

# Firewall Evasion Lab

Lab Environment Setup	2
Task 0 : Get Familiar with the Lab Setup	3
Task 1 : Static Port Forwarding	6
Task 2: Dynamic Port Forwarding	9
Task 2.1: Setting Up Dynamic Port Forwarding	9
Task 2.2: Testing the Tunnel Using Browser	12
Task 2.3: Writing a SOCKS Client Using Python	17
Task 3: Comparing SOCKS5 Proxy and VPN	20
Submission	<b>Error! Bookmark not defined.</b>

## Lab Environment Setup

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

[https://seedsecuritylabs.org/Labs\\_20.04/Networking/Firewall\\_Evasion/](https://seedsecuritylabs.org/Labs_20.04/Networking/Firewall_Evasion/)

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

```
[10/20/23]seed@VM:~/.../Labsetup$ docker-compose build
hostA1 uses an image, skipping
hostA2 uses an image, skipping
hostB uses an image, skipping
hostB1 uses an image, skipping
hostB2 uses an image, skipping
Router uses an image, skipping
Building hostA
Step 1/5 : FROM handsonsecurity/seed-ubuntu:large
--> cecb04fbf1dd
Step 2/5 : ARG DEBIAN_FRONTEND=noninteractive
--> Running in 583dbc81f608
Removing intermediate container 583dbc81f608
--> 24df9b5d1009
Step 3/5 : RUN apt-get update && apt-get -y install openssh-server && rm -rf /var/lib/apt/lists/*
--> Running in dde6047014fb
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [3145 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [29.3 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [2942 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [1117 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [32.0 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [3092 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [3633 kB]
[10/20/23]seed@VM:~/.../Labsetup$ docker-compose up
Creating network "net-10.8.0.0" with the default driver
Creating network "net-192.168.20.0" with the default driver
WARNING: Found orphan containers (host2-192.168.60.6, hostA-10.9.0.5, host3-192.168.60.7, host1-192.168.60.5) for this project. If you removed or
renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
Creating B-192.168.20.99 ... done
Creating A1-10.8.0.5 ... done
Creating B2-192.168.20.6 ... done
Creating A-10.8.0.99 ... done
Recreating seed-router ... done
Creating B1-192.168.20.5 ... done
Creating A2-10.8.0.6 ... done
Attaching to A1-10.8.0.5, B2-192.168.20.6, B-192.168.20.99, A-10.8.0.99, A2-10.8.0.6, B1-192.168.20.5, router-firewall
A-10.8.0.99 | * Starting internet superserver inetd [ OK ]
A-10.8.0.99 | * Starting OpenBSD Secure Shell server sshd [ OK ]
A1-10.8.0.5 | * Starting internet superserver inetd [ OK ]
A2-10.8.0.6 | * Starting internet superserver inetd [ OK ]
B-192.168.20.99 | * Starting internet superserver inetd [ OK ]
B-192.168.20.99 | * Starting OpenBSD Secure Shell server sshd [ OK ]
B1-192.168.20.5 | * Starting internet superserver inetd [ OK ]
B2-192.168.20.6 | * Starting internet superserver inetd [ OK ]
router-firewall | * Starting internet superserver inetd [ OK ]
```

Checking Router configuration

```
Router:PES2UG21CS283:Maryam:/
$>ip -br address
lo                UNKNOWN      127.0.0.1/8
eth0@if83         UP            10.8.0.11/24
eth1@if85         UP            192.168.20.11/24
Router:PES2UG21CS283:Maryam:/
$>
```

eth0@if83	UP	10.8.0.11/24	Private interface
eth1@if85	UP	192.168.20.11/24	Public interface

## 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.

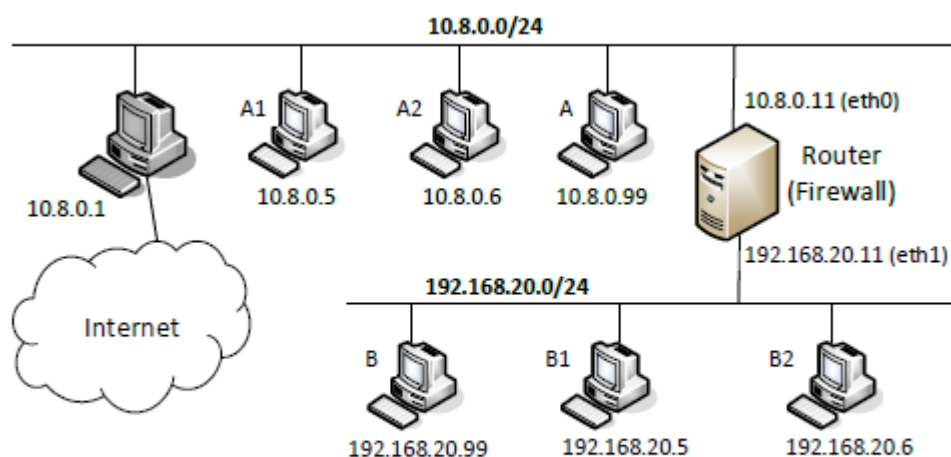


Figure 1: Network setup

**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.

```
// 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](http://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:**

These are the rules to block linkedin and miniclip websites

```
# 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:PES2UG21CS283:Maryam:/
$>iptables -A FORWARD -i eth1 -d 13.107.42.0/24 -j DROP
Router:PES2UG21CS283:Maryam:/
$>iptables -A FORWARD -i eth1 -d 13.249.221.0/24 -j DROP
Router:PES2UG21CS283:Maryam:/
$>
```

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
# ping www.miniclip.com
HostB1:PES2UG21CS283:Maryam:/
$>ping www.linkedin.com
PING 1-0005.1-msedge.net (13.107.42.14) 56(84) bytes of data.
^C
--- 1-0005.1-msedge.net ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2035ms

HostB1:PES2UG21CS283:Maryam:/
$>ping www.miniclip.com
PING www.miniclip.com (13.249.221.75) 56(84) bytes of data.
^C
--- www.miniclip.com ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3064ms

HostB1:PES2UG21CS283:Maryam:/
$>
```

100% data loss

## 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.

Three TCP connections are involved:

SSH connection from A to B.

Local socket connection from A to B's telnet port (initiated by A1)

Local socket connection from A to B's telnet port (initiated by A2)

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

The SSH tunnel operates over a single outbound SSH connection, which is often permitted through firewalls. It allows traffic from A1 and A2 to reach B's telnet service, bypassing firewall restrictions by using an encrypted and authorized channel through A to B.

**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
```

We try to evade the firewall using static and dynamic port forwarding

```
HostA1:PES2UG21CS283:Maryam:/
$>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
HostB: 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@HostB: ~$
```

```
HostA2:PES2UG21CS283:Maryam:/
$>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
HostB: 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.

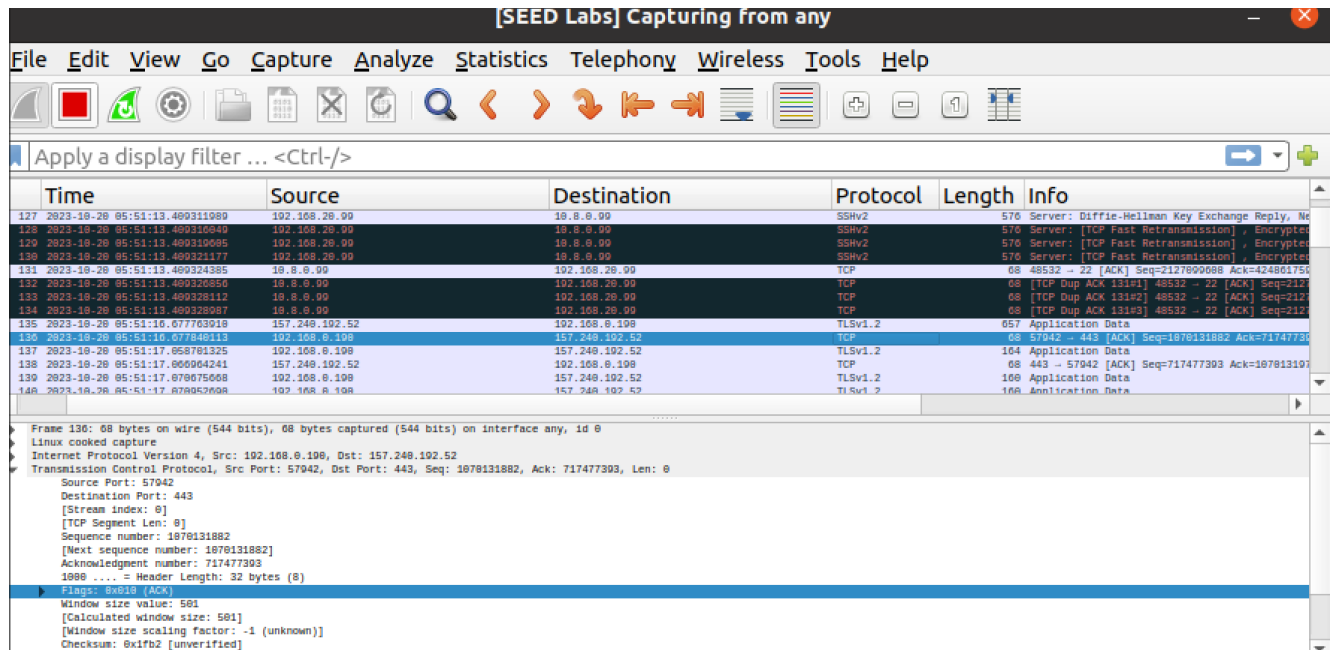
To restore this content, you can run the 'unminimize' command.

```
Last login: Fri Oct 20 09:52:04 UTC 2023 from HostB: on pts/3
seed@HostB: ~$
```



Run the below command on containers A1 and A2:

**# telnet 10.8.0.99 8000**



**[SEED Labs] Capturing from any**

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

Time	Source	Destination	Protocol	Length	Info
127	2023-10-20 05:51:13.400311080	10.8.0.99	SSHv2	576	Server: Diffie-Hellman Key Exchange Reply, No
128	2023-10-20 05:51:13.400316040	10.8.0.99	SSHv2	576	Server: [TCP Fast Retransmission], Encrypted
129	2023-10-20 05:51:13.400319000	10.8.0.99	SSHv2	576	Server: [TCP Fast Retransmission], Encrypted
130	2023-10-20 05:51:13.400321177	10.8.0.99	SSHv2	576	Server: [TCP Fast Retransmission], Encrypted
131	2023-10-20 05:51:13.400324385	10.8.0.99	TCP	68	48532 -> 22 [ACK] Seq=2117800008 Ack=424601758
132	2023-10-20 05:51:13.400326550	10.8.0.99	TCP	68	[TCP Dup ACK 131#1] 48532 -> 22 [ACK] Seq=2117
133	2023-10-20 05:51:13.400328112	10.8.0.99	TCP	68	[TCP Dup ACK 131#2] 48532 -> 22 [ACK] Seq=2117
134	2023-10-20 05:51:13.400329987	10.8.0.99	TCP	68	[TCP Dup ACK 131#3] 48532 -> 22 [ACK] Seq=2117
135	2023-10-20 05:51:16.677703019	157.248.192.52	TLSv1.2	657	Application Data
136	2023-10-20 05:51:16.677705110	157.248.192.52	TCP	68	57042 -> 443 [ACK] Seq=1878131882 Ack=717477303
137	2023-10-20 05:51:17.058701325	157.248.192.52	TLSv1.2	104	Application Data
138	2023-10-20 05:51:17.060604241	157.248.192.52	TCP	68	443 -> 57042 [ACK] Seq=717477303 Ack=187813190
139	2023-10-20 05:51:17.070675008	157.248.192.52	TLSv1.2	108	Application Data
140	2023-10-20 05:51:17.070675008	157.248.192.52	TLSv1.2	108	Application Data

Frame 136: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0

Linux cooked capture

Internet Protocol Version 4, Src: 10.8.0.99, Dst: 157.248.192.52

Transmission Control Protocol, Src Port: 57042, Dst Port: 443, Seq: 1878131882, Ack: 717477303, Len: 0

Source Port: 57042

Destination Port: 443

[Stream index: 0]

[TCP Segment Len: 0]

Sequence number: 1878131882

[Next sequence number: 1878131882]

Acknowledgment number: 717477303

1900 .... = Header Length: 32 bytes (8)

**1900 ... = Window (ACK)**

Window size value: 581

[Calculated window size: 581]

[Window size scaling factor: -1 (unknown)]

Checksum: 0x1fb2 [unverified]



## 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
root@HostB: :/# export PS1="HostB:PES2UG21CS283:Maryam:\w\n\${>}"
HostB:PES2UG21CS283:Maryam:/
$>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:e7S8ExtXMqIQpEsbJlLhwuEM5c3JA80vuos4qbK1xKk.
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:
HostB:PES2UG21CS283:Maryam:/
$> █
```

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 the 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:

**We are able to visit the websites (linkedin and miniclip) that had been blocked**

```
HostB:PES2UG21CS283:Maryam:/
$>curl -x socks5h://0.0.0.0:8000 https://www.linkedin.com
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <meta name="pageKey" content="d_homepage-guest-home">
<!-->
  <meta name="locale" content="en_US">
  <meta id="config" data-app-version="2.1.1032" data-call-tree-id="AAYII2XcDa0QPFTx3o4PEw==" data-jet-tags="guest-homepage" data-multiprodu
ct-name="homepage-guest-frontend" data-service-name="homepage-guest-frontend" data-browser-id="1b505aab-df44-4c75-80fc-8c02bc01c7fd" data-enable-
page-view-heartbeat-tracking data-page-instance="urn:li:page:d_homepage-guest-home;xXkE7ZK1SsmI+TXtYgzH0g==" data-disable-jsbeacon-pagekey-suffix
="false" data-member-id="0">

  <link rel="canonical" href="https://www.linkedin.com/">
  <link rel="alternate" hreflang="de" href="https://de.linkedin.com/">
  <link rel="alternate" hreflang="en-IE" href="https://ie.linkedin.com/">
  <link rel="alternate" hreflang="en-US" href="https://www.linkedin.com/">
  <link rel="alternate" hreflang="pt" href="https://pt.linkedin.com/">
  <link rel="alternate" hreflang="en-IL" href="https://il.linkedin.com/">
  <link rel="alternate" hreflang="ms-MY" href="https://my.linkedin.com/">
  <link rel="alternate" hreflang="en-IN" href="https://in.linkedin.com/">
  <link rel="alternate" hreflang="es-BO" href="https://bo.linkedin.com/">
  <link rel="alternate" hreflang="en-ZA" href="https://za.linkedin.com/">
  <link rel="alternate" hreflang="zh-CN" href="https://cn.linkedin.com/">
  <link rel="alternate" hreflang="en-AU" href="https://au.linkedin.com/">
  <link rel="alternate" hreflang="id" href="https://id.linkedin.com/">
  <link rel="alternate" hreflang="en-NG" href="https://ng.linkedin.com/">
  <link rel="alternate" hreflang="de-CH" href="https://ch.linkedin.com/">
  <link rel="alternate" hreflang="ja-JP" href="https://jp.linkedin.com/">
  <link rel="alternate" hreflang="en-ZW" href="https://zw.linkedin.com/">
  <link rel="alternate" hreflang="en-JM" href="https://jm.linkedin.com/">
  <link rel="alternate" hreflang="es-SV" href="https://sv.linkedin.com/">
```

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

Host B (the proxy) establishes the actual connection with the intended web server.

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

The computer knows which server to connect to based on the URL specified in the curl command. The URL tells the curl command the target server to request.

Run in container B:

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

```
HostB:PES2UG21CS283:Maryam:/
$>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;
```

We are able to access the example.com domain

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>**

```
HostB1:PES2UG21CS283:Maryam:/
$>curl -x socks5h://192.168.20.99:8080 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;
    }
  </style>
</head>
<body>
  <div>
    <h1>Example Domain</h1>
    <p>This domain does not have any content. Please click the link below to visit the site, or you can search for it to find the content you want to view.</p>
    <a href="http://www.example.com">www.example.com</a>
  </div>
</body>
</html>
```

```
HostB2:PES2UG21CS283:Maryam:/
$>curl -x socks5h://192.168.20.99:8080 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;
    }
  </style>
</head>
<body>
  <div>
    <h1>Example Domain</h1>
    <p>This domain does not have any content. Please click the link below to visit the site, or you can search for it to find the content you want to view.</p>
    <a href="http://www.example.com">www.example.com</a>
  </div>
</body>
</html>
```

## **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.

proxy

### Connection Settings

**Configure Proxy Access to the Internet**

☐ No proxy

☐ Auto-detect proxy settings for this network

☐ Use system proxy settings

☒ Manual proxy configuration

HTTP Proxy  Port

☐ Also use this proxy for FTP and HTTPS

HTTPS Proxy  Port

FTP Proxy  Port

SOCKS Host  Port

☐ SOCKS v4 ☒ SOCKS v5

☐ Automatic proxy configuration URL

No proxy for

Example: .mozilla.org, .net.nz, 192.168.1.0/24

Connections to localhost, 127.0.0.1, and ::1 are never proxied.

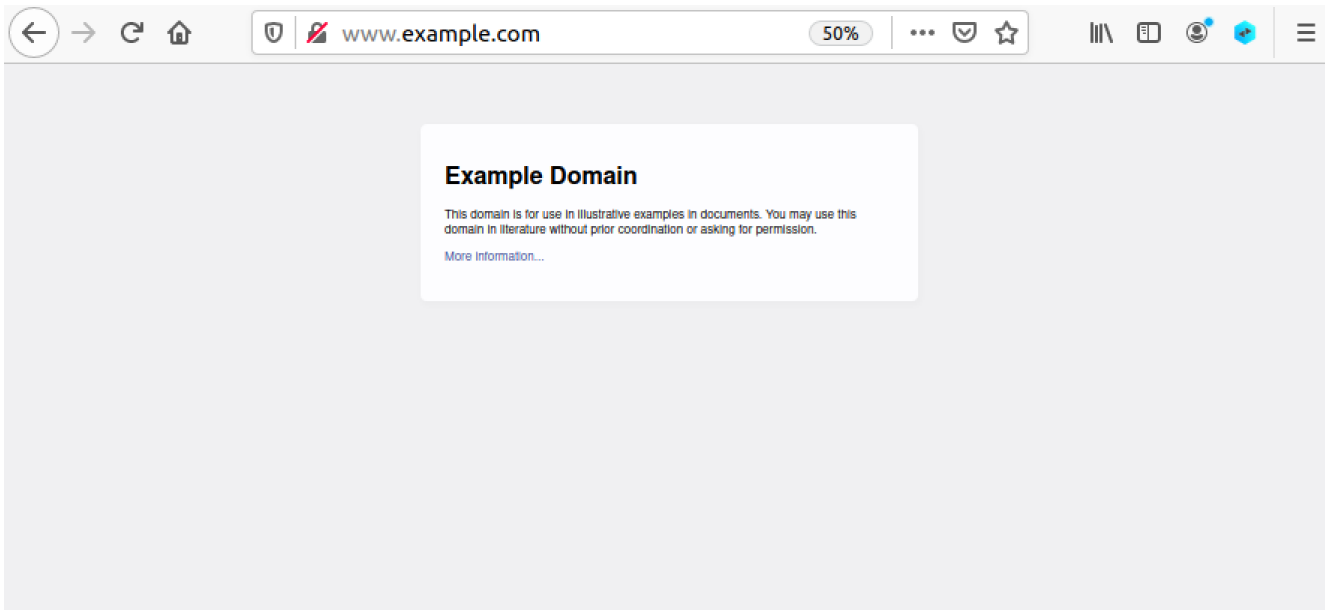
☒ Do not prompt for authentication if password is saved

☒ Proxy DNS when using SOCKS v5

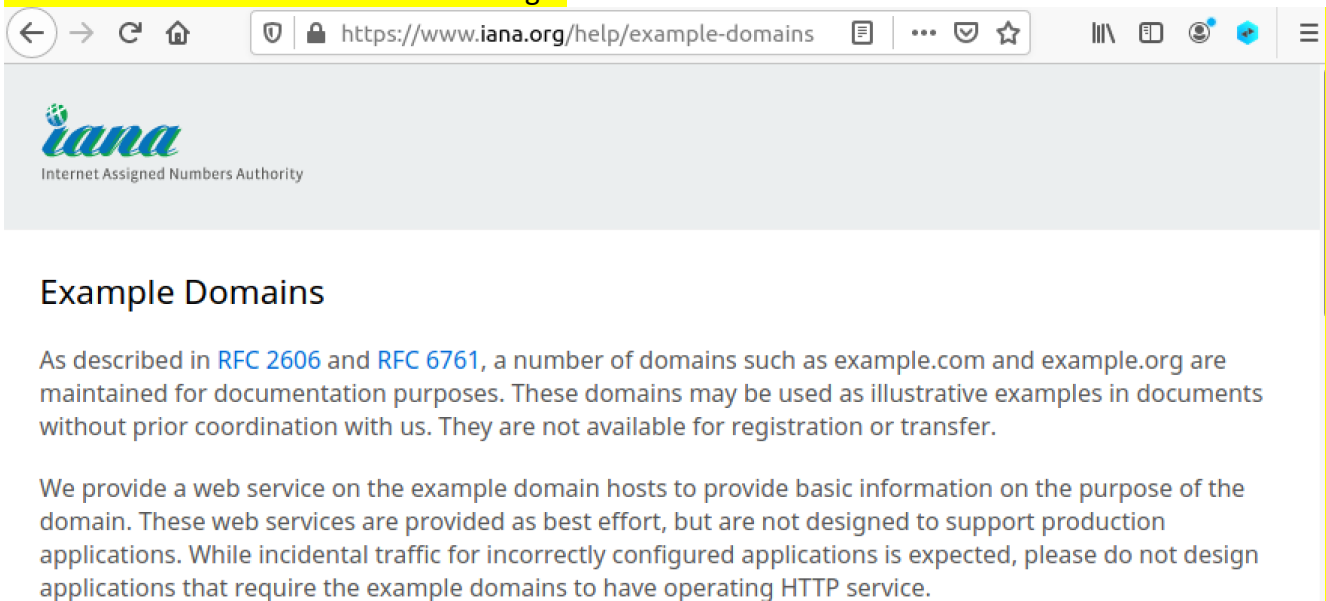
☒ Enable DNS over HTTPS

Use Provider

**Lab task.** Once the proxy is configured, we can then browse any website. The requests and replies will



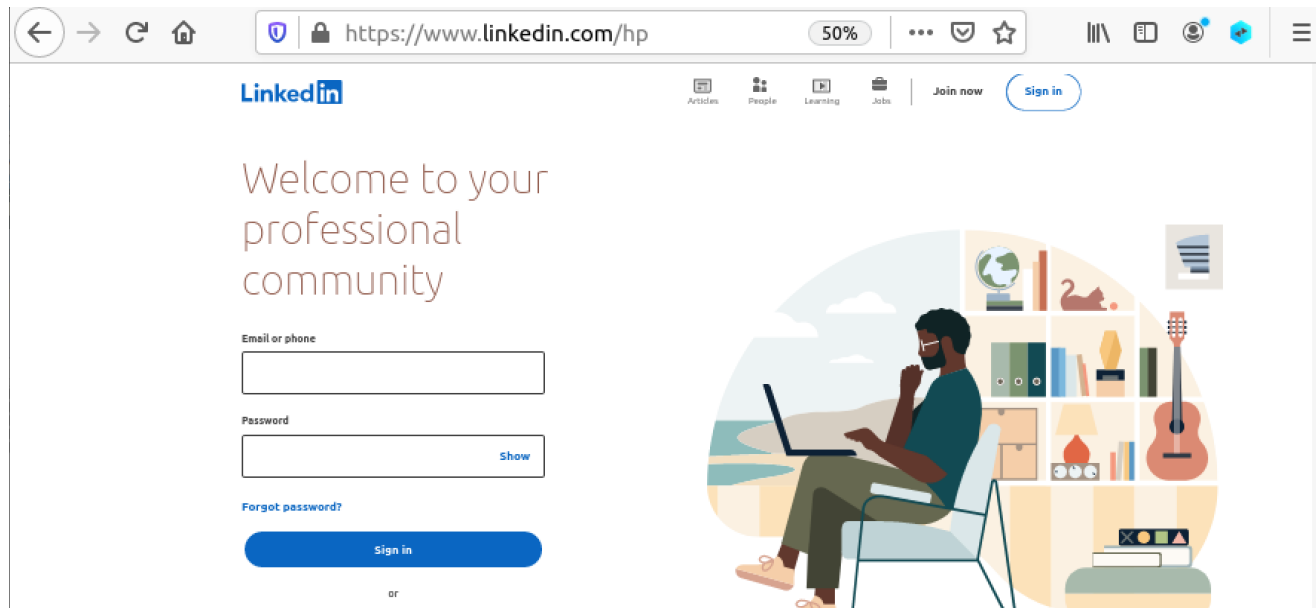
When we click on more information we get



Testing the tunnel through the browser works.

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:

We test the browser for website [www.linkedin.com](https://www.linkedin.com)



- (1) run tcpdump on the router-firewall, and point out the traffic involved in the entire port forwarding process.
- (2) Break the SSH tunnel, and then try to browse a website. Describe your observation.

**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:**

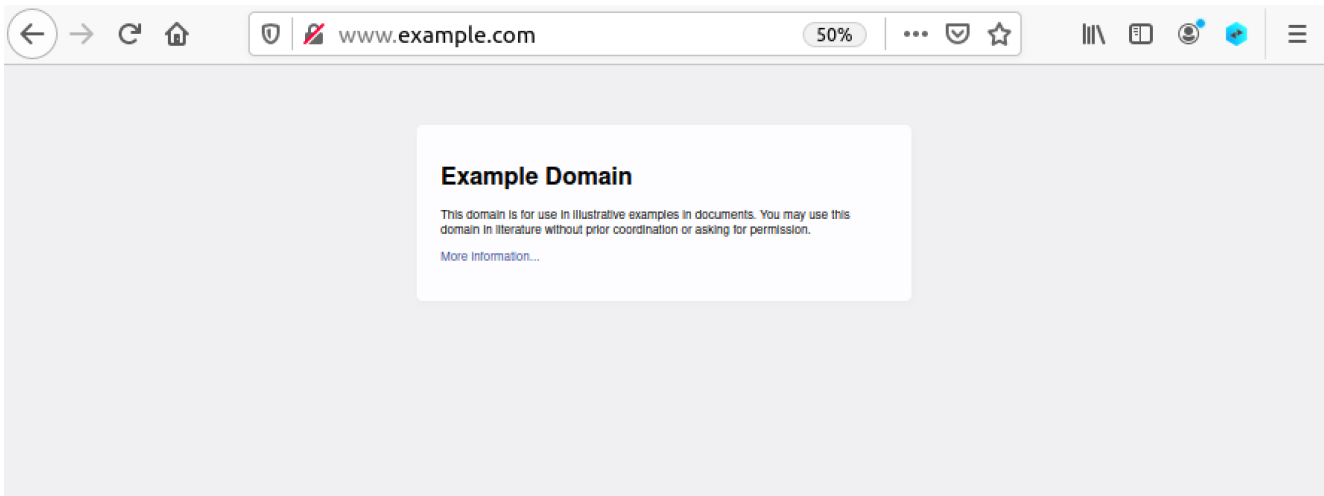
```
# ps -eaf | grep "ssh"
# kill [corresponding pid of the process]
```

Now try to access the websites again and provide your observations.

**We kill the ssh process and try accessing the website**

```
HostB:PES2UG21CS283:Maryam:/
$>ps -eaf | grep "ssh"
root      40      1  0 09:09 ?        00:00:00 sshd: /usr/sbin/sshd [listener] 0 of 10-100 startups
root      96      1  0 10:22 ?        00:00:00 ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N
root     100     42  0 11:02 pts/1    00:00:00 grep ssh
HostB:PES2UG21CS283:Maryam:/
$>kill 96
HostB:PES2UG21CS283:Maryam:/
$>
```





The proxy server refuses the connections and we are not able to access the website

Cleanup. After this task, please make sure to remove the proxy setting from Firefox by checking the "No proxy" option. Without a proper cleanup, future labs may be affected.

## 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
```

```
HostB: PES2UG21CS283: Maryam: /
$> ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N
root@10.8.0.99's password:
HostB: PES2UG21CS283: Maryam: /
$> 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);
    }
  </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 illustrations of hypothetical documents; however, you must not claim ownership of Example Domain. For more information on the history and future of Example Domain, please see the related historical documents: <a href="https://www.iana.org/domains/example">https://www.iana.org/domains/example</a> and <a href="https://www.iana.org/domains/example">https://www.iana.org/domains/example</a>.
```



```
HostB2:PES2UG21CS283:Maryam:/
$>nano B1-B2-sock-client.py
HostB2:PES2UG21CS283:Maryam:/
$>python3 B1-B2-sock-client.py
[b'HTTP/1.0 200 OK', b'Accept-Ranges: bytes', b'Age: 37257', b'Cache-Control: max-age=604800', b'Content-Type: text/html; charset=UTF-8', b'Date:
Fri, 20 Oct 2023 11:29:11 GMT', b'Etag: "3147526947+gzip"', b'Expires: Fri, 27 Oct 2023 11:29:11 GMT', b'Last-Modified: Thu, 17 Oct 2019 07:18:2
6 GMT', b'Server: ECS (nyb/1000)', b'Vary: Accept-Encoding', b'X-Cache: HIT', b'Content-Length: 1256', b'Connection: close', b'', b'<!doctype htm
l>\n<html>\n<head>\n  <title>Example Domain</title>\n\n  <meta charset="utf-8" />\n  <meta http-equiv="Content-type" content="text/html; ch
arset=utf-8" />\n  <meta name="viewport" content="width=device-width, initial-scale=1" />\n  <style type="text/css">\n    body {\n      bac
kground-color: #f0f0f2;\n      margin: 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: #f0f0f2;\n      border-radius: 0.5em;\n      box-shadow: 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 illustrative examples in documents. You may use this\n  domain in literature without prior coordination or asking f
or permission.</p>\n  <p><a href="https://www.iana.org/domains/example">More information...</a></p>\n</div>\n</body>\n</html>\n']
HostB2:PES2UG21CS283:Maryam:/
$>
```

We get the same output in B1 and B2 that we got in B

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]**

```
HostB:PES2UG21CS283:Maryam:/
$>ps -eaf | grep "ssh"
root      40      1  0 09:09 ?        00:00:00 sshd: /usr/sbin/sshd [listener] 0 of 10-100 startups
root      96      1  0 10:22 ?        00:00:00 [ssh] <defunct>
root     103      1  0 11:15 ?        00:00:00 ssh -4 -D 0.0.0.0:8000 root@10.8.0.99 -f -N
root     111     42  0 11:30 pts/1    00:00:00 grep ssh
HostB:PES2UG21CS283:Maryam:/
$>kill 103
HostB:PES2UG21CS283:Maryam:/
$>
```

## 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.

SOCKS5 proxies and VPNs are both technologies used for tunneling and protecting online communication, but they have distinct differences, advantages, and disadvantages:

SOCKS5 Proxy:

Pros:

1. **Speed:** SOCKS5 proxies are often faster because they don't encrypt data by default. They are ideal for tasks like bypassing geo-restrictions or circumventing firewalls.
2. **Application-Level Proxy:** SOCKS5 can be set up at the application level. This means you can configure specific applications or services to use the proxy, while others do not, providing more granular control.
3. **Simple Setup** Setting up a SOCKS5 proxy is relatively easy and doesn't require the installation of specialized software.
4. **No Encryption Overhead:** SOCKS5 doesn't add encryption overhead, which makes it suitable for tasks where speed is crucial.

Cons:

1. **Limited Security:** SOCKS5 proxies provide minimal security. They don't encrypt your data, so it's not suitable for securing sensitive information or privacy.
2. **Limited Anonymity:** While they can hide your IP address, SOCKS5 proxies do not provide the same level of anonymity as VPNs.
3. **Single Application:** SOCKS5 is typically configured on a per-application basis. You need to set up each application individually if you want to use the proxy.

VPN (Virtual Private Network):

Pros:

1. **Strong Encryption:** VPNs encrypt all your internet traffic, providing a high level of security and privacy, making them suitable for sensitive data and online anonymity.
2. **Network-Wide:** A VPN secures your entire network connection, not just specific applications. This means all your internet traffic is protected without configuring each app separately.
3. **Geographic Flexibility:** VPNs can help you bypass geo-restrictions, similar to SOCKS5 proxies.
4. **Diverse Server Locations:** VPNs often have a larger network of servers in multiple locations, which can provide improved access and performance.

Cons:

1. Speed: VPNs can be slower than SOCKS5 proxies due to the encryption and routing overhead.
2. Complex Setup: Setting up a VPN can be more complex than a SOCKS5 proxy, often requiring the installation of dedicated software.
3. Potential Costs: VPNs may require a subscription fee to access reliable services.
4. Resource Intensive: VPNs can consume more system resources due to encryption and decryption processes.

SOCKS5 proxies are faster and suitable for specific tasks like bypassing firewalls or accessing geo-restricted content, but they lack the robust security and privacy features of VPNs. VPNs are more comprehensive, offering network-wide security and privacy, but they can be slower and more complex to set up

Aspect	SOCKS5 Proxy	VPN (Virtual Private Network)
Encryption	Minimal or None	Strong Encryption
Security	Low (No Encryption)	High (Data Encrypted)
Privacy	Limited	High (Provides Anonymity)
Speed	Fast (No Encryption)	Slower (Due to Encryption)
Application Control	Per Application	Network-Wide
Complexity	Simple Setup	May Require Software Installation
Anonymity	Basic IP Masking	High Level of Anonymity
Cost	Often Free or Lower Cost	Subscription-Based Services
Use Cases	Bypassing Firewalls, Geo-Restrictions	Secure Data, Privacy, Anonymity
Resource Usage	Light on System Resources	Heavier Resource Consumption
Setup Granularity	Configured Per App/Service	Network-Wide Security