Class: CompSci 4C03
Student Name: Jatin Chowdhary
Student ID: 400033011
Date: March 15th, 2021

Assignment #4: MiniNet

4.1 – Task 1: MiniNet Test

1.

Yes, the controller does act as a HUB. The IP address of the *leg* node is 10.0.0.2, and the IP address of the *vic* node is 10.0.0.3. Pinging the *vic* node from the *leg* node causes the *leg* node to send 64 bytes of data every few milliseconds. This can be seen in the figure below.

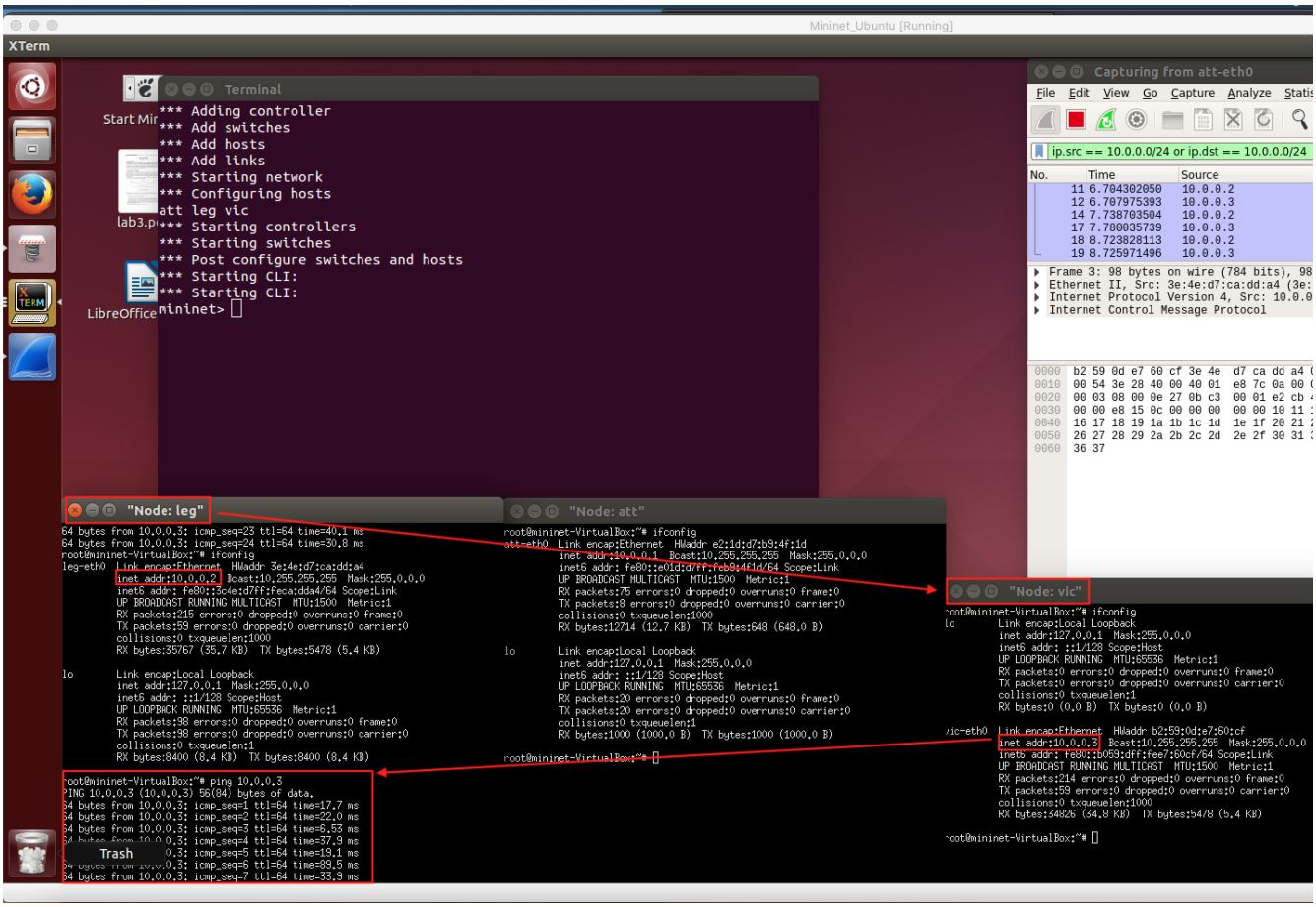

Figure 1: The boxes in red highlight the IP address of the 'vic' and 'leg' nodes. The red boxes also circle the ping command and its output, which can be found at the bottom left corner.

Yes, the ICMP packets can be sniffed by Wireshark running on the *att* node. The Wireshark packet-listing window shows that ICMP packets are being sent back and forth between 10.0.0.2 (*leg*) and 10.0.0.3 (*vic*). This can be seen in the figure below

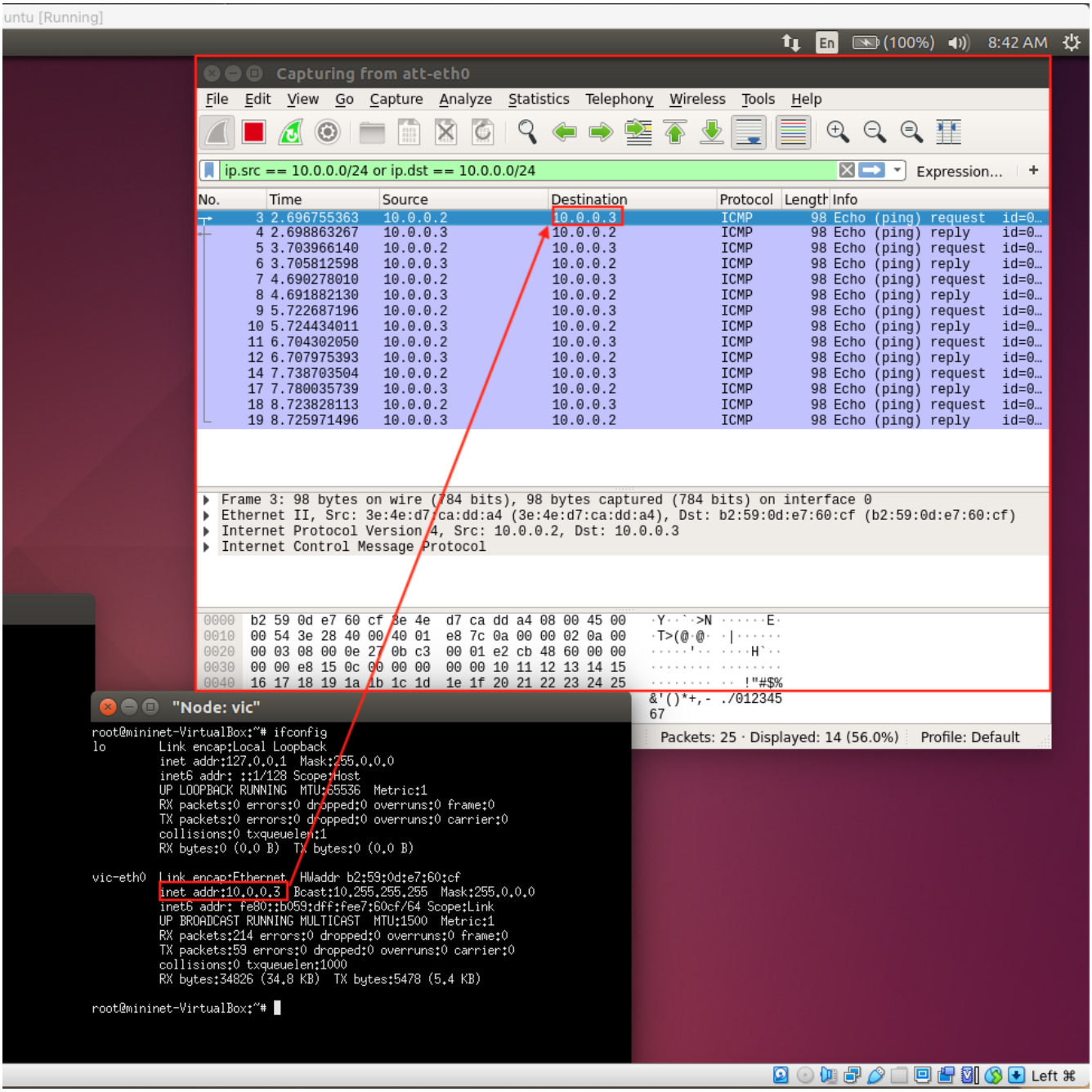

Figure 2: The entire Wireshark window is enclosed in a red box. Inside the packet-listing window of Wireshark, the first ICMP packet's destination is highlighted. This shows that packets are being sent to the 'vic' node. The IP address of the 'vic' node is highlighted at the bottom in red.

2.

Necessary services on the *vic* node are running. The services running are:

- HTTP
- SSH
- TELNET

This is verified by the output of the *netstat -a* command. This is accurate because the *vic* node is an imitation of a server, thus it needs to be running HTTP, SSH, and TELNET services. The other nodes, *leg* and *att*, are clients and do need to be running these services because they do not service requests; because they are not (web) servers. The output for *netstat -a* on the *vic* node can be seen in the figure below.

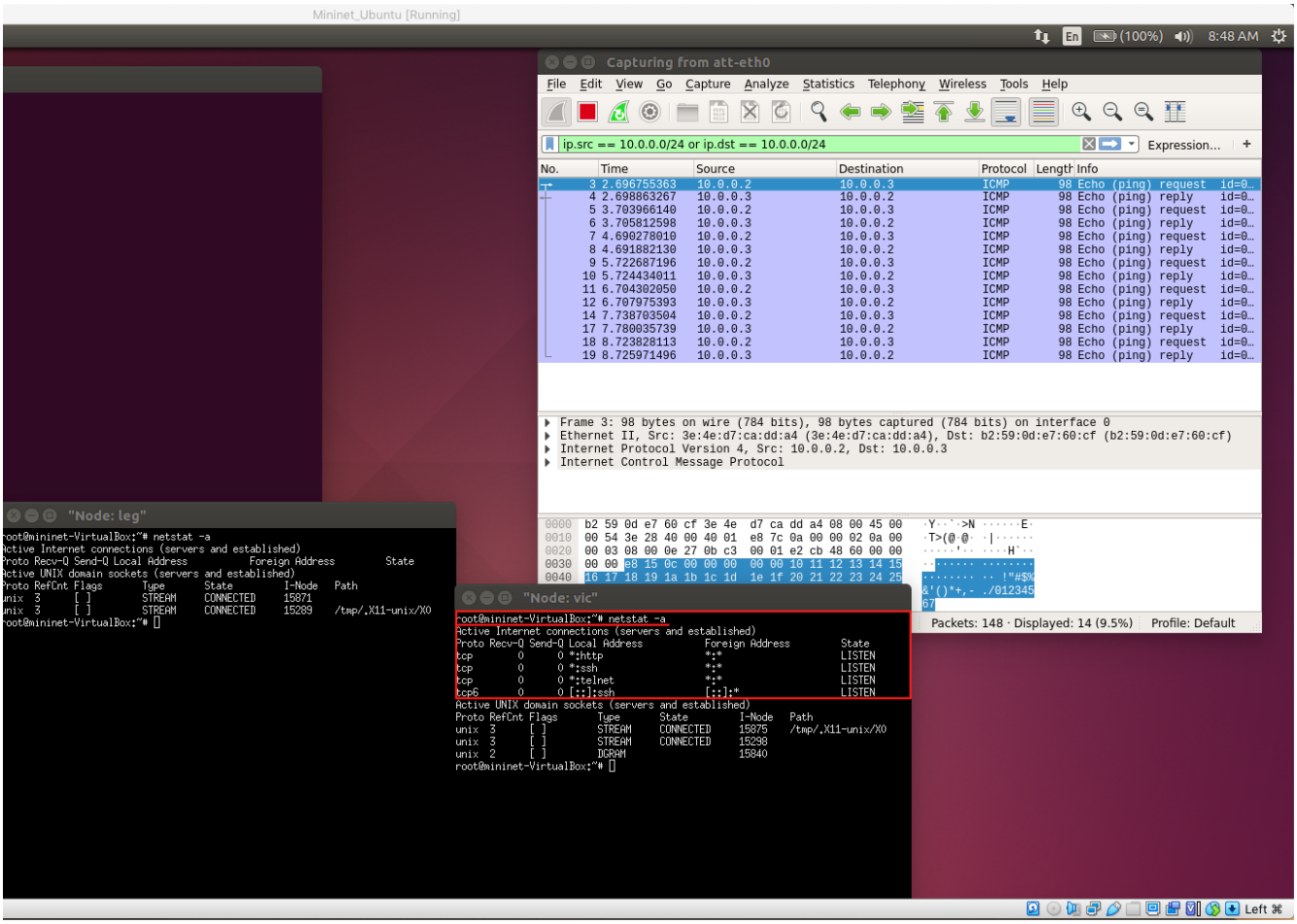

Figure 3: The red box in the bottom middle of the screen highlights the output of the 'netstat -a' command on the 'vic' node

3.

The *vic* node is available to service requests. It returns an *index.html* file when the *leg* node requests it via the *wget* command. The HTTP packets corresponding to this request are captured by Wireshark and can be seen in the packet-listing window. This is demonstrated in the figure below.

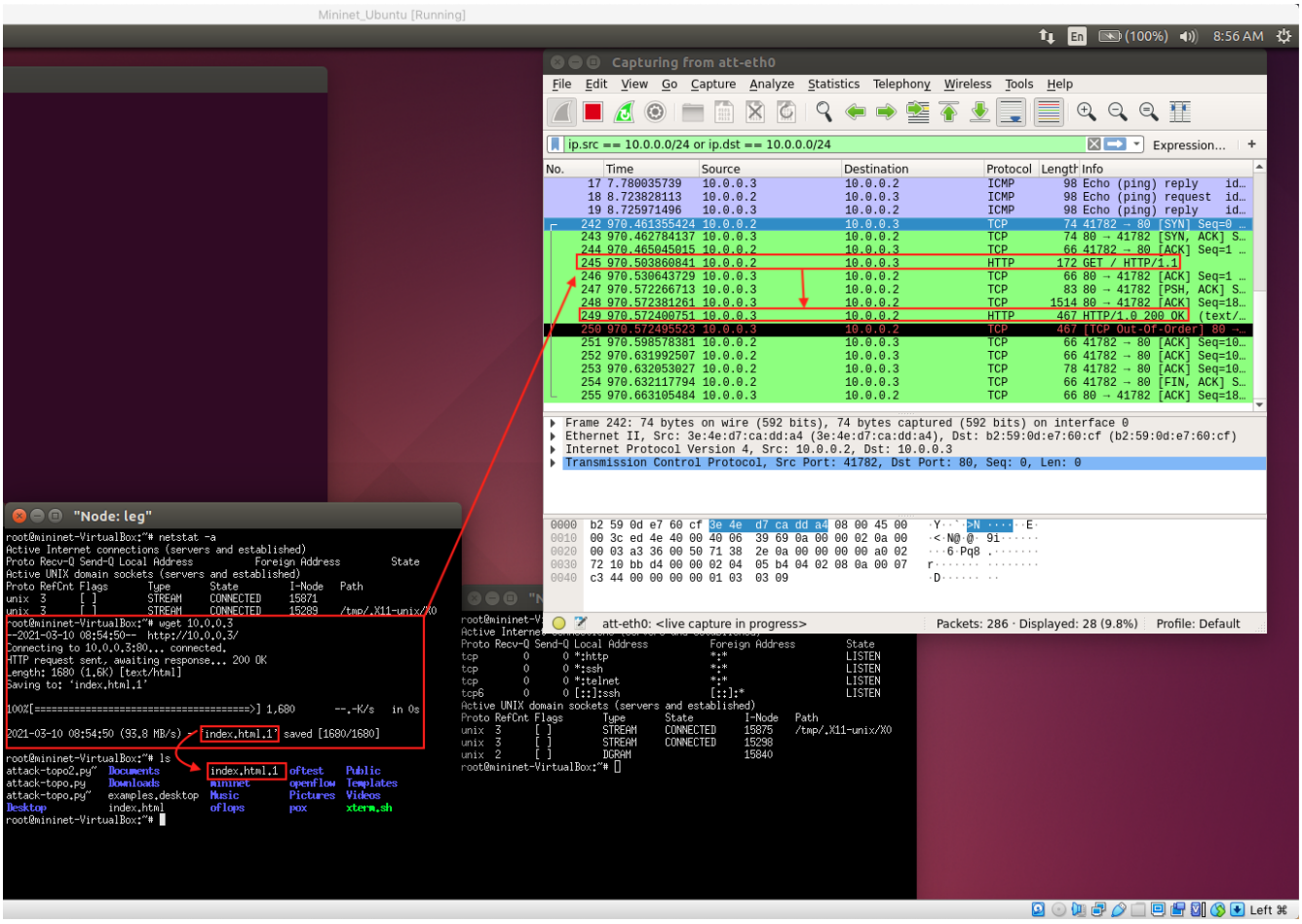

Figure 4: The 'leg' node is able to retrieve an HTML file from the 'vic' node using the 'wget' command. This can be seen in the red box on the bottom left of the figure. Furthermore, the captured HTTP packets that correspond to the 'wget' request are highlighted, in red, in Wireshark on the right side of the figure. In Wireshark's packet-listing window, the HTTP GET and RESPONSE packets are highlighted in red.

4.2 – Task 2: SYN Flooding Attack

1. The *netwox* command and arguments used for this attack are:

- netwox 76 -i 10.0.0.3 -p 80
  - netwox 76
    - Specifies the type of attack. In this case it is *Synflood*
  - -i 10.0.0.3
    - Specifies the (destination) IP address of the victim using the '-i' flag
  - -p 80
    - Specifies the (destination) port number of the victim using the '-p' flag
    - Port 80 corresponds to HTTP

This is shown in the figure below.

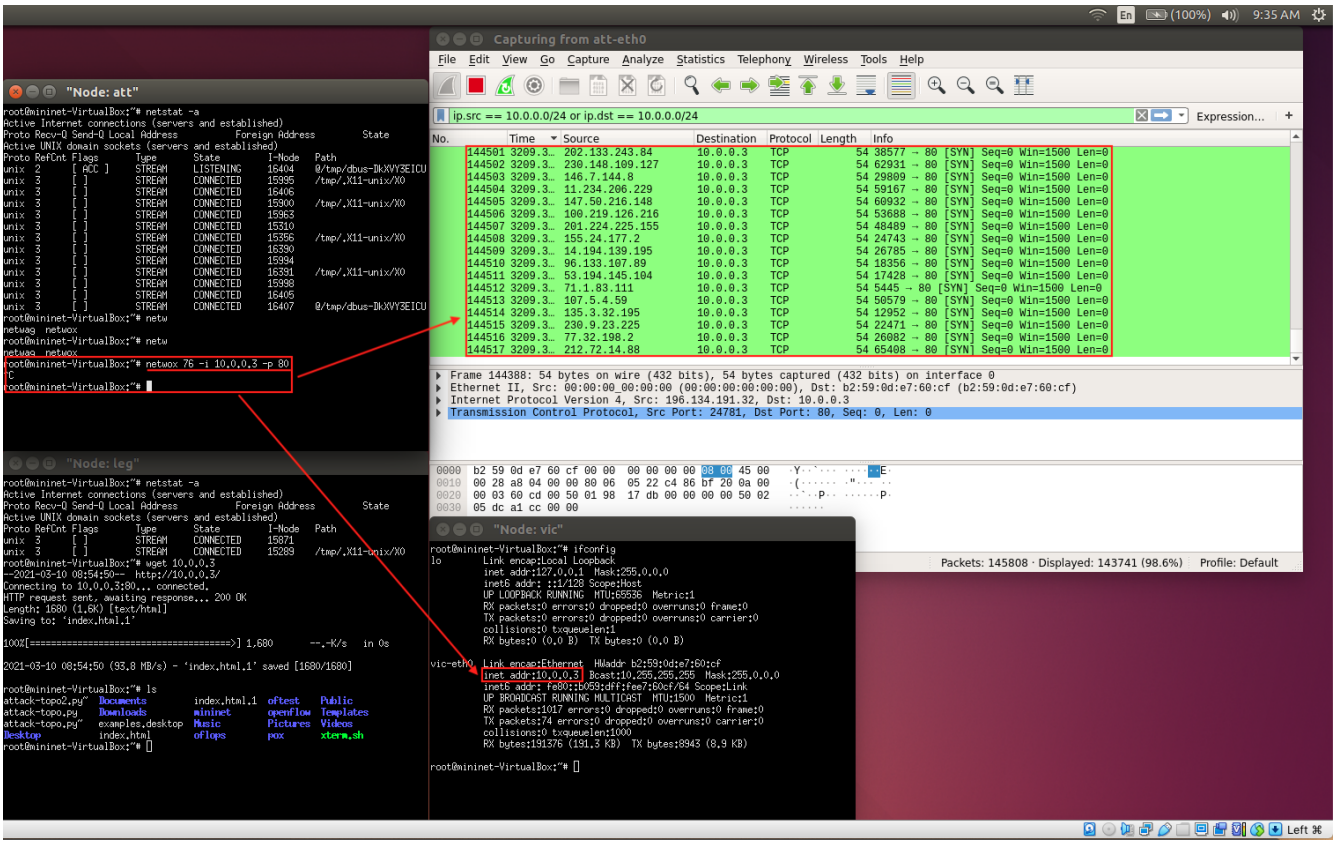

Figure 5: The 'synflood' command is shown in the 'att' node window on the left side of the figure; it is highlighted in a red box. On the right side of the figure, the result is shown in Wireshark's packet-listing window. As you can see, a lot of SYN-TCP packets are sent to the destination IP address of 10.0.0.3 on port number 80. At the bottom of the figure, the IP address of the 'vic' node is shown. This shows that the 'netwox' command is attacking the correct server by creating a lot of half-connections.

2. The output of the command *netstat -na* on node *vic* shows that there are several half-opened connections on port 80. To be specific, there are 5 half-opened connections, where the state is “stuck” in *SYN_RECV*. These are bogus requests and occupy limited space in the buffer of the *vic* node. This is demonstrated in the figure below.

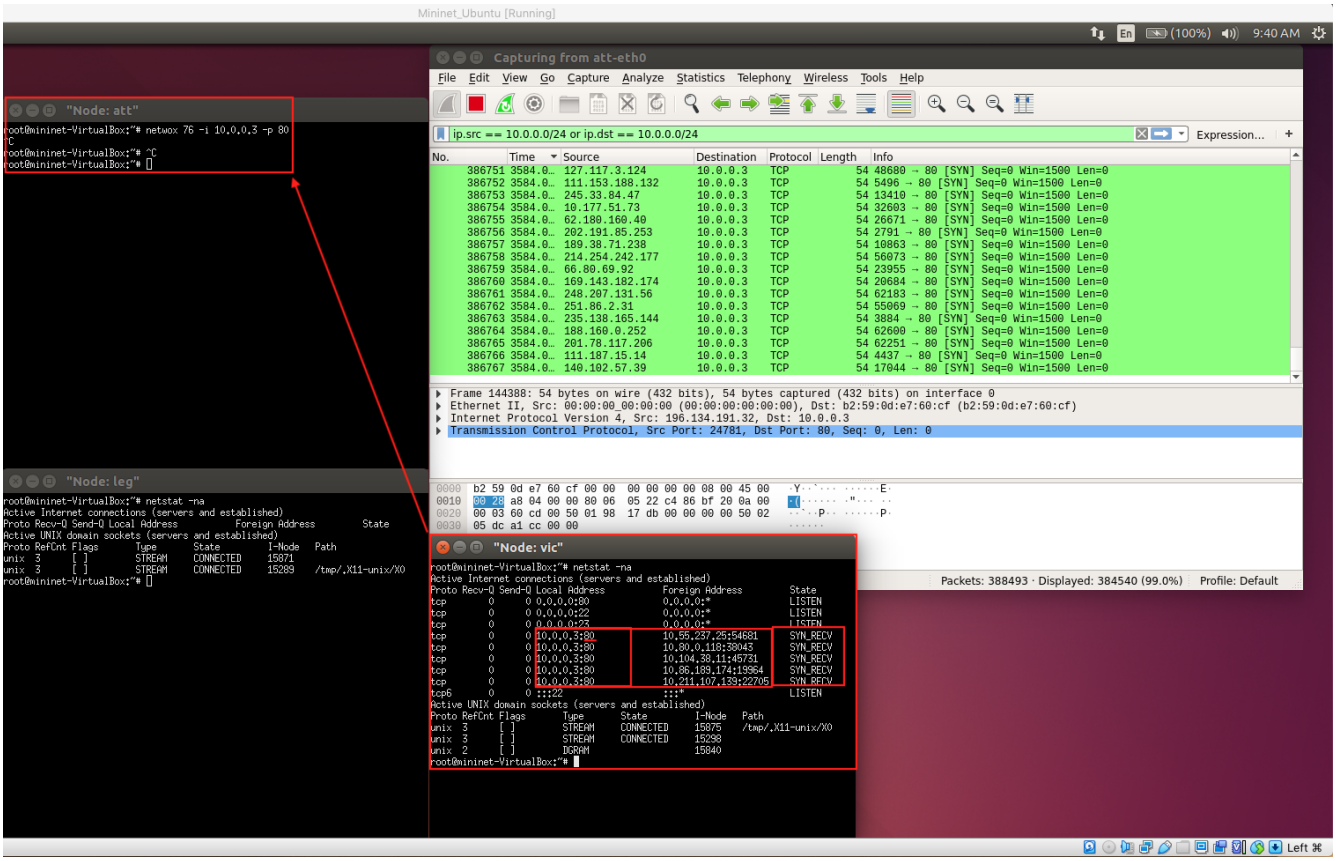

Figure 6: At the bottom of the figure, the output of the command 'netstat -na' is circled in red. The state of the 5 HTTP packets are in 'SYN_RECV'; these are half-connections. Also, the source IP addresses and port numbers are bogus. On the top-left side of the figure, the 'att' node is highlighted in red, and it shows the attack being performed by the 'att' node.

3. The source IP address AND port numbers of the *SYN* packets are spoofed by *netwox*. In other words, *netwox* floods the (web) server with packets from fake/bogus IP addresses that create half-connections. Since the source IP addresses are fake, and their corresponding requests are bogus, the (web) server's receiver buffer is flooded/full. This is illustrated in the figures below. The bogus requests are shown in the packet-listing window. *It's all a fugazi.*

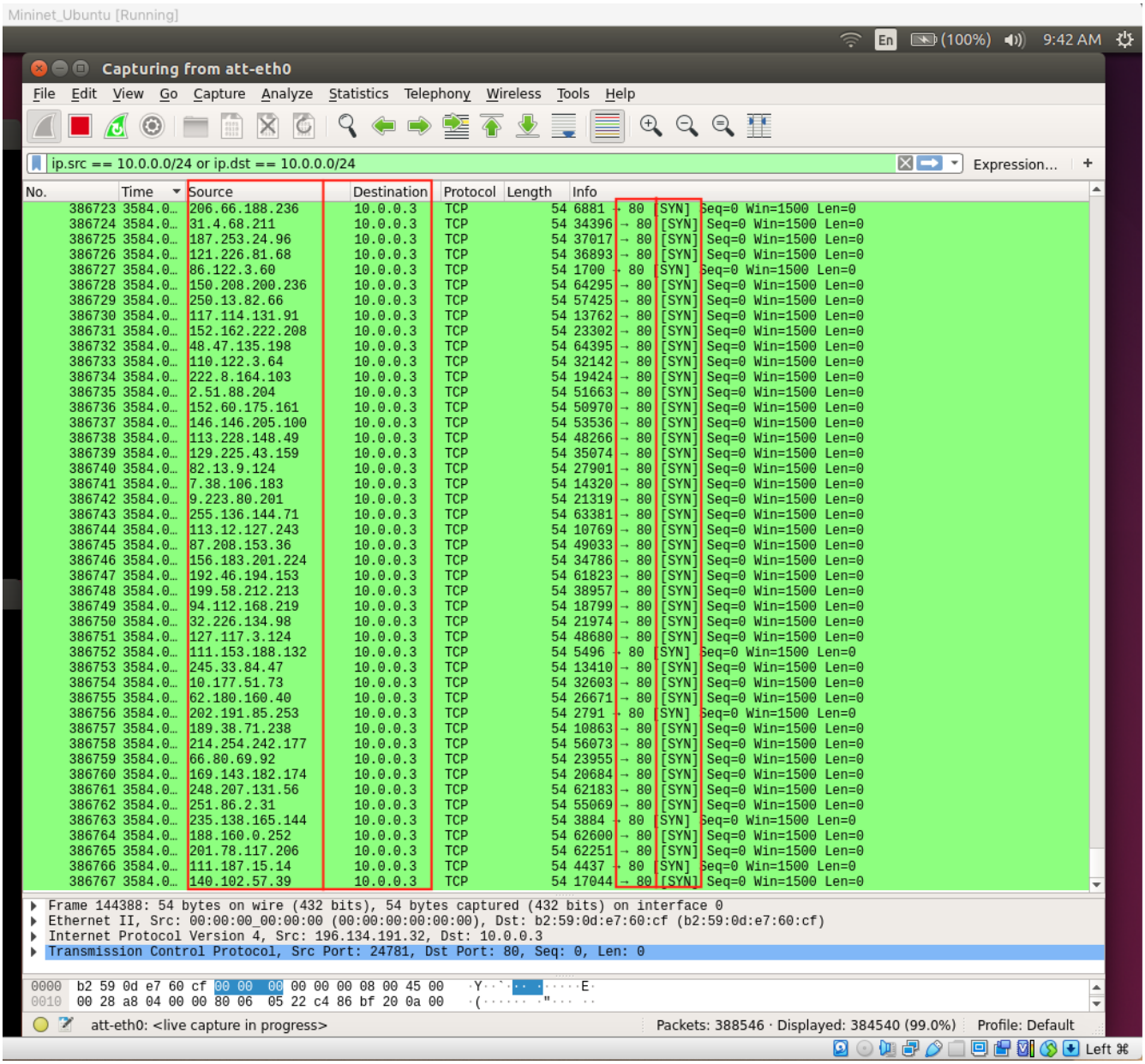

Figure 7: The entire packet-listing window in Wireshark is flooded with bogus SYN requests from the 'netwox' command being executed on the 'att' node. As you can see, the source column is full of spoofed IP addresses, all of the IPs in the destination column correspond to the 'vic' node, and the Info column is full of bogus 'SYN' requests to port number 80 on the destination host. These TCP-SYN packets create half-connections that flood the receiver host's buffer.

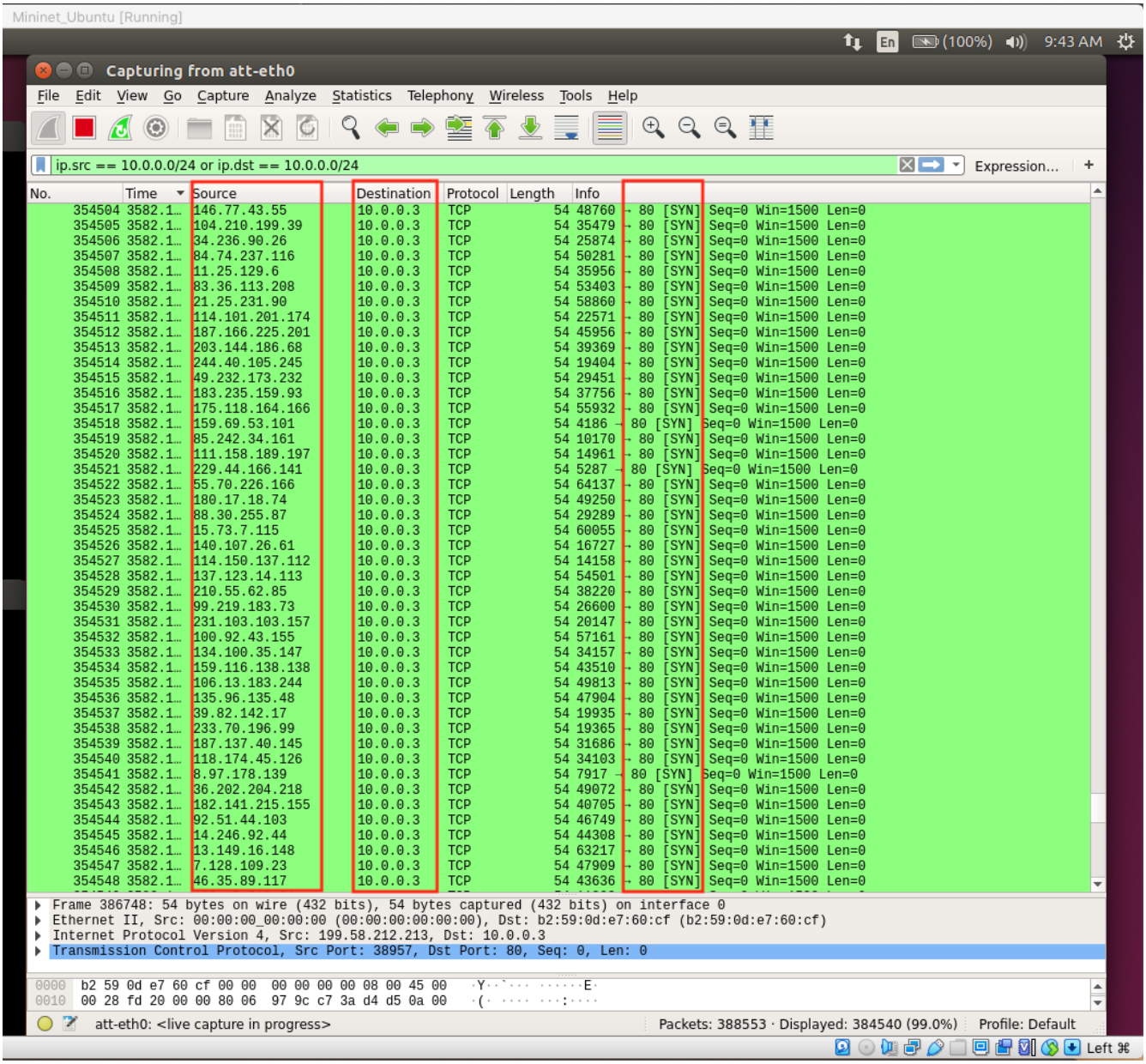

Figure 8: The entire packet-listing window in Wireshark is flooded with bogus SYN requests from the 'netwox' command being executed on the 'att' node. As you can see, the source column is full of spoofed IP addresses, all of the IPs in the destination column correspond to the 'vic' node, and the Info column is full of bogus 'SYN' requests to port number 80 on the destination host.

4. No, the *leg* node was not able to get any HTTP response from the server node *vic* during the attack from the *att* node. The *leg* node tried to connect to the server node *vic* for more than several dozen seconds before the attempt was manually terminated by me. Based on the output of the *netstat -na* command on the *vic* node, it seems like the *leg* node would not have been able to connect to it at all. It would have continued to wait until a synchronous or asynchronous interrupt occurred; timeout or manual cancellation, respectively. The trace in the Wireshark window does not display any packets/requests from the *leg* node to the *vic* node, or vice versa. The Wireshark window only displays half-connections created by *netwox* on the *att* node.
This is demonstrated in the figures below.

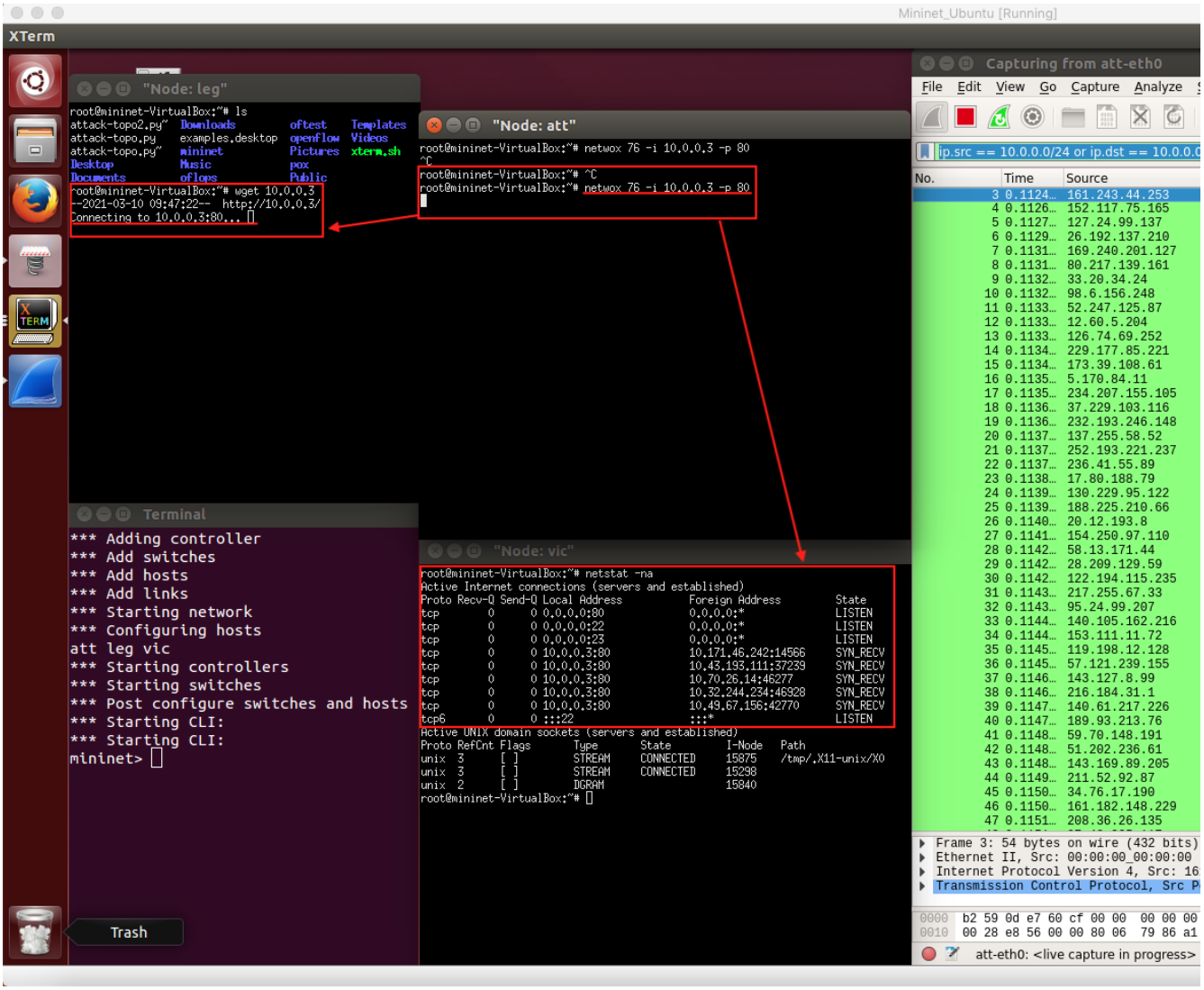

Figure 9: The 'att' node is attacking the 'vic' node, as shown in the red box at the top of the figure. The 'vic' node's buffer is flooded with bogus requests from spoofed IP address generated by the 'netwox' command on the 'att' node. This is shown in the red box at the bottom of the figure. The top-left of the figure shows the 'leg' node trying to connect to the 'vic' node via 'wget', but it is unable to get an ACK back. Hence, it is stuck in the connecting state. The right side of the figure shows the Wireshark trace of the entire communication between all three nodes. As you can see, the packet-listing window is flooded with bogus requests from spoofed IP addresses to the 'vic' node.

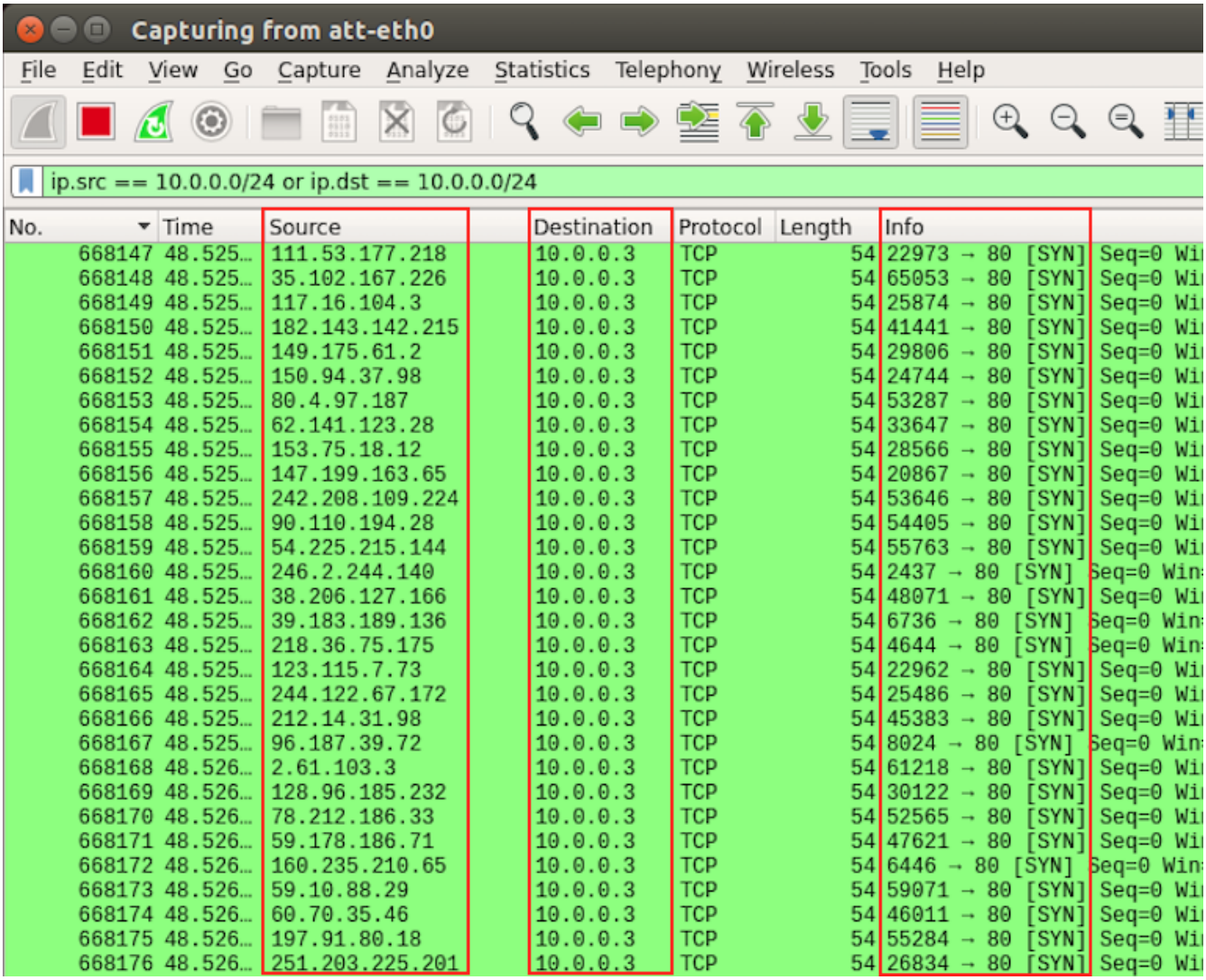

Figure 10: This is the Wireshark trace of the connections from and to '10.0.0.3', which corresponds to the 'vic' (server) node. As you can see, the packet-listing window is flooded with bogus SYN packets generated by the 'netwox' command from the 'att' node. The IP address and port numbers of these fake requests are spoofed by 'netwox'. There is no HTTP GET or RESPONSE packet from 10.0.0.2, which corresponds to the 'leg' node. In fact, the 'leg' node was not able to connect at all to the 'vic' node because of the DOS attack from the 'att' node.

5 – Report

Based on the definition of a SYN flood attack, and the results of figures 8, 9, & 10, it is evident that the *SYN flood* attack carried out by the *att* node on the *vic* server node was successful. This is because the *leg* node was unable to setup a connection with the *vic* node, thus it was unable to download/get the *index.html* file from the server. The connection was attempted for several dozens of seconds, but it failed to setup a connection, because the receiver host's buffer was flooded with half-connections. Furthermore, once the SYN flood attack was manually terminated on the *att* node, the *leg* node was able to setup a connection with the *vic* server node, and download the corresponding HTML file. This is shown in the figure below. Also, the packets listed in Wireshark's packet-listing window ticks all the boxes of a SYN Flood attack, and the *vic* server displayed all the symptoms of a SYN Flood attack. Numerous bogus packets are sent from the *att* node to the *vic* node, and their source IP address and port number are forged. The *leg* node was only able to communicate with the *vic* server node after the attack from the *att* was terminated. This is also demonstrated in the figure below, in the Wireshark window.

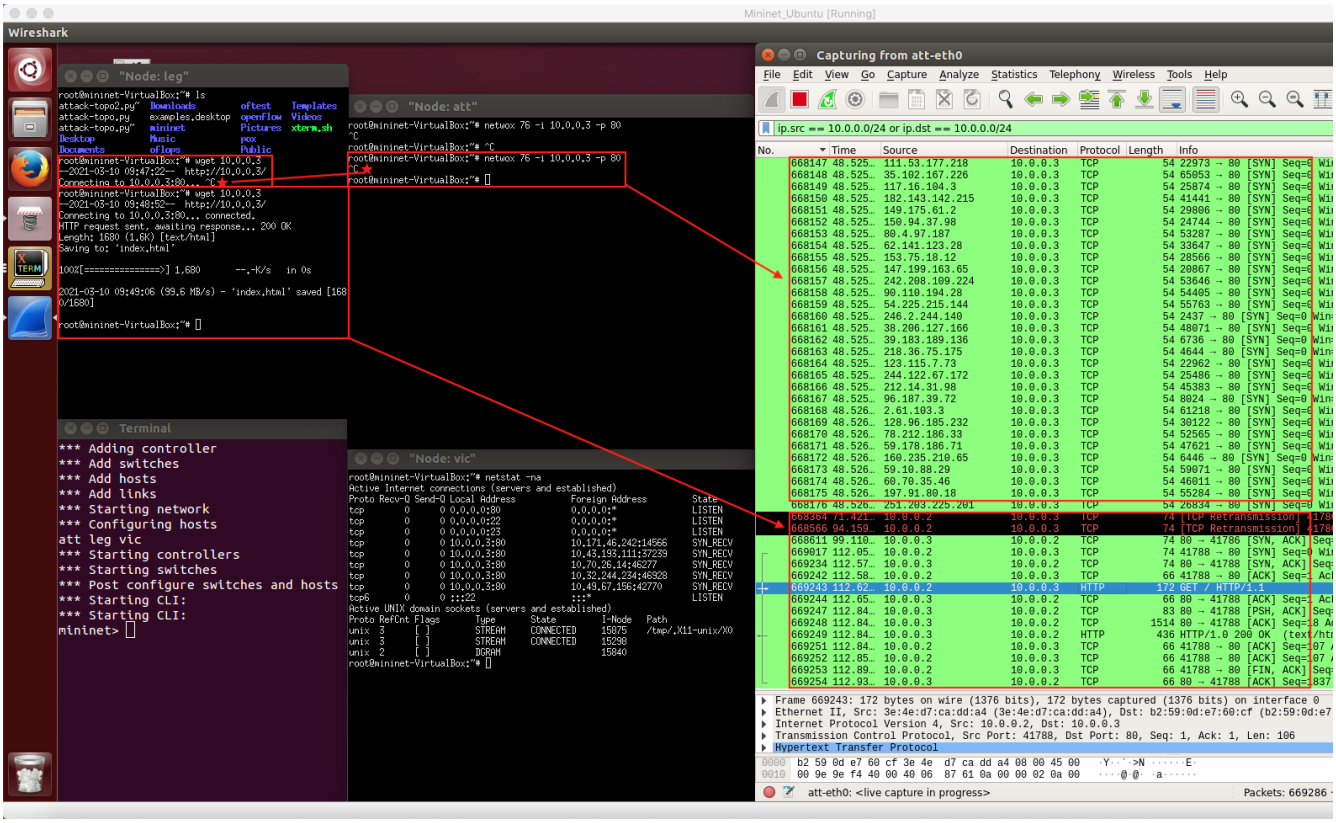

Figure 11: Once the attack from the 'att' node was terminated, the 'leg' node was able to communicate with the 'vic' node. This is evident in the terminal window of the 'att' and 'vic' node; they are highlighted in a red box at the top-left of the screen. The star indicates when the commands were terminated. In the 'leg' node, another connection to the 'vic' node is attempted, and is successful after a few seconds. On the right side of the figure is the Wireshark window. The packet-listing window in Wireshark is divided into two red boxes. The box at the top corresponds to the bogus requests from the 'att' node, and the box at the bottom corresponds to the legitimate request from the 'leg' node. As you can see, based on the Time column, the 'leg' node was only able to communicate with the 'vic' node after the 'att' node halted its SYN flood attack. Therefore, the SYN flood attack carried out by the 'att' node on the 'vic' node was successful.