Firewall Evasion Lab

Submitted by: Mahika Gupta
SRN: PES1UG20CS243
DATE: 06/11/2022

Lab Environment Setup

Please download the Labsetup.zip file from the link given below :

https://seedsecuritylabs.org/Labs_20.04/Networking/Firewall_Evasion/

Follow the instructions in the lab setup document to set up the lab environment.

Task 0 : Get Familiar with the Lab Setup

We will conduct a series of experiments in this chapter. These experiments need to use several computers in two separate networks. The experiment setup is depicted in Figure 1. We use docker containers for these machines. Readers can find the container setup file from the website of this lab. In this lab, the network 10.8.0.0/24 serves as an external network, while 192.168.20.0/24 serves as the internal network.
The host 10.8.0.1 is not a container; this IP address is given to the host machine (i.e., the VM in our case). This machine is the gateway to the Internet. To reach the Internet from the hosts in both 192.168.20.0/24 and 10.8.0.0/24 networks, packets must be routed to 10.8.0.1. The routing has already been set up.

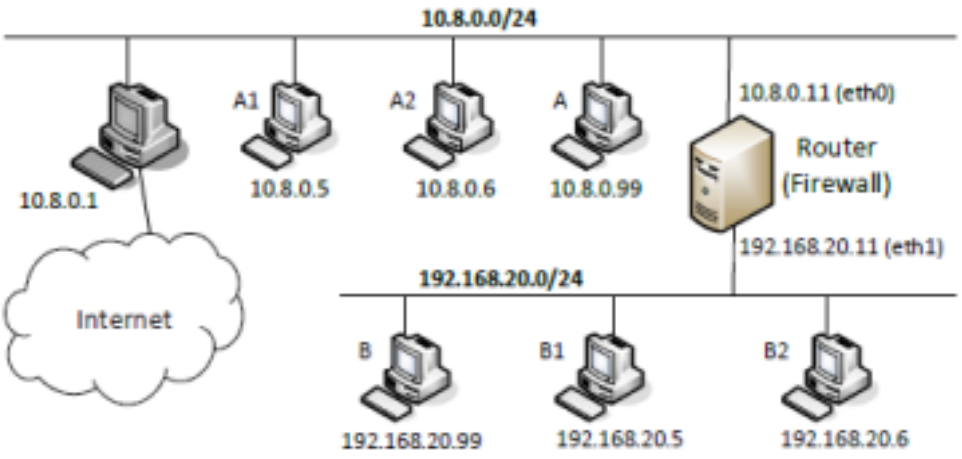


Figure 1: Network setup

Router configuration: setting up NAT. The following iptables command is included in the router configuration inside the docker-compose.yml file. This command sets up a NAT on the router for the traffic going out from its eth0 interface, except for the packets to 10.8.0.0/24. With this rule,
for packets going out to the Internet, their source IP address will be replaced by the router's IP address 10.8.0.11. Packets going to 10.8.0.0/24 will not go through NAT.

iptables -t nat -A POSTROUTING ! -d 10.8.0.0/24 -j MASQUERADE -o eth0

In the above command, we assume that eth0 is the name assigned to the interface connecting the router to the 10.8.0.0/24 network. This is not guaranteed. The router has two Ethernet interfaces; when the router container is created, the name assigned to this interface might be eth1. You can find out the correct interface name using the following command. If the name is not eth0, you should make a change to the command above inside the docker-compose.yml file, and then restart the containers.

```
# ip -br address
lo               UNKNOWN        127.0.0.1/8
eth1@if1907      UP             192.168.20.11/24
eth0@if1909      UP             10.8.0.11/24
```

Router configuration: Firewall rules. We have also added the following firewall rules on the router. Please make sure that eth0 is the interface connected to the 10.8.0.0/24 network and that eth1 is the one connected to 192.168.20.0/24. If not, make changes accordingly.

```
router-firewall:PES1UG20CS243:Mahika:/
#>ip -br address
lo               UNKNOWN        127.0.0.1/8
eth0@if23        UP             10.8.0.11/24
eth1@if25        UP             192.168.20.11/24
router-firewall:PES1UG20CS243:Mahika:/
#>
```

The interfaces are assigned to the right networks.

```
// Ingress filtering: only allows SSH traffic
iptables -A FORWARD -i eth0 -p tcp -m conntrack \
        --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp -j DROP

// Egress filtering: block www.example.com
iptables -A FORWARD -i eth1 -d 93.184.216.0/24 -j DROP
```

The first rule allows TCP packets to come in if they belong to an established or related connection. This is a stateful firewall rule. The second rule allows SSH, and the third rule drops all other TCP packets if they do not satisfy the first or the second rule. The fourth rule is an egress firewall rule, and it prevents the internal hosts from sending packets to 93.184.216.0/24, which is the network for www.example.com.

Lab task. Please block two more websites and add the firewall rules to the setup files. The choice of websites are up to you. **Keep in mind that most popular websites have multiple IP addresses that can change from time to time.** After adding the rules, start the containers, and verify that all the ingress and egress firewall rules are working as expected.

On the router-firewall container:

iptables -A FORWARD -i eth1 -d 13.107.42.0/24 -j DROP
This IP address was for www.linkedin.com

iptables -A FORWARD -i eth1 -d 13.249.221.0/24 -j DROP
This IP address was for www.miniclip.com

```
router-firewall:PES1UG20CS243:Mahika:/
#>iptables -A FORWARD -i eth1 -d 13.107.42.0/24 -j DROP
router-firewall:PES1UG20CS243:Mahika:/
#>iptables -A FORWARD -i eth1 -d 13.249.221.0/24 -j DROP
router-firewall:PES1UG20CS243:Mahika:/
#>
```

These rules are appended to the iptable, where all packets coming from www.linkedin.com and www.miniclip.com will be dropped on eth1 interface

To verify that the websites have been blocked, ping them from any machine in the internal network ie. **B/B1/B2**.

 # ping www.linkedin.com

```
B-192.168.20.99:PES1UG20CS243:Mahika:/
#>ping www.linkedin.com
PING l-0005.l-msedge.net (13.107.42.14) 56(84) bytes of data.
^C
--- l-0005.l-msedge.net ping statistics ---
2074 packets transmitted, 0 received, 100% packet loss, time 8220886ms
```

 # ping www.miniclip.com

```
B1-192.168.20.5:PES1UG20CS243:Mahika:/
#>ping www.miniclip.com
PING www.miniclip.com (13.249.221.87) 56(84) bytes of data.
^C
--- www.miniclip.com ping statistics ---
3207 packets transmitted, 0 received, 100% packet loss, time 12132784ms

B1-192.168.20.5:PES1UG20CS243:Mahika:/
#>
```

As we can see, the ping is not successful in either case.

Task 1 : Static Port Forwarding

The firewall in the lab setup prevents outside machines from connecting to any TCP server on the internal network, other than the SSH server. In this task, we would like to use static port forwarding to evade this restriction. More specifically, we will use ssh to create a static port forwarding tunnel between host A (on the external network) and host B (on the internal network), so whatever data received on A's port X will be sent to B, from where the data is forwarded to the target T's port Y. In the following command, we use ssh to create such a tunnel.

```
$ ssh -4NT -L <A's IP>:<A's port X>:<T's IP>:<T's port Y>  <user id>@<B's IP>

// -4: use IPv4 only, or we will see some error message.
// -N: do not execute a remote command.
// -T: disable pseudo-terminal allocation (save resources).
```

Regarding A's IP, typically we use 0.0.0.0, indicating that our port forwarding will listen to the connection from all the interfaces on A. If we want to limit the connections to a particular interface, we should use that interface's IP address. For example, if we want to limit the connection to the loopback interface, so only the program on the local host can use this port forwarding, we can use 127.0.0.1:<port> or simply omit the IP address (the default IP address is 127.0.0.1).

Lab task. Please use static port forwarding to create a tunnel between the external network and the internal network, so we can telnet into the server on B. Please demonstrate that you can do such telnet from hosts A, A1 and A2. Moreover, please answer the following questions:

(1) How many TCP connections are involved in this entire process. You should run wireshark or tcpdump to capture the network traffic, and then point out all the involved TCP connections from the captured traffic.

Ans: There are 3 tcp connections in total, one ssh tunnel from host A to B, telnet from A1 to A, telnet from A2 to A. SSH uses tcp connection and hence ssh tunnel created is considered to be one tcp connection.

(2) Why can this tunnel successfully help users evade the firewall rule specified in the lab setup?

Ans: In the firewall rule, it says all tcp packets from external hosts will be dropped. However, all ssh traffic is allowed to pass. Hence by making an ssh tunnel, we send the tcp packets from host A, A1, A2 as ssh traffic, which is allowed by the firewall. Those tcp packets are then forwarded to port 23 of host B from the port 22 of host B. The rule that allows ssh packets to flow through is what helps users to evade firewall.

Keep wireshark open and select the interface with the IP address of 192.168.20.1. Wireshark is to be opened on the host VM.

Run the below command on container A:

 # ssh -L 0.0.0.0:8000:192.168.20.99:23 root@192.168.20.99

```
A-10.8.0.99:PES1UG20CS243:Mahika:/
#>ssh -L 0.0.0.0:8000:192.168.20.99:23 root@192.168.20.99
The authenticity of host '192.168.20.99 (192.168.20.99)' can't be established.
ECDSA key fingerprint is SHA256:nSfnpySG1K6SCVYFCi+TYjD0C7kW07NiT6g9HgOeu9c.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.20.99' (ECDSA) to the list of known hosts.
root@192.168.20.99's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

B-192.168.20.99:PES1UG20CS243:Mahika:~
#>
```

We can see below that we have successfully tunneled to B's system. We have done local port forwarding between A and B.

Run the below command on containers A1 and A2:

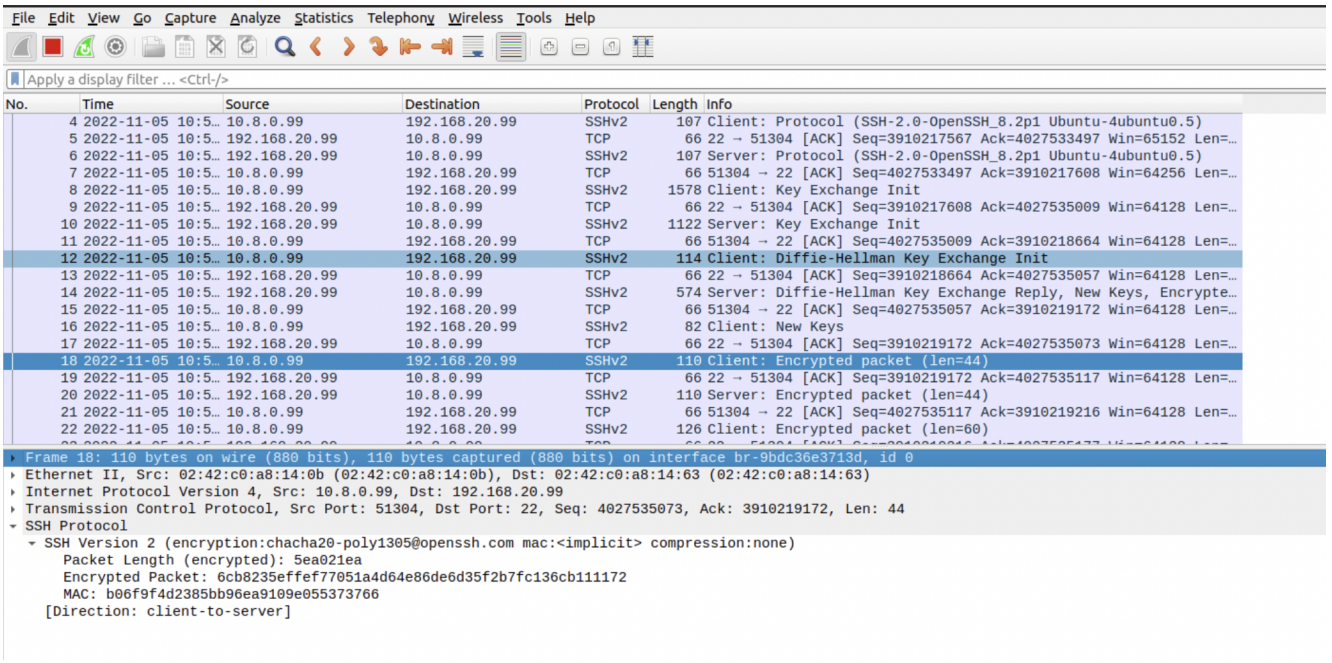
 # telnet 10.8.0.99 8000

```
A1-10.8.0.5:PES1UG20CS243:Mahika:/
#>telnet 10.8.0.99 8000
Trying 10.8.0.99...
Connected to 10.8.0.99.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
c0bdcf4cf538 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@c0bdcf4cf538:~$
```

```
A2-10.8.0.6:PES1UG20CS243:Mahika:/
#>telnet 10.8.0.99 8000
Trying 10.8.0.99...
Connected to 10.8.0.99.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
c0bdcf4cf538 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Nov  5 14:57:15 UTC 2022 from c0bdcf4cf538 on pts/3
seed@c0bdcf4cf538:~$
```

We can see that the telnet connection is established and successful.

Observing the wireshark capture:



We can see that no telnet connection packets are visible, only ssh packets are visible, which means the telnet traffic is inside the ssh traffic, and hence the firewall allows the packets to enter, considering them as ssh traffic.

Task 2: Dynamic Port Forwarding

In the static port forwarding, each port-forwarding tunnel forwards the data to a particular destination. If we want to forward data to multiple destinations, we need to set up multiple tunnels. For example, using port forwarding, we can successfully visit the blocked example.com website, but what if the firewall blocks many other sites, how do we avoid tediously establishing an SSH tunnel for each site? We can use dynamic port forwarding to solve this problem.

Task 2.1: Setting Up Dynamic Port Forwarding

We can use ssh to create a dynamic port-forwarding tunnel between B and A. We run the following command on host B. In dynamic port forwarding, B is often called proxy.

Run in container B to set up the ssh shell with dynamic port forwarding

enabled: **# ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N**

```
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N
The authenticity of host '10.8.0.99 (10.8.0.99)' can't be established.
ECDSA key fingerprint is SHA256:nSfnpySG1K6SCVYFCi+TYjD0C7kW07NiT6g9HgOeu9c.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.8.0.99' (ECDSA) to the list of known hosts.
root@10.8.0.99's password:
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>
```

Here a dynamic ssh tunnel is created B and A where B acts as a proxy server.

Regarding B's IP, typically we use 0.0.0.0, indicating that our port forwarding will listen to the connection from all the interfaces on B. After the tunnel is set up, we can test it using the curl command.
We specify a proxy option, so curl will send its HTTP request to proxy B, which listens on port X. The proxy forwards the data received on this port to the other end of the tunnel (host A), from where the data will be further forwarded to the target website. The type of proxy is called SOCKS version 5, so that is why we specify socks5h.

Lab task. Please demonstrate that you can visit all the blocked websites using curl from hosts B, B1, and B2 on the internal network. Please also answer the following questions:

(1) Which computer establishes the actual connection with the intended web server?

Ans: A is the computer that establishes the actual connection with the web server.

(2) How does this computer know which server it should connect to?

Ans: The SOCKS proxy is used so that the browser tells the proxy which destination to use, as it cannot find that information itself.

Run in container B:

 # curl -x socks5h://0.0.0.0:8000 http://www.example.com

```
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>curl -x socks5h://0.0.0.0:8000 http://www.example.com
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
```

```
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>
```

To access the websites from B1,B2 run the following command in the corresponding containers:

 # curl -x socks5h://192.168.20.99:8000 http://www.example.com

```
B1-192.168.20.5:PES1UG20CS243:Mahika:/
#>curl -x socks5h://192.168.20.99:8000 http://www.example.com
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
```

```
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
B1-192.168.20.5:PES1UG20CS243:Mahika:/
#>
```

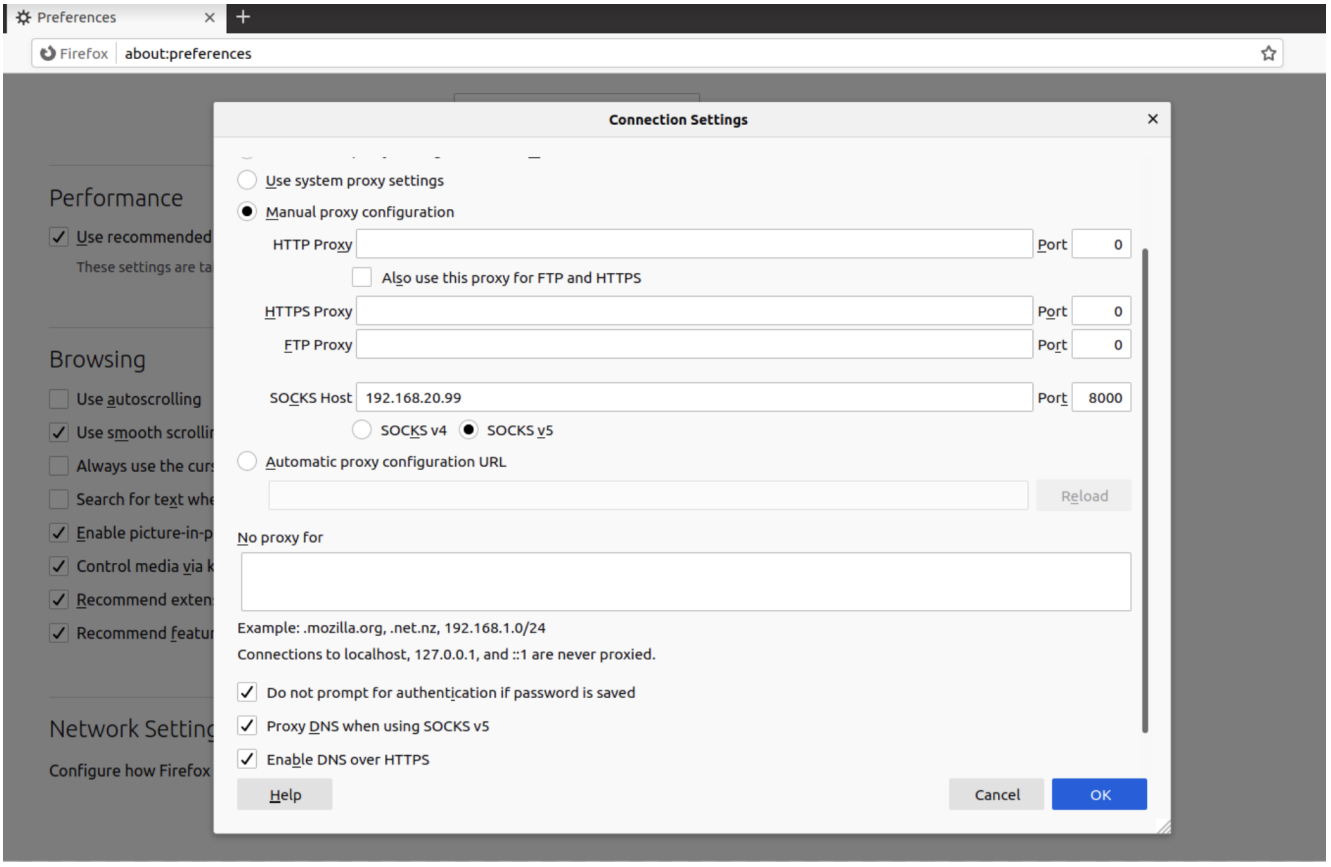
We can see that B ,B1 and B2 are able to access the websites due to dynamic port forwarding.

Task 2.2: Testing the Tunnel Using Browser

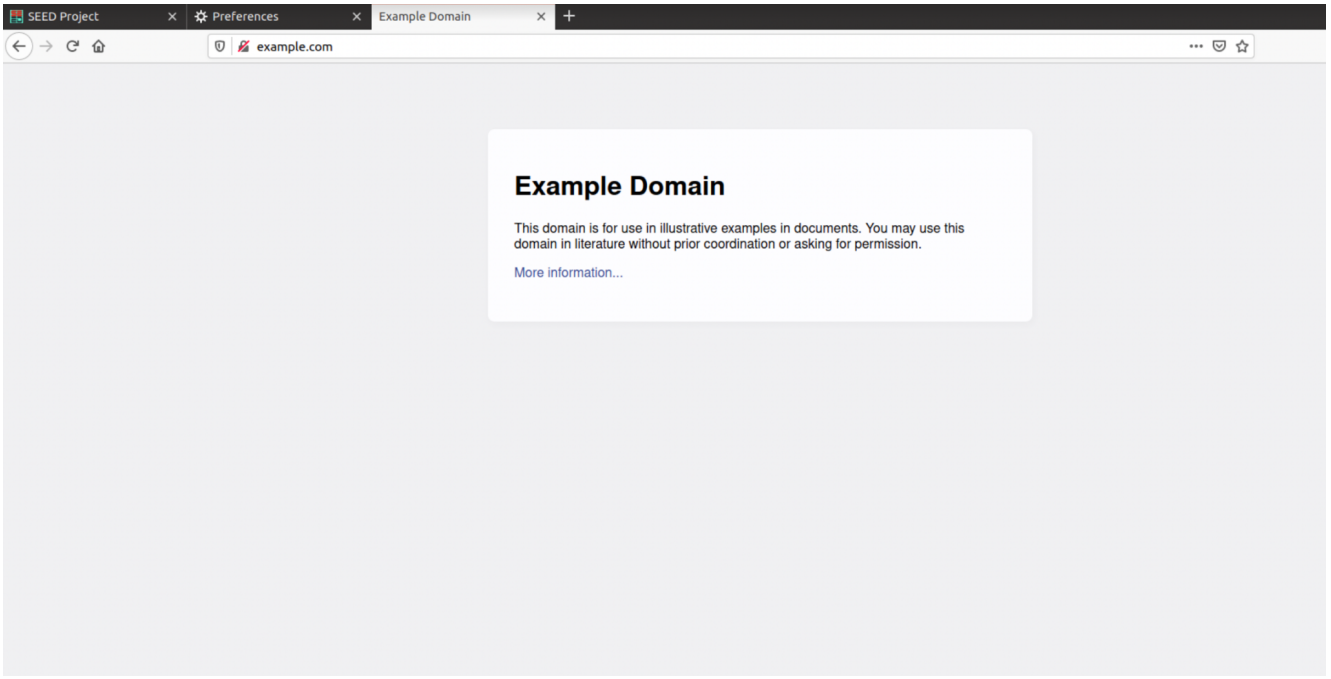
We can also test the tunnel using a real browser, instead of using curl. Although it is hard to run a browser inside a container, in the docker setup, by default, the host machine is always attached to any network created inside docker, and the first IP address on that network is assigned to the host machine. For example, in our setup, the host machine is the SEED VM; its IP address on the internal network 192.168.20.0/24 is 192.168.20.1.

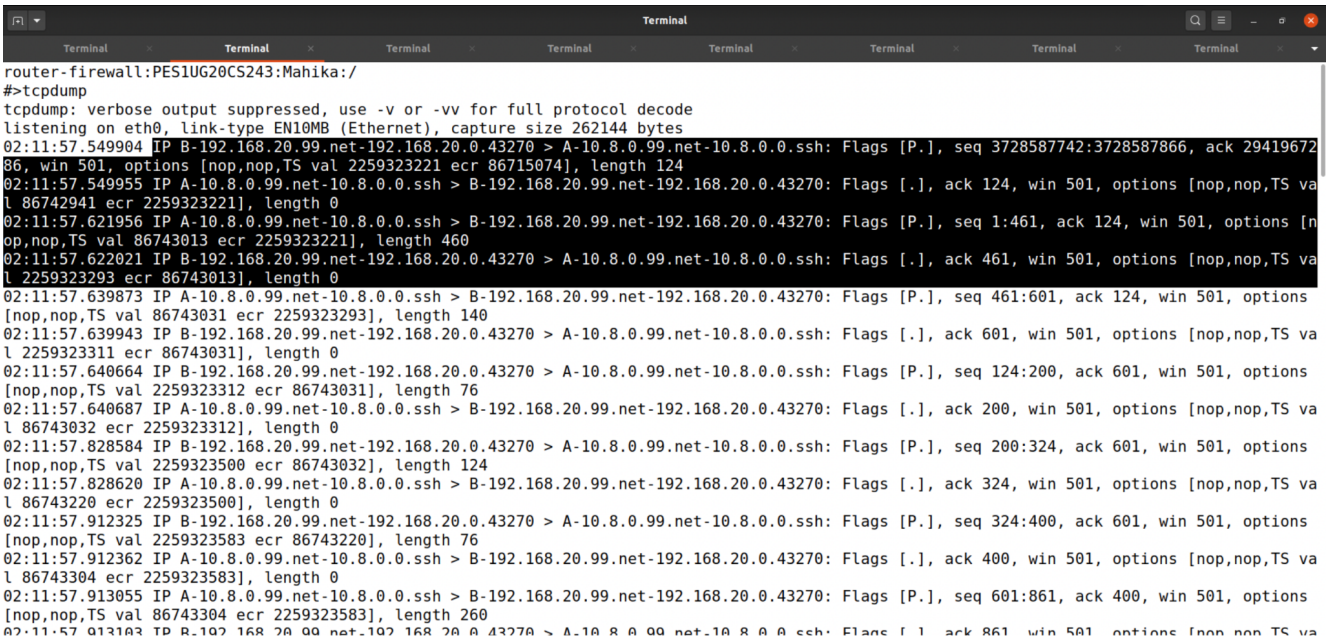
To use the dynamic port forwarding, we need to configure Firefox's proxy setting. To get to the setting page, we can type about:preferences in the URL field or click the Preference menu item. On the General page, find the "Network Settings" section, click the Settings button, and a window will pop up. Follow the figure to set up the SOCKS proxy.

Lab task. Once the proxy is configured, we can then browse any website. The requests and replies will go through the SSH tunnel. Since the host VM can reach the Internet directly, to make sure that our web browsing traffic has gone through the tunnel, you should do the following:



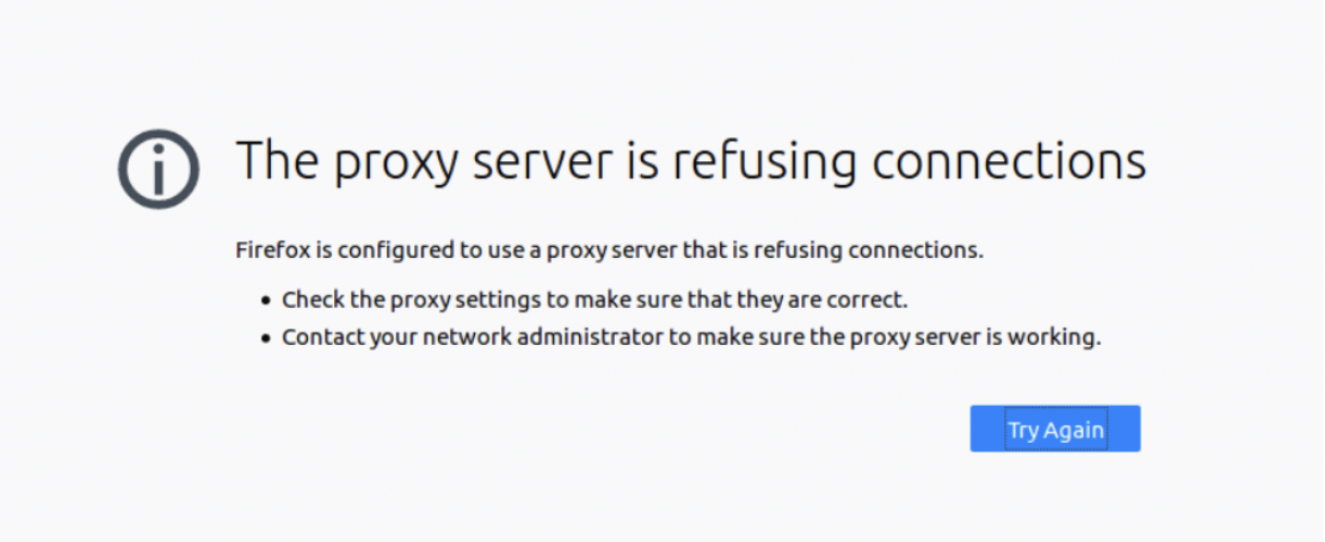
(1) run tcpdump on the router-firewall, and point out the traffic involved in the entire port forwarding process.





Here we can see that the traffic is ssh traffic between host A and host B. There is no HTTP traffic involved. The website can be accessed.

(2) Break the SSH tunnel, and then try to browse a website. Describe your observation.




```
router-firewall:PES1UG20CS243:Mahika:/
#>tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
02:59:11.824097 IP6 fe80::a478:9fff:fe38:b81f > ip6-allrouters: ICMP6, router solicitation, length 16
02:59:11.824792 IP 3372caafcf1c.58075 > dlink.router.domain: 53702+ PTR? f.1.8.b.8.3.e.f.f.f.9.8.7.4.a.0.0.0.0.0.0.0.0.0.6
arpa. (90)
02:59:12.308241 IP dlink.router.domain > 3372caafcf1c.58075: 53702*- 0/0/0 (90)
02:59:12.310983 IP 3372caafcf1c.37648 > dlink.router.domain: 26843+ PTR? 1.0.168.192.in-addr.arpa. (42)
02:59:12.323919 IP dlink.router.domain > 3372caafcf1c.37648: 26843- 1/0/0 PTR dlink.router. (68)
02:59:16.952740 ARP, Request who-has 10.8.0.1 tell 3372caafcf1c, length 28
02:59:16.953669 ARP, Reply 10.8.0.1 is-at 02:42:78:d7:14:fe (oui Unknown), length 28
02:59:16.954510 IP 3372caafcf1c.51027 > dlink.router.domain: 15623+ PTR? 1.0.8.10.in-addr.arpa. (39)
02:59:17.455202 ARP, Request who-has 3372caafcf1c tell 10.8.0.1, length 28
02:59:17.455222 ARP, Reply 3372caafcf1c is-at 02:42:0a:08:00:0b (oui Unknown), length 28
02:59:17.601428 IP dlink.router.domain > 3372caafcf1c.51027: 15623*- 0/0/0 (39)
02:59:17.713851 IP6 fe80::42:78ff:fed7:14fe.mdns > ff02::fb.mdns: 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.l
02:59:17.715004 IP 3372caafcf1c.48986 > dlink.router.domain: 6044+ PTR? b.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
rpa. (90)
02:59:17.730891 IP dlink.router.domain > 3372caafcf1c.48986: 6044*- 0/0/0 (90)
02:59:17.732466 IP 3372caafcf1c.54962 > dlink.router.domain: 26720+ PTR? e.f.4.1.7.d.e.f.f.f.8.7.2.4.0.0.0.0.0.0.0.0.0.0.6
arpa. (90)
02:59:18.378756 IP dlink.router.domain > 3372caafcf1c.54962: 26720*- 0/0/0 (90)
02:59:19.824873 IP6 fe80::a478:9fff:fe38:b81f.mdns > ff02::fb.mdns: 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp
02:59:50.735773 IP6 fe80::42:78ff:fed7:14fe > ip6-allrouters: ICMP6, router solicitation, length 16
02:59:55.363267 IP6 fe80::a478:9fff:fe38:b81f.mdns > ff02::fb.mdns: 0 [7q] PTR (QM)? _ftp._tcp.local. PTR (QM)? _nfs._tcp.
fpovertcp._tcp.local. PTR (QM)? _smb._tcp.local. PTR (QM)? _sftp-ssh._tcp.local. PTR (QM)? _webdavs._tcp.local. PTR (QM)?
 (118)
02:59:55.363447 IP6 fe80::42:78ff:fed7:14fe.mdns > ff02::fb.mdns: 0 [7q] PTR (QM)? _ftp._tcp.local. PTR (QM)? _nfs._tcp.lo
overtcp._tcp.local. PTR (QM)? _smb._tcp.local. PTR (QM)? _sftp-ssh._tcp.local. PTR (QM)? _webdavs._tcp.local. PTR (QM)? _w
```

We can see that we are not able to access the website now as ssh tunnel was removed.

Please note, we are still using the ssh connection established in the previous Task 2.1.

Access the websites as you normally would in firefox and provide screenshots of your

observations.

To close the ssh shell that is in the background created using the previous “ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N” command :

Run the below commands in container B:

 # ps -eaf | grep "ssh"

 # kill [corresponding pid of the process]

```
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>ps -eaf | grep "ssh"
root          40       1  0 Nov05 ?        00:00:00 sshd: /usr/sbin/sshd [listener] 0 of 10-100 startups
root          56      40  0 00:30 ?        00:00:00 sshd: root@pts/2
root         104       1  0 00:35 ?        00:00:00 ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N
root         108      50  0 02:21 pts/1    00:00:00 grep ssh
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>kill 104
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>
```

Task 2.3: Writing a SOCKS Client Using Python

For port forwarding to work, we need to specify where the data should be forwarded to (the final destination). In the static case, this piece of information is provided when we set up the tunnel, i.e., it is hard-wired into the tunnel setup. In the dynamic case, the final destination is dynamic, not specified during the setup, so how can the proxy know where to forward the data? Applications using a dynamic port forwarding proxy must tell the proxy where to forward their data. This is done through an additional protocol between the application and the proxy. A common protocol for such a purpose is the SOCKS (Socket Secure) protocol, which becomes a de facto proxy standard.
Since the application needs to interact with the proxy using the SOCKS protocol, the application software must have a native SOCKS support in order to use SOCKS proxies. Both Firefox and curl have such a support, but we cannot directly use this type of proxy for the telnet program, because it does not provide a native SOCKS support. In this task, we implement a very simple SOCKS client program using Python.

Lab task. Please complete this program, and use it to access http://www.example.com from hosts B, B1, and B2. The code given above is only for sending HTTP requests, not HTTPS requests (sending HTTPS requests are much more complicated due to the TLS handshake). For this task, students only need to send HTTP requests.

Run in container B:

 # ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N

```
B-192.168.20.99:PES1UG20CS243:Mahika:/
#>ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N
root@10.8.0.99's password:
B-192.168.20.99:PES1UG20CS243:Mahika:/
```

A dynamic port forwarding tunnel is created between B and A

Run in container B :

 # python3 B-Socks-Client.py

```
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>python3 B-Socks-Client.py
[b'HTTP/1.0 200 OK', b'Accept-Ranges: bytes', b'Age: 299095', b'Cache-Cont
rol: max-age=604800', b'Content-Type: text/html; charset=UTF-8', b'Date: S
un, 06 Nov 2022 03:02:43 GMT', b'Etag: "3147526947"', b'Expires: Sun, 13 N
ov 2022 03:02:43 GMT', b'Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT', b'
Server: ECS (dcb/7EEC)', b'Vary: Accept-Encoding', b'X-Cache: HIT', b'Cont
ent-Length: 1256', b'Connection: close', b'', b'<!doctype html>\n<html>\n<
head>\n    <title>Example Domain</title>\n\n    <meta charset="utf-8" />\n
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />\
n    <meta name="viewport" content="width=device-width, initial-scale=1" /
>\n    <style type="text/css">\n    body {\n        background-color: #f0f
0f2;\n        margin: 0;\n        padding: 0;\n        font-family: -apple
-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetic
a Neue", Helvetica, Arial, sans-serif;\n        \n    }\n    div {\n
  width: 600px;\n        margin: 5em auto;\n        padding: 2em;\n
 background-color: #fdfdff;\n        border-radius: 0.5em;\n        box-sh
adow: 2px 3px 7px 2px rgba(0,0,0,0.02);\n    }\n    a:link, a:visited {\n
       color: #38488f;\n        text-decoration: none;\n    }\n    @media
(max-width: 700px) {\n        div {\n            margin: 0 auto;\n
    width: auto;\n        }\n    }\n    </style>    \n</head>\n\n<body>\n<
div>\n    <h1>Example Domain</h1>\n    <p>This domain is for use in illust
```

Run in containers B1 and B2:

 # python3 B1-B2-Socks-Client.py

```
PES1UG20CS243:Mahika~/.../Labsetup$>docksh 12
B1-192.168.20.5:PES1UG20CS243:Mahika:/
#>cd pyfiles
B1-192.168.20.5:PES1UG20CS243:Mahika:/pyfiles
#>python3 B1-B2-Socks-Client.py
[b'HTTP/1.0 200 OK', b'Age: 406886', b'Cache-Control: max-age=604800', b'C
ontent-Type: text/html; charset=UTF-8', b'Date: Sun, 06 Nov 2022 03:03:21
GMT', b'Etag: "3147526947+ident"', b'Expires: Sun, 13 Nov 2022 03:03:21 GM
T', b'Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT', b'Server: ECS (dcb/7F
3A)', b'Vary: Accept-Encoding', b'X-Cache: HIT', b'Content-Length: 1256',
b'Connection: close', b'', b'<!doctype html>\n<html>\n<head>\n    <title>E
xample Domain</title>\n\n    <meta charset="utf-8" />\n    <meta http-equi
v="Content-type" content="text/html; charset=utf-8" />\n    <meta name="vi
ewport" content="width=device-width, initial-scale=1" />\n    <style type=
"text/css">\n    body {\n        background-color: #f0f0f2;\n        margi
n: 0;\n        padding: 0;\n        font-family: -apple-system, system-ui,
 BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica,
 Arial, sans-serif;\n        \n    }\n    div {\n        width: 600px;\n
      margin: 5em auto;\n        padding: 2em;\n        background-color:
#fdfdff;\n        border-radius: 0.5em;\n        box-shadow: 2px 3px 7px 2
px rgba(0,0,0,0.02);\n    }\n    a:link, a:visited {\n        color: #3848
8f;\n        text-decoration: none;\n    }\n    @media (max-width: 700px)
{\n        div {\n            margin: 0 auto;\n            width: auto;\n
       }\n    }\n    </style>    \n</head>\n\n<body>\n<div>\n    <h1>Examp
le Domain</h1>\n    <p>This domain is for use in illustrative examples in
documents. You may use this\n    domain in literature without prior coordi
nation or asking for permission.</p>\n    <p><a href="https://www.iana.org
/domains/example">More information...</a></p>\n</div>\n</body>\n</html>\n'
]
B1-192.168.20.5:PES1UG20CS243:Mahika:/pyfiles
```

```
B2-192.168.20.6:PES1UG20CS243:Mahika:/pythonfile
#>python3 B1-B2-Socks-Client.py
[b'HTTP/1.0 200 OK', b'Age: 227107', b'Cache-Control: max-age=604800', b'C
ontent-Type: text/html; charset=UTF-8', b'Date: Sun, 06 Nov 2022 03:06:19
GMT', b'Etag: "3147526947+ident"', b'Expires: Sun, 13 Nov 2022 03:06:19 GM
T', b'Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT', b'Server: ECS (dcb/7E
EA)', b'Vary: Accept-Encoding', b'X-Cache: HIT', b'Content-Length: 1256',
b'Connection: close', b'', b'<!doctype html>\n<html>\n<head>\n    <title>E
xample Domain</title>\n\n    <meta charset="utf-8" />\n    <meta http-equi
v="Content-type" content="text/html; charset=utf-8" />\n    <meta name="vi
ewport" content="width=device-width, initial-scale=1" />\n    <style type=
"text/css">\n    body {\n        background-color: #f0f0f2;\n        margi
n: 0;\n        padding: 0;\n        font-family: -apple-system, system-ui,
 BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica,
 Arial, sans-serif;\n        \n    }\n    div {\n        width: 600px;\n
     margin: 5em auto;\n        padding: 2em;\n        background-color:
#fdfdff;\n        border-radius: 0.5em;\n        box-shadow: 2px 3px 7px 2
px rgba(0,0,0,0.02);\n    }\n    a:link, a:visited {\n        color: #3848
8f;\n        text-decoration: none;\n    }\n    @media (max-width: 700px)
{\n        div {\n            margin: 0 auto;\n            width: auto;\n
       }\n    }\n    </style>    \n</head>\n\n<body>\n<div>\n    <h1>Examp
le Domain</h1>\n    <p>This domain is for use in illustrative examples in
documents. You may use this\n    domain in literature without prior coordi
nation or asking for permission.</p>\n    <p><a href="https://www.iana.org
/domains/example">More information...</a></p>\n</div>\n</body>\n</html>\n'
]
B2-192.168.20.6:PES1UG20CS243:Mahika:/pythonfile
#>
```

HTTP request is sent, and information is received from the website. The html contents of the website are printed above.

To close the ssh shell that is in the background created using the previous “ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N” command :

 # ps -eaf | grep "ssh"

 # kill [corresponding pid of the process]

```
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>ps -eaf | grep "ssh"
root          40       1  0 Nov05 ?        00:00:00 sshd: /usr/sbin/sshd [listener] 0 of 10-100 startups
root          56      40  0 00:30 ?        00:00:00 sshd: root@pts/2
root         104       1  0 00:35 ?        00:00:00 ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N
root         108      50  0 02:21 pts/1    00:00:00 grep ssh
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>kill 104
B-192.168.20.99:PES1UG20CS243:Mahika:/pyfiles
#>
```

Task 3: Comparing SOCKS5 Proxy and VPN

Both SOCKS5 proxy (dynamic port forwarding) and VPN are commonly used in creating tunnels to bypass firewalls, as well as to protect communications. Many VPN service providers provide both types of services. Sometimes, when a VPN service provider tells you that it provides the VPN service, but in reality, it is just a SOCKS5 proxy. Although both technologies can be used to solve the same problem, they do have significant differences. Please compare these two technologies, describing their differences, pros and cons.

SOCKS PROXY:

- **In SOCKS5 proxy, your data is not encrypted, whereas in VPN traffic is encrypted.**
- **Due to data not being encrypted, SOCKS5 is much faster than VPN.**
- **VPN provides complete anonymity whereas SOCKS5 does not.**