_sent_count}')

    #  Total packets received by B
    print(f'\tTotal packets received by B: {B_packet_received_count}')

    #  Total bytes received by B
    print(f'\tTotal bytes received by B: {B_byte_received_count}\n')

    #  Total number of unique source ports seen from B
    print(f'\tUnique source ports seen from B: {len(B_uniq_src_prt_sent)}')

    #  Total number of unique destination ports seen from B
    print(f'\tUnique destination ports seen from B: {len(B_uniq_dst_prt_sent)}')

    #  Total number of unique transport-layer protocols seen from B
    print(f'\tUnique transport-layer protocols seen from B: {B_tp_uniq_from}\n')

    #  Total number of unique source ports seen to B
    print(f'\tUnique source ports seen to B: {len(B_uniq_src_prt_received)}')

    #  Total number of unique destination ports seen to B
    print(f'\tUnique destination ports seen to B: {len(B_uniq_dst_prt_received)}')

    #  Total number of unique transport-layer protocols seen to B
    print(f'\tUnique transport-layer protocols seen to B: {B_tp_uniq_to}')
    print()


if __name__ == "__main__":
    main()
```