The Laboratory of Computer Networks Security

(UE19CS326)

Documented by Anurag.R.Simha

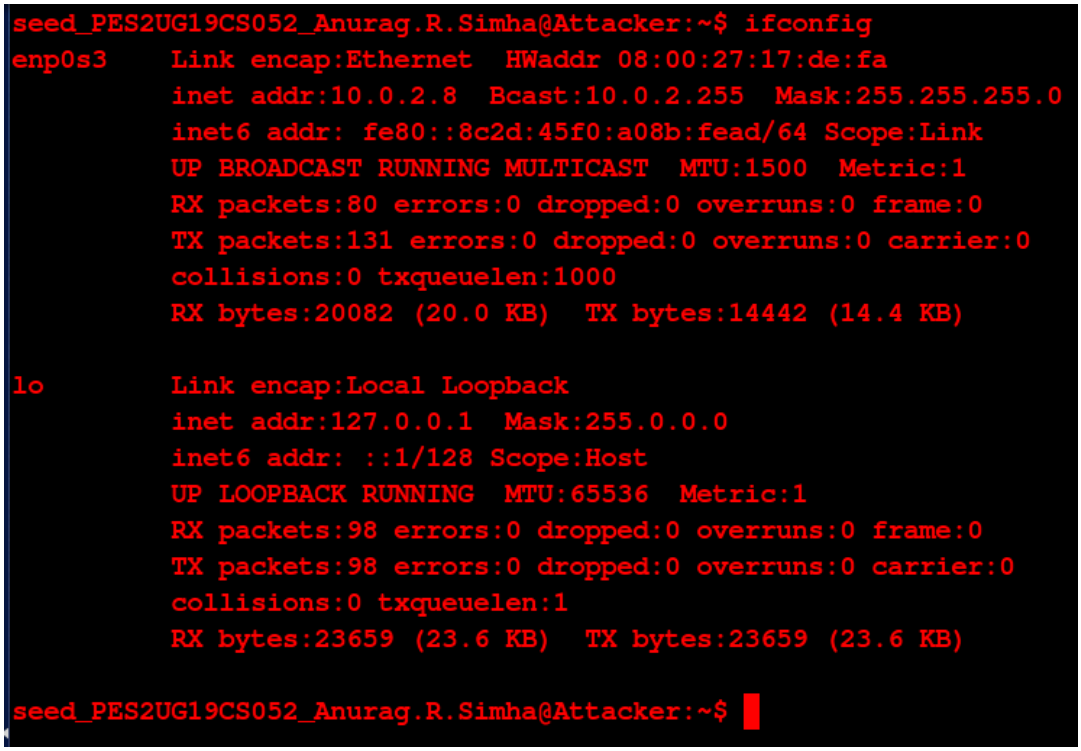
SRN	:	PES2UG19CS052
Name	:	Anurag.R.Simha
Date	:	18/10/2021
Section	:	A
Week	:	4

The Table of Contents

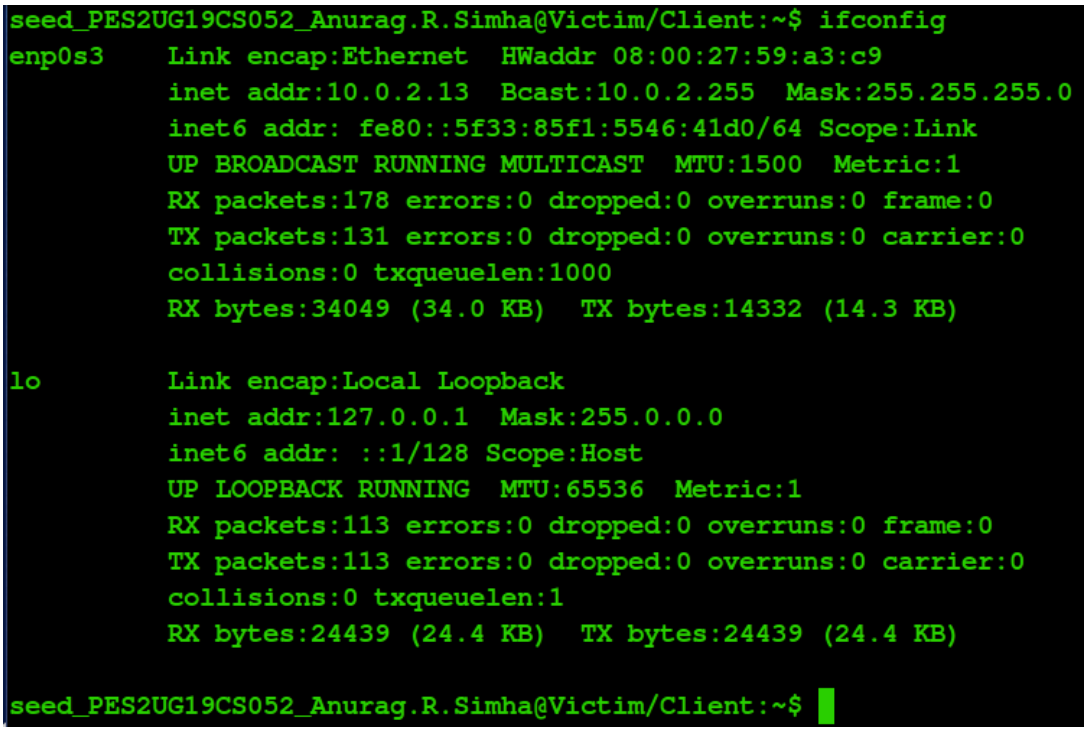
The Setup

For the experimentation of various attacks, three virtual machines were employed.

1. The Attacker machine (10.0.2.8) [VM 3]



2. The Victim/Client machine (10.0.2.13) [VM 1]



3. The DNS Server machine (10.0.2.14) [VM 2]

```
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:70:0c:00
          inet addr:10.0.2.14  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::6839:90ab:7428:5dec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:122 errors:0 dropped:0 overruns:0 frame:0
          TX packets:125 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:25764 (25.7 KB)  TX bytes:13692 (13.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:102 errors:0 dropped:0 overruns:0 frame:0
          TX packets:102 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:23927 (23.9 KB)  TX bytes:23927 (23.9 KB)

seed_PES2UG19CS052_Anurag.R.Simha@Server:~$
```

Task 1: Using Firewall

Firstly, a telnet communication between VM 1 (10.0.2.13) and VM 2 (10.0.2.14) is attempted.

The command: `telnet 10.0.2.14`

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:59:a3:c9
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::5f33:85f1:5546:41d0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:37 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5755 (5.7 KB)  TX bytes:7506 (7.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:69 errors:0 dropped:0 overruns:0 frame:0
          TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:21344 (21.3 KB)  TX bytes:21344 (21.3 KB)

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ telnet 10.0.2.14
Trying 10.0.2.14...
Connected to 10.0.2.14.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
```

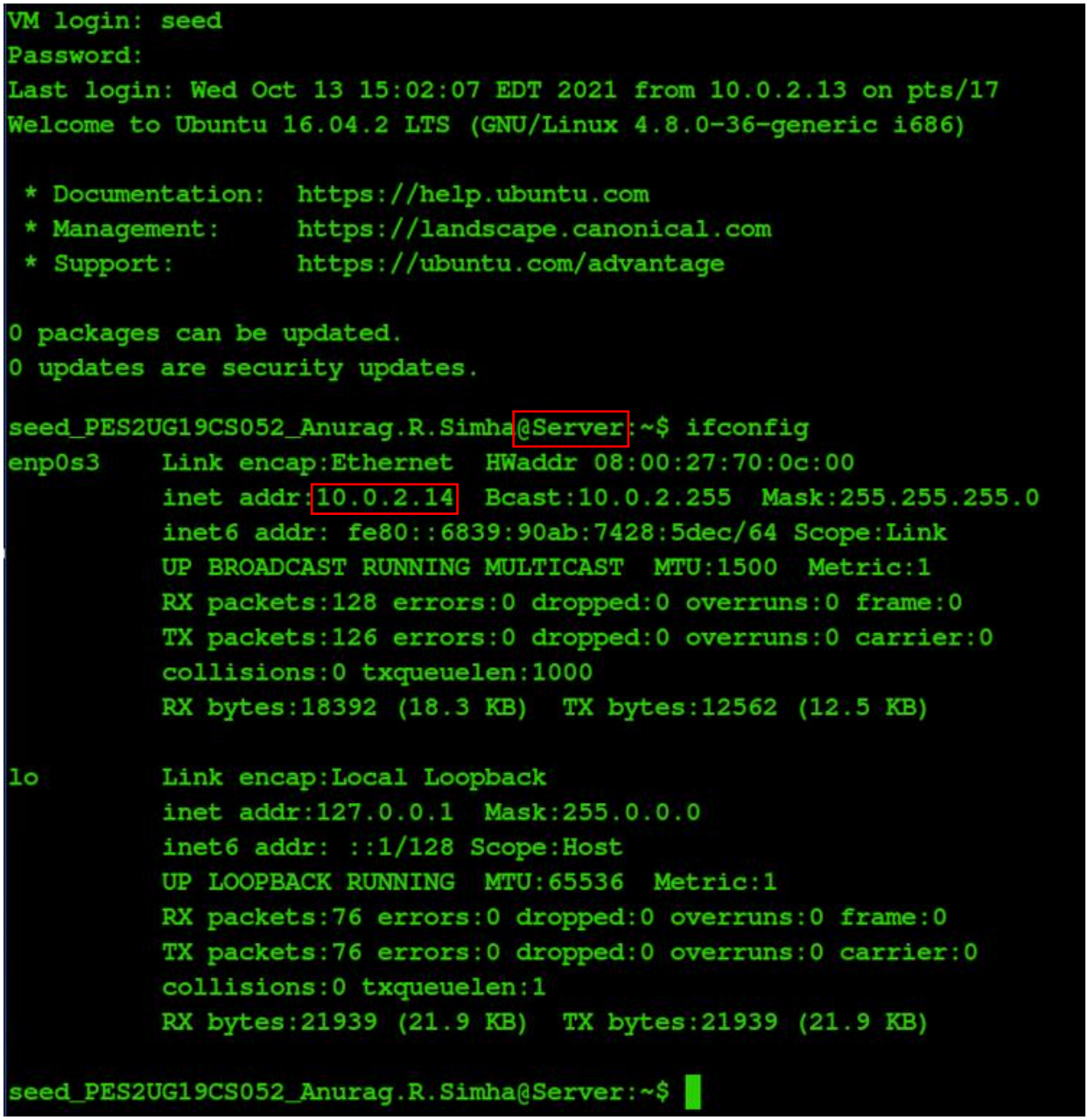


Fig. 1(a): Communicating over telnet to 10.0.2.14 (VM 2) from 10.0.2.13 (VM 1).

The sine qua non here is to foil VM 1 from contacting VM 2. With the aid of *ufw*, the firewall is primarily enabled (on VM 1).

The ufw tool quashes any incoming connections. So, a new rule was added on both the machines.

The commands:

```
sudo ufw enable
sudo ufw default allow incoming
sudo ufw status verbose
```

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw enable
Firewall is active and enabled on system startup
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 1(b): Enabling *ufw* on VM 1 and investigating the status of verbose.

Verbose is active and is running with *ufw* enabled.

Next, the firewall on VM 1 to deny telnet (port 23) to VM2 is configured.

The commands:

`sudo ufw deny out from 10.0.2.13 to 10.0.2.14 port 23`

`sudo ufw status verbose`

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw deny out from 10.0.2.13 to 10.0.2.14 port 23
Rule added
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
10.0.2.14 23               DENY OUT    10.0.2.13

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 1(c): The new rule is added to the rules table.

Once again, the telnet connection from VM 1 (10.0.2.13) to VM 2 (10.0.2.14) is attempted.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ telnet 10.0.2.14
Trying 10.0.2.14...
telnet: Unable to connect to remote host: Connection timed out
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 1(d): The activation of the firewall results in a failed telnet connection

From figure 1(d), it's limpid that the activation of a firewall foils the communication between 10.0.2.13 (VM 1) and 10.0.2.14 (VM 2).

Next, it's desired to telnet from VM 2 to VM 1. Initially, the connection is tested.

The command: `telnet 10.0.2.13`

```
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$ telnet 10.0.2.13
Trying 10.0.2.13...
Connected to 10.0.2.13.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue Oct 19 02:24:09 EDT 2021 from 10.0.2.14 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:59:a3:c9
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::5f33:85f1:5546:41d0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:627 errors:0 dropped:0 overruns:0 frame:0
          TX packets:562 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:55273 (55.2 KB)  TX bytes:49163 (49.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:234 errors:0 dropped:0 overruns:0 frame:0
          TX packets:234 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:29853 (29.8 KB)  TX bytes:29853 (29.8 KB)
```

Fig. 1(e): A triumphant telnet connection to 10.0.2.13 (VM 1) from 10.0.2.14 (VM 2) is observed.

Now, the telnet connection is blocked.

The command:

```
sudo ufw delete 1
sudo ufw deny in from 10.0.2.14 to 10.0.2.13 port 23
```

The first command in the list erases the entry previously made on the table in the client machine and the second command denies the incoming telnet connection from VM 2 to VM 1.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw delete 1
Deleting:
 deny out from 10.0.2.13 to 10.0.2.14 port 23
Proceed with operation (y|n)? y
Rule deleted
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw deny in from 10.0.2.14 to 10.0.2.13 port 23
Rule added
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
10.0.2.13 23               DENY IN     10.0.2.14

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 1(f): Deleting the previous entry and adding a new denial.

After erasing the old entry from the table, the desired new entry is added (on VM 1).

```
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$ telnet 10.0.2.13
Trying 10.0.2.13...
telnet: Unable to connect to remote host: Connection timed out
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$
```

Fig. 1(g): The connection to 10.0.2.13 (VM 1) from 10.0.2.14 (VM 2) has failed.

The opposite connection is possible.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ telnet 10.0.2.14
Trying 10.0.2.14...
Connected to 10.0.2.14.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue Oct 19 02:19:13 EDT 2021 from 10.0.2.13 on pts/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.
```

```
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

seed_PES2UG19CS052_Anurag.R.Simha@Server:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:70:0c:00
          inet addr:10.0.2.14  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::6839:90ab:7428:5dec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:606 errors:0 dropped:0 overruns:0 frame:0
          TX packets:715 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:57494 (57.4 KB)  TX bytes:58619 (58.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:296 errors:0 dropped:0 overruns:0 frame:0
          TX packets:296 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:32840 (32.8 KB)  TX bytes:32840 (32.8 KB)
```

Fig. 1(h): The connection to 10.0.2.14 (VM 2) from 10.0.2.13 (VM 1) is successful.

VM 1 is now desired to be blocked from visiting a website. In this case, the targetted website is www.pes.edu. The IP address is first obtained.

The command: `ping www.pes.edu`

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ ping www.pes.edu
PING waws-prod-pn1-007.cloudapp.net (52.172.204.196) 56(84) bytes of data.
^C
--- waws-prod-pn1-007.cloudapp.net ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4093ms

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 1(i): The ping result.

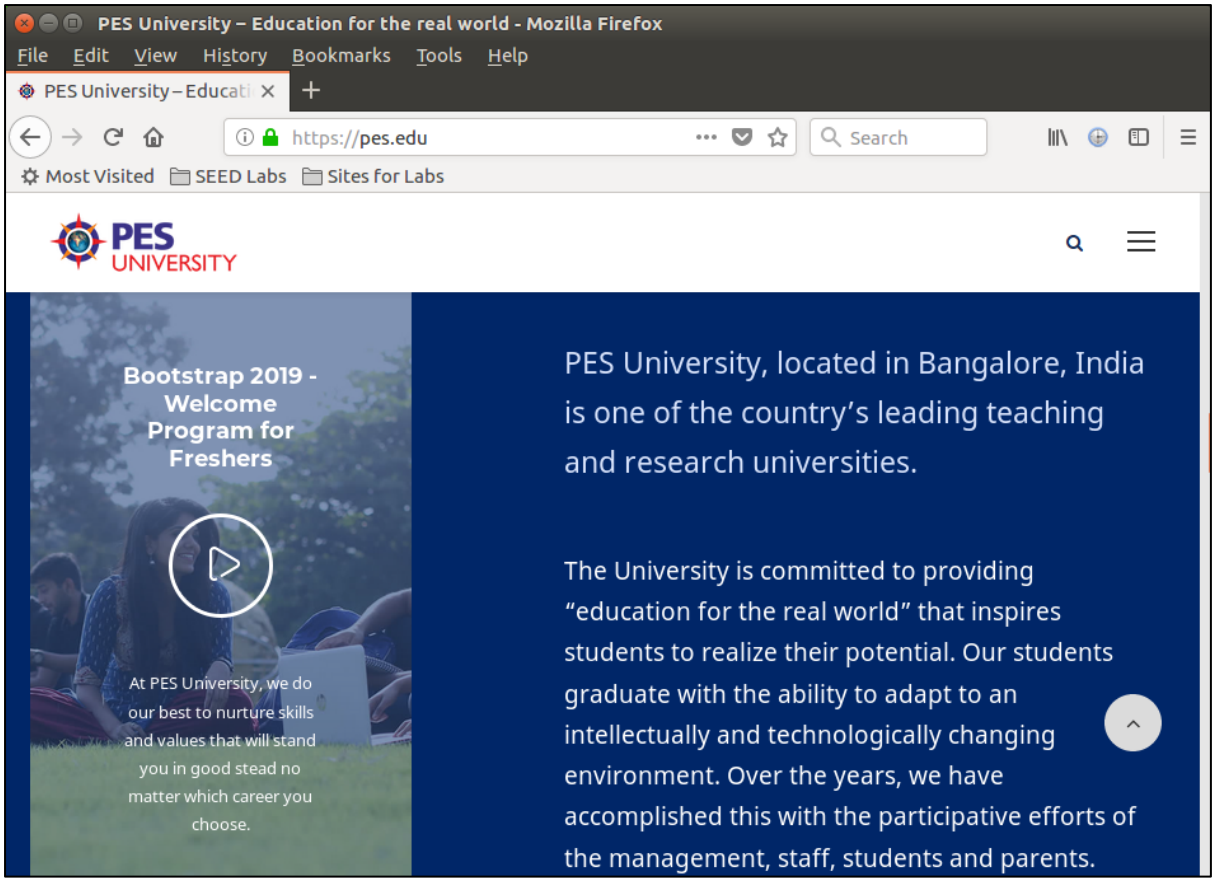


Fig. 1(j): Access to the website is allowed.

Before proceeding further, the cache is cleared.

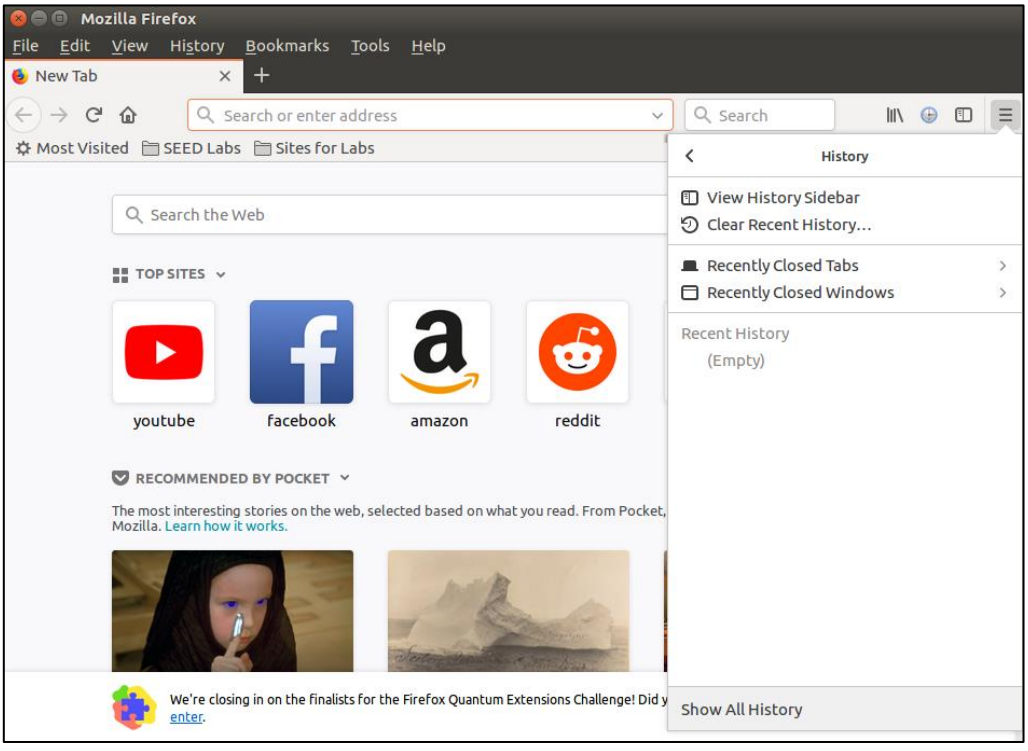


Fig. 1(k): The cache is cleared.

Next, firewall rule is added to prevent VM1 from accessing the IP address for www.pes.edu (52.172.204.196)

The commands:

```
sudo ufw delete 1
sudo ufw deny out to 52.172.204.196
sudo ufw status verbose
```

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw delete 1
Deleting:
 deny from 10.0.2.14 to 10.0.2.13 port 23
Proceed with operation (y|n)? y
Rule deleted
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw deny out to 52.172.204.196
Rule added
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
52.172.204.196             DENY OUT    Anywhere

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 1(*l*): The firewall entry is made.

With the firewall rule in place, a ping to www.pes.edu is attempted. It's observed that a message labelled, "Operation not permitted" appears. This is due to the blocking action of the firewall.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ ping www.pes.edu
PING waws-prod-pn1-007.cloudapp.net (52.172.204.196) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- waws-prod-pn1-007.cloudapp.net ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4100ms

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 1(m): The firewall blocks access to www.pes.edu

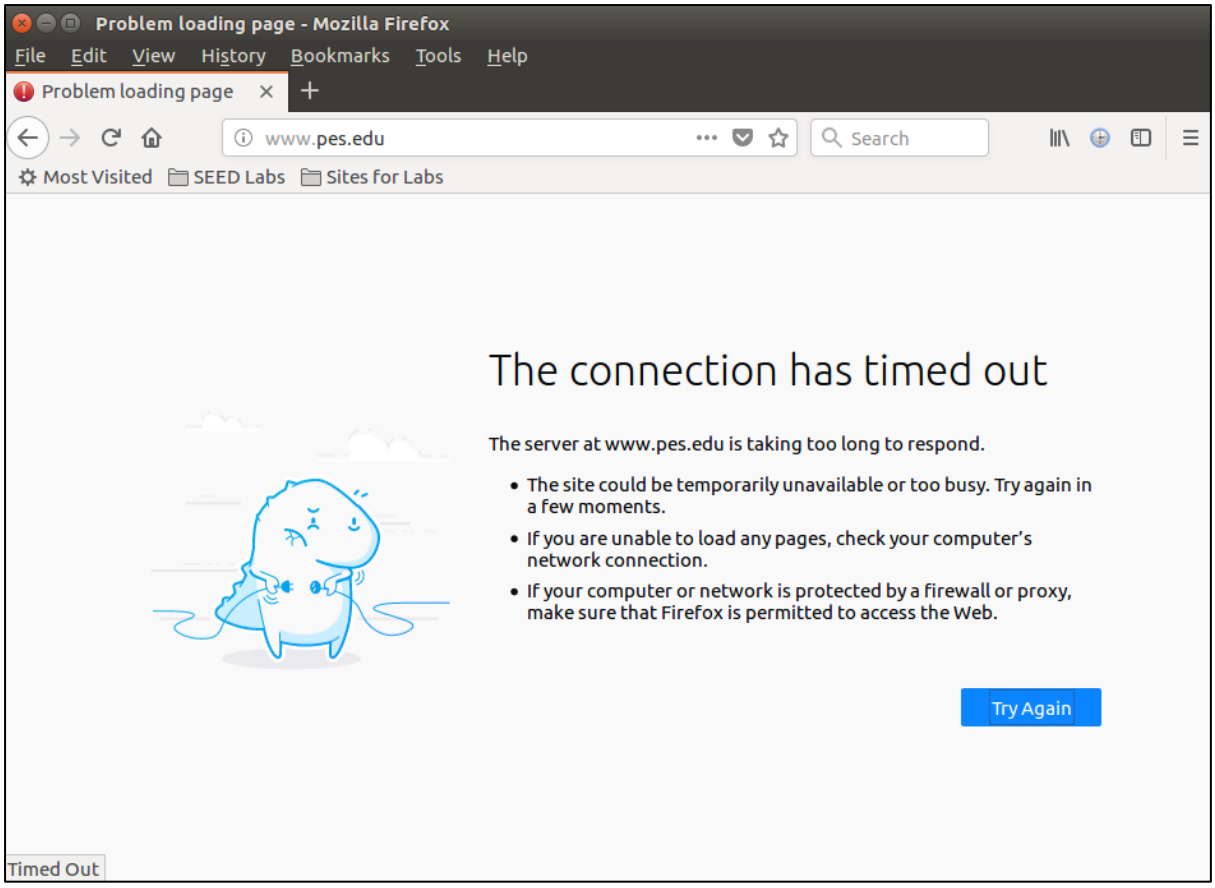


Fig. 1(n): The domain's inaccessible also.

It's observed that the firewall blocks accessing the website.

Task 2: How the Firewall Works

In this task, a firewall is developed using netfilter and LKM. The five rules implemented in this firewall are:

- Block telnet from VM1 to VM2
- Block telnet from VM2 to VM1
- Block external website access from VM1
- Block SSH from VM1 to VM2
- Block SSH from VM2 to VM1

Below is the programme for the firewall:

Name: `lkm.c`

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
```

```
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/inet.h>

#define NIPQUAD(addr) ((unsigned char *)&addr)[0], ((unsigned char
*)&addr)[1], ((unsigned char *)&addr)[2], ((unsigned char *)&addr)[3]

static struct nf_hook_ops nfho;
static struct nf_hook_ops nfho1;
static struct nf_hook_ops nfho2;
static struct nf_hook_ops nfho3;
static struct nf_hook_ops nfho4;

unsigned int telnet_outgoing(void *priv, struct sk_buff *skb, const struct
nf_hook_state *state)
{
    struct iphdr *iph;
    struct tcphdr *tcph;

    iph = ip_hdr(skb);
    tcph = (void *)iph+iph->ihl*4;

    if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23) && iph->saddr
== in_aton("10.0.2.13") && iph->daddr==in_aton("10.0.2.14")) {
        printk(KERN_INFO "Dropping Telnet Packet to destination address:
%d.%d.%d.%d\n",NIPQUAD(iph->daddr));
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}

unsigned int ssh_outgoing(void *priv, struct sk_buff *skb, const struct
nf_hook_state *state)
{
    struct iphdr *iph;
    struct tcphdr *tcph;

    iph = ip_hdr(skb);
    tcph = (void *)iph+iph->ihl*4;


    if (iph->protocol == IPPROTO_TCP &&  tcph->dest == htons(22) && iph->saddr
== in_aton("10.0.2.13") && iph->daddr==in_aton("10.0.2.14")) {
        printk(KERN_INFO "Dropping SSH Packet to destination address:
%d.%d.%d.%d\n",NIPQUAD(iph->daddr));
        return NF_DROP;
    } else {
        return NF_ACCEPT;
```

```
    }
}

unsigned int telnet_incoming(void *priv, struct sk_buff *skb, const struct
nf_hook_state *state)
{
    struct iphdr *iph;
    struct tcphdr *tcph;

    iph = ip_hdr(skb);
    tcph = (void *)iph+iph->ihl*4;

    if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23) && iph->saddr
== in_aton("10.0.2.14") && iph->daddr==in_aton("10.0.2.13")) {
        printk(KERN_INFO "Dropping Telnet Packet from source address:
%d.%d.%d.%d\n",NIPQUAD(iph->saddr));
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}

unsigned int ssh_incoming(void *priv, struct sk_buff *skb, const struct
nf_hook_state *state)
{
    struct iphdr *iph;
    struct tcphdr *tcph;

    iph = ip_hdr(skb);
    tcph = (void *)iph+iph->ihl*4;

    if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(22) && iph->saddr
== in_aton("10.0.2.14") && iph->daddr==in_aton("10.0.2.13")) {
        printk(KERN_INFO "Dropping SSH Packet from source address:
%d.%d.%d.%d\n",NIPQUAD(iph->saddr));
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}

unsigned int web_block(void *priv, struct sk_buff *skb, const struct
nf_hook_state *state)
{
    struct iphdr *iph;
    struct tcphdr *tcph;

    iph = ip_hdr(skb);
```

```
    tcph = (void *)iph+iph->ihl*4;

    if (iph->protocol == IPPROTO_TCP && iph->saddr == in_aton("10.0.2.13") &&
iph->daddr==in_aton("52.172.204.196") && (tcph->dest == htons(80) || tcph-
>dest == htons(443)) ) {
        printk(KERN_INFO "Dropping Web Packet to web page on address:
%d.%d.%d.%d\n",NIPQUAD(iph->daddr));
        return NF_DROP;
    } else {
        return NF_ACCEPT;
    }
}

int init_module()
{
    nfho.hook = telnet_outgoing; /* Handler function */
    nfho.hooknum = NF_INET_LOCAL_OUT;
    nfho.pf = PF_INET;
    nfho.priority = NF_IP_PRI_FIRST; /* Make our function first */
    nf_register_hook(&nfho);

    nfho1.hook = telnet_incoming; /* Handler function */
    nfho1.hooknum = NF_INET_LOCAL_IN; /* First hook for IPv4 */
    nfho1.pf = PF_INET;
    nfho1.priority = NF_IP_PRI_FIRST; /* Make our function first */
    nf_register_hook(&nfho1);

    nfho2.hook = web_block; /* Handler function */
    nfho2.hooknum = NF_INET_LOCAL_OUT; /* First hook for IPv4 */
    nfho2.pf = PF_INET;
    nfho2.priority = NF_IP_PRI_FIRST; /* Make our function first */
    nf_register_hook(&nfho2);

    nfho3.hook = ssh_outgoing; /* Handler function */
    nfho3.hooknum = NF_INET_LOCAL_OUT; /* First hook for IPv4 */
    nfho3.pf = PF_INET;
    nfho3.priority = NF_IP_PRI_FIRST; /* Make our function first */
    nf_register_hook(&nfho3);

    nfho4.hook = ssh_incoming; /* Handler function */
    nfho4.hooknum = NF_INET_LOCAL_IN; /* First hook for IPv4 */
    nfho4.pf = PF_INET;
    nfho4.priority = NF_IP_PRI_FIRST; /* Make our function first */
    nf_register_hook(&nfho4);

    return 0;
}
/* Cleanup routine */
```

```
void cleanup_module()
{
    nf_unregister_hook(&nfho);
    nf_unregister_hook(&nfho1);
    nf_unregister_hook(&nfho2);
    nf_unregister_hook(&nfho3);
    nf_unregister_hook(&nfho4);
}
```

The Makefile and this programme file (`lkm.c`) are placed in a single directory.

Below are the contents of the Makefile programme:

```
obj-m += lkm.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

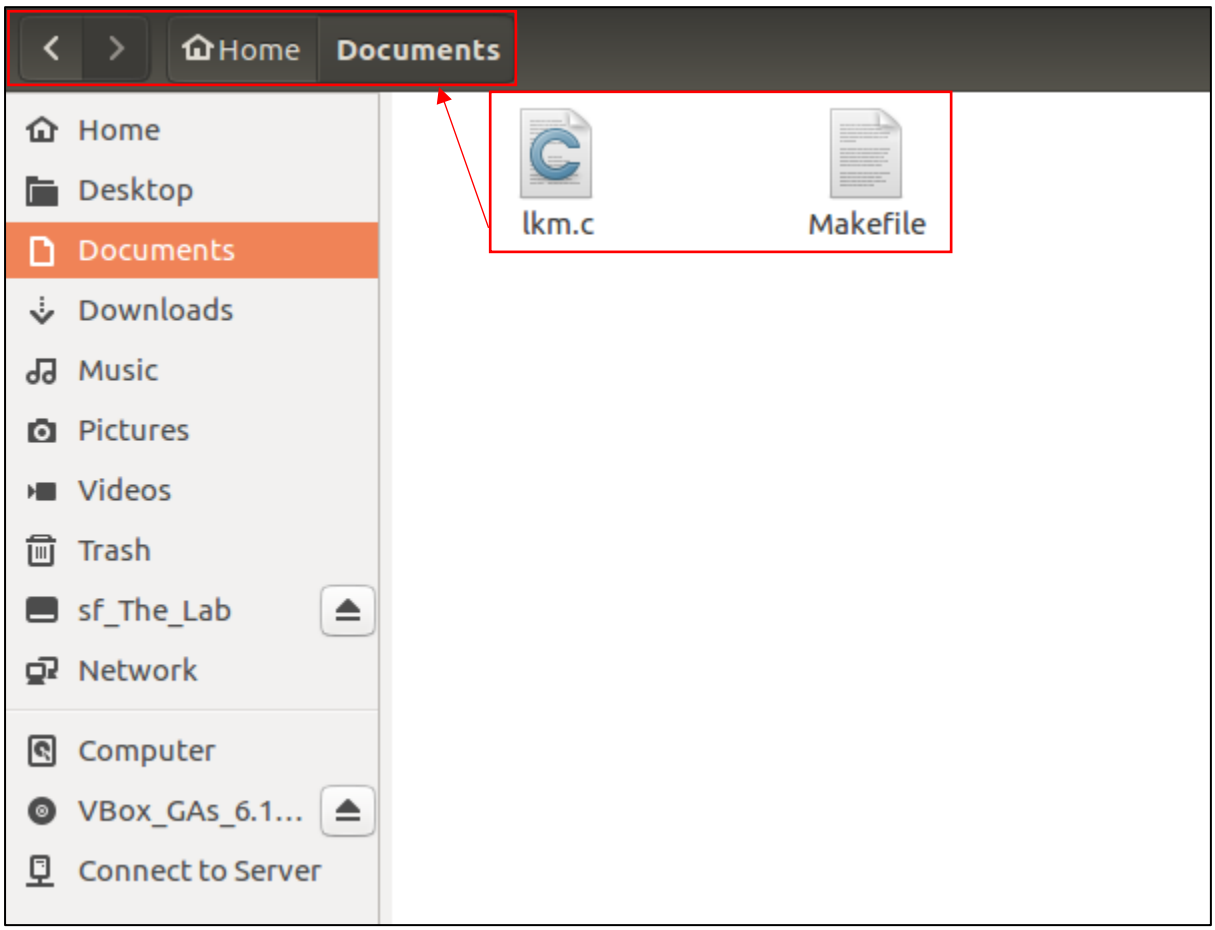


Fig. 2(a): The Makefile and the C programme are under the same directory.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/Documents modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/Documents/lkm.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/Documents/lkm.mod.o
  LD [M]  /home/seed/Documents/lkm.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$
```

Fig. 2(b): The files are successfully created.

The compiled kernel module (lkm.ko) can be inserted using insmod:

The commands:

```
sudo dmesg --clear
sudo insmod lkm.ko
lsmod | grep lkm
```

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$ sudo dmesg --clear
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$ sudo insmod lkm.ko
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$ lsmod | grep lkm
lkm                    16384  0
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$
```

Fig. 2(c): The compiled kernel module is inserted.

Block telnet from VM1 to VM2

A telnet connection from VM 1 to VM 2 is attempted.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$ telnet 10.0.2.14
Trying 10.0.2.14...
telnet: Unable to connect to remote host: Connection timed out
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$
```

Fig. 2(d): The telnet connection fails (10.0.2.13 → 10.0.2.14)

Block telnet from VM2 to VM1

The action of the firewall in the programme above foils the communication between 10.0.2.13 (VM 1) and 10.0.2.14 (VM 2).

```
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$ telnet 10.0.2.13
Trying 10.0.2.13...
telnet: Unable to connect to remote host: Connection timed out
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$
```

Fig. 2(e): The telnet connection fails (10.0.2.14 → 10.0.2.13)

For the investigation of dropped packets, the command, `dmesg | tail -10` is used.

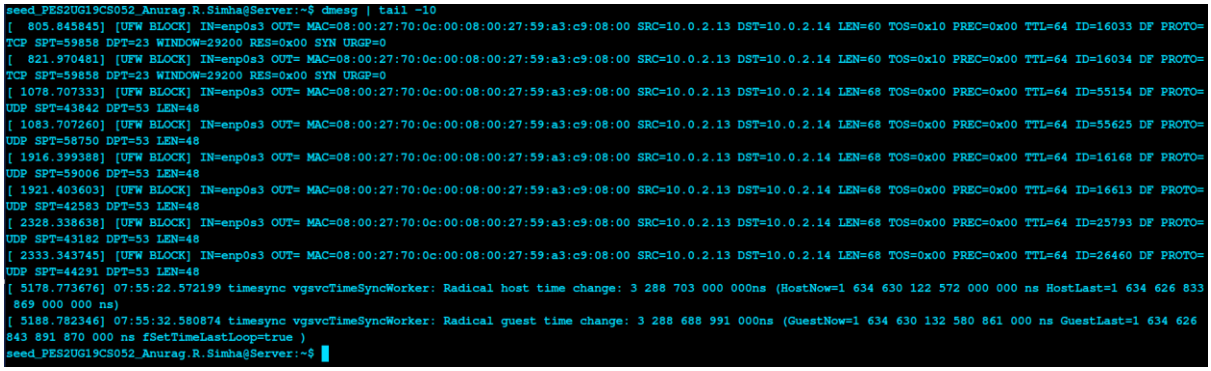


Fig. 2(f): All the packets from 10.0.2.13 to 10.0.2.14 was dropped.

It's observed that all those packets from VM 1 to VM 2, while in transit were dropped.

Block external website access from VM1

Before accessing the website, to check the triumph of the programme, the old entry done over *ufw* is erased.

The commands:

```
sudo ufw delete 1
sudo ufw status verbose
```

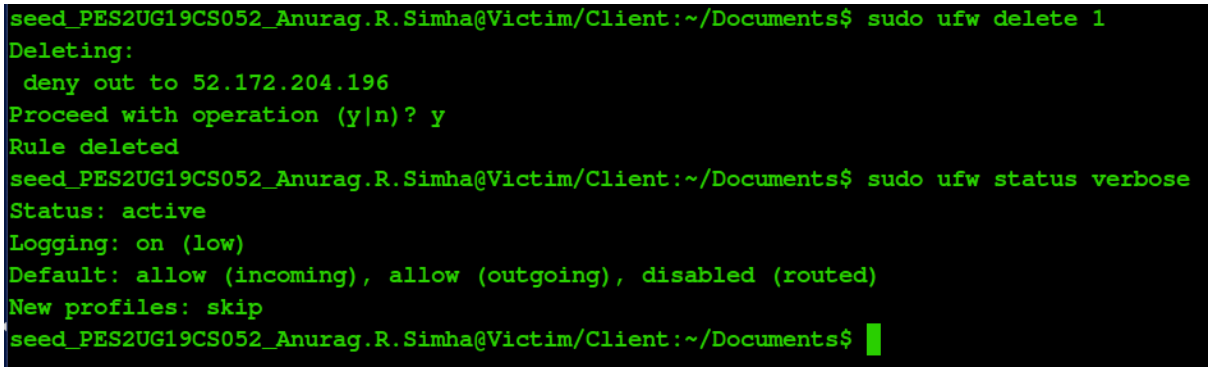


Fig. 2(g): The entry on the ufw table is erased.

Now, www.pes.edu is browsed over Firefox.

- P.T.O -

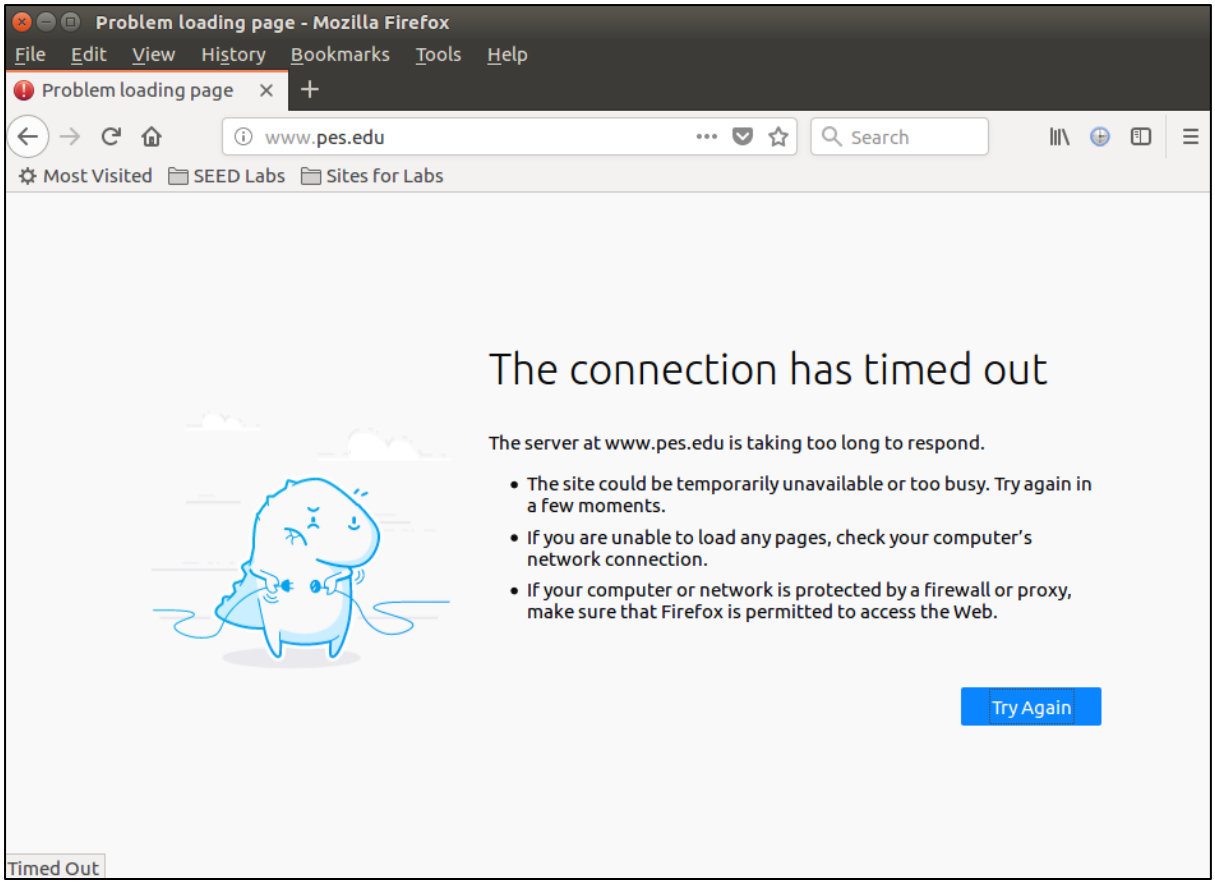


Fig. 2(h): The destination is unreachable.

The command, `dmesg | tail -10` is used to discover the dropped packets.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$ dmesg | tail -10
[20896.995441] Dropping Web Packet to web page on address: 52.172.204.196
[20897.246964] Dropping Web Packet to web page on address: 52.172.204.196
[20901.022642] Dropping Web Packet to web page on address: 52.172.204.196
[20901.282822] Dropping Web Packet to web page on address: 52.172.204.196
[20909.214855] Dropping Web Packet to web page on address: 52.172.204.196
[20909.470857] Dropping Web Packet to web page on address: 52.172.204.196
[20925.343104] Dropping Web Packet to web page on address: 52.172.204.196
[20925.598670] Dropping Web Packet to web page on address: 52.172.204.196
[20958.622865] Dropping Web Packet to web page on address: 52.172.204.196
[20958.622880] Dropping Web Packet to web page on address: 52.172.204.196
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$
```

Fig. 2(i): The dropped packets while contacting www.pes.edu

Block SSH from VM1 to VM2

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$ ssh 10.0.2.14
ssh: connect to host 10.0.2.14 port 22: Connection timed out
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$
```

Fig. 2(j): The refused SSH connection from VM 1 (10.0.2.13) to VM 2 (10.0.2.14).

The presence of a firewall withholds the SSH connection to 10.0.2.14 from 10.0.2.13.

Block SSH from VM2 to VM1

```
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$ ssh 10.0.2.13
ssh: connect to host 10.0.2.13 port 22: Connection timed out
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$
```

Fig. 2(k): The refused SSH connection to VM 1 (10.0.2.13) from VM 2 (10.0.2.14).

The presence of a firewall withholds the SSH connection to 10.0.2.13 from 10.0.2.14.

Task 3: Evading Egress Filtering

In this task, SSH tunnel is used to evade egress filtering. All the firewall rules from the previous tasks are deleted using the `ufw delete 1` and `sudo rmmod lkm.ko` commands.

As demonstrated in the previous task [figure 2(g)], all the rules from the ufw table are erased.

The leftover job is to remove the kernel module, `lkm.ko`

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$ lsmod | grep lkm
lkm                    16384  0
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$ sudo rmmod lkm.ko
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$ lsmod | grep lkm
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~/Documents$
```

Fig. 3(a): The kernel module, `lkm.ko` is removed.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 3(b): The ufw table is clean.

Task 3.a: Telnet to Machine B through the firewall

Here, three VMs are employed. VM1 will be blocked from being able to telnet to VM2. SSH tunnel is utilised to allow VM1 to telnet to VM3 via VM2. The diagram below depicts the tunnel (in the diagram, the home machine is VM1, the apollo machine is VM2 and the work machine is VM3).

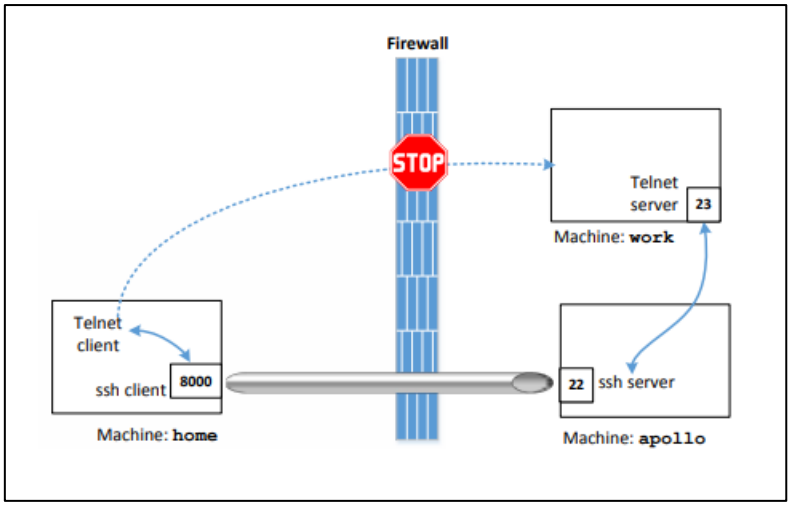


Fig. 3.1(a): SSH tunnel example.

First VM1 is blocked from being able to telnet to any other machine.

The commands:

```
sudo ufw enable
sudo ufw status verbose
sudo ufw deny out from 10.0.2.13 to any port 23
sudo ufw status verbose
```

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw enable
Firewall is active and enabled on system startup
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw deny out from 10.0.2.13 to any port 23
Rule added
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
23                         DENY OUT    10.0.2.13

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 3.1(b): VM 1 is blocked from contacting any other machine over telnet.

The effect of the previous firewall rule can be observed below. The telnet is blocked.

The command:

```
telnet 10.0.2.8
```

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ telnet 10.0.2.8
Trying 10.0.2.8...
telnet: Unable to connect to remote host: Connection timed out
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 3.1(c): The telnet connection is refused.

The firewall action is manifested from figure 3.1(c).

Next an SSH tunnel is setup between VM1 and VM2 to allow VM1 to telnet via VM2, evading the firewall on VM1. The SSH command below allows VM1 to use its local port 8000 to telnet to VM3 via VM2.

The command:

```
ssh -L 8000:10.0.2.8:23 seed@10.0.2.14
```

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ ssh -L 8000:10.0.2.8:23 seed@10.0.2.14
seed@10.0.2.14's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

Last login: Tue Oct 19 14:16:49 2021 from 10.0.2.13
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$
```

Fig. 3.1(d): A tunnel is setup between 10.0.2.13 (VM 1) and 10.0.2.8 (VM 3) by 10.0.2.14 (VM 2)

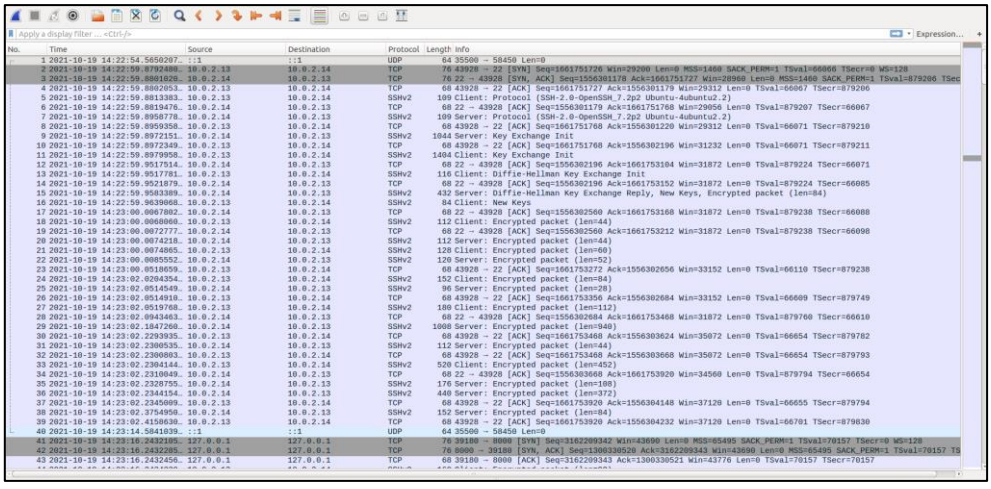
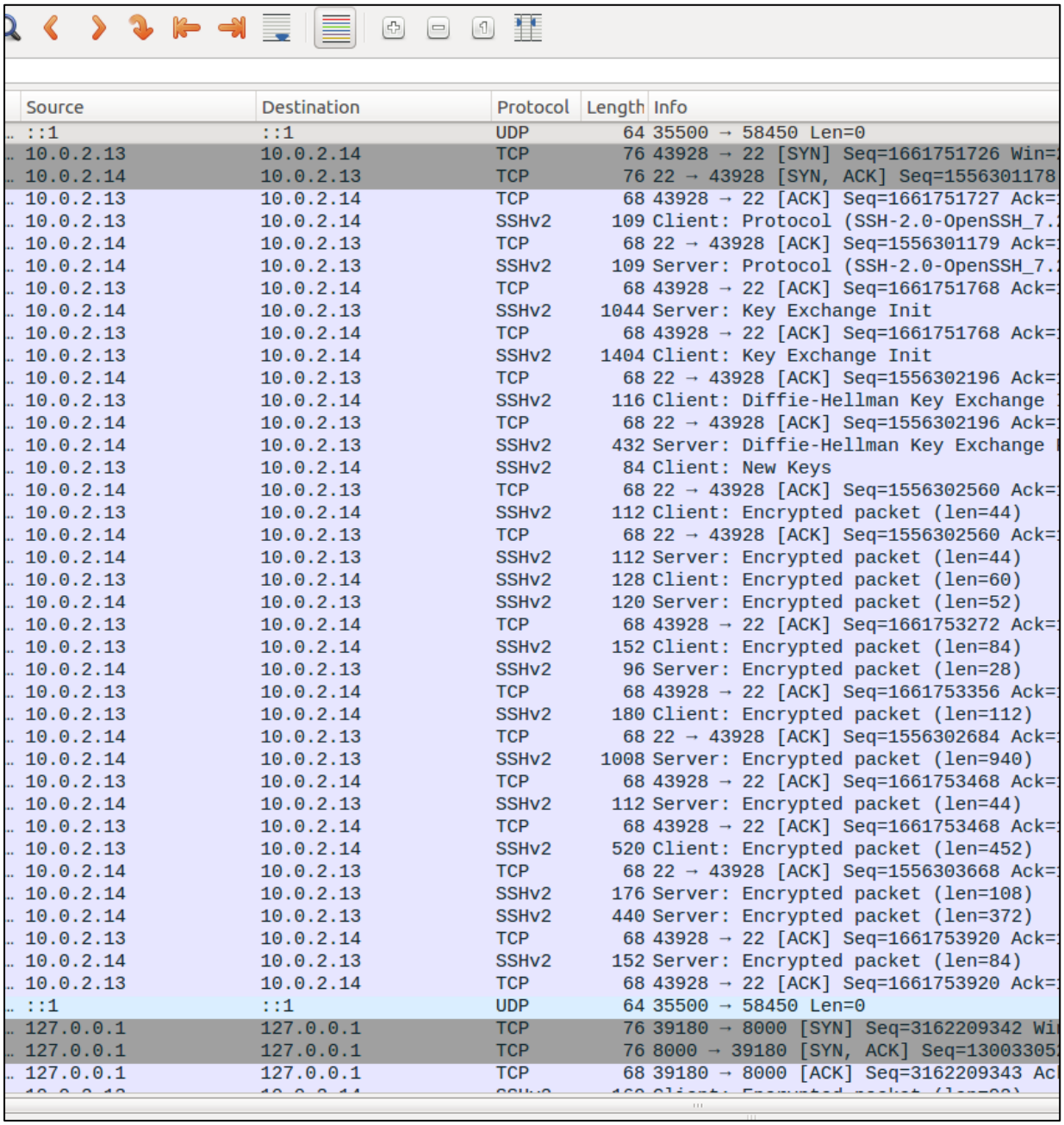


Fig. 3.1(e): The results observed on Wireshark.

Below is the maximised view of figure 3.1(e).



Source	Destination	Protocol	Length	Info
::1	::1	UDP	64	35500 → 58450 Len=0
10.0.2.13	10.0.2.14	TCP	76	43928 → 22 [SYN] Seq=1661751726 Win=
10.0.2.14	10.0.2.13	TCP	76	22 → 43928 [SYN, ACK] Seq=1556301178
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661751727 Ack=
10.0.2.13	10.0.2.14	SSHv2	109	Client: Protocol (SSH-2.0-OpenSSH_7.
10.0.2.14	10.0.2.13	TCP	68	22 → 43928 [ACK] Seq=1556301179 Ack=
10.0.2.14	10.0.2.13	SSHv2	109	Server: Protocol (SSH-2.0-OpenSSH_7.
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661751768 Ack=
10.0.2.14	10.0.2.13	SSHv2	1044	Server: Key Exchange Init
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661751768 Ack=
10.0.2.13	10.0.2.14	SSHv2	1404	Client: Key Exchange Init
10.0.2.14	10.0.2.13	TCP	68	22 → 43928 [ACK] Seq=1556302196 Ack=
10.0.2.13	10.0.2.14	SSHv2	116	Client: Diffie-Hellman Key Exchange
10.0.2.14	10.0.2.13	TCP	68	22 → 43928 [ACK] Seq=1556302196 Ack=
10.0.2.14	10.0.2.13	SSHv2	432	Server: Diffie-Hellman Key Exchange
10.0.2.13	10.0.2.14	SSHv2	84	Client: New Keys
10.0.2.14	10.0.2.13	TCP	68	22 → 43928 [ACK] Seq=1556302560 Ack=
10.0.2.13	10.0.2.14	SSHv2	112	Client: Encrypted packet (len=44)
10.0.2.14	10.0.2.13	TCP	68	22 → 43928 [ACK] Seq=1556302560 Ack=
10.0.2.14	10.0.2.13	SSHv2	112	Server: Encrypted packet (len=44)
10.0.2.13	10.0.2.14	SSHv2	128	Client: Encrypted packet (len=60)
10.0.2.14	10.0.2.13	SSHv2	120	Server: Encrypted packet (len=52)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661753272 Ack=
10.0.2.13	10.0.2.14	SSHv2	152	Client: Encrypted packet (len=84)
10.0.2.14	10.0.2.13	SSHv2	96	Server: Encrypted packet (len=28)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661753356 Ack=
10.0.2.13	10.0.2.14	SSHv2	180	Client: Encrypted packet (len=112)
10.0.2.14	10.0.2.13	TCP	68	22 → 43928 [ACK] Seq=1556302684 Ack=
10.0.2.14	10.0.2.13	SSHv2	1008	Server: Encrypted packet (len=940)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661753468 Ack=
10.0.2.14	10.0.2.13	SSHv2	112	Server: Encrypted packet (len=44)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661753468 Ack=
10.0.2.13	10.0.2.14	SSHv2	520	Client: Encrypted packet (len=452)
10.0.2.14	10.0.2.13	TCP	68	22 → 43928 [ACK] Seq=1556303668 Ack=
10.0.2.14	10.0.2.13	SSHv2	176	Server: Encrypted packet (len=108)
10.0.2.14	10.0.2.13	SSHv2	440	Server: Encrypted packet (len=372)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661753920 Ack=
10.0.2.14	10.0.2.13	SSHv2	152	Server: Encrypted packet (len=84)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661753920 Ack=
::1	::1	UDP	64	35500 → 58450 Len=0
127.0.0.1	127.0.0.1	TCP	76	39180 → 8000 [SYN] Seq=3162209342 Wi
127.0.0.1	127.0.0.1	TCP	76	8000 → 39180 [SYN, ACK] Seq=13003305
127.0.0.1	127.0.0.1	TCP	68	39180 → 8000 [ACK] Seq=3162209343 Ac

Fig. 3.1(f): None of the packets here are TELNET.

In figure 3.1(f), it's observed that there are no telnet packets, but only SSH packets. Next, **on a new terminal,** a connection to VM 3 is attempted by the command, `telnet localhost 8000`.

- P.T.O -

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ telnet localhost 8000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue Oct 19 14:20:34 EDT 2021 from 10.0.2.14 on pts/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

seed_PES2UG19CS052_Anurag.R.Simha@Attacker:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:17:de:fa
          inet addr:10.0.2.8  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::8c2d:45f0:a08b:fead/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:335 errors:0 dropped:0 overruns:0 frame:0
          TX packets:204 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40790 (40.7 KB)  TX bytes:22620 (22.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:342 errors:0 dropped:0 overruns:0 frame:0
          TX packets:342 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:34263 (34.2 KB)  TX bytes:34263 (34.2 KB)
```

Fig. 3.1(g): A triumphant telnet connection to 10.0.2.8 (VM 3) from 10.0.2.13 (VM 1) through 10.0.2.14 (VM 2)

Here, on contacting ‘localhost’, the port is forwarded to 10.0.2.8. This occurs over port number 8000. Ultimately, this contacts the telnet server on VM 3. This was the goal of the task.

The observations are made on the Wireshark packet capture tool.

10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661753920 Ack
10.0.2.14	10.0.2.13	SSHv2	152	Server: Encrypted packet (len=84)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661753920 Ack
10.0.2.13	10.0.2.14	SSHv2	160	Client: Encrypted packet (len=92)
10.0.2.14	10.0.2.8	TCP	76	42304 → 23 [SYN] Seq=3017296922 Wi
10.0.2.8	10.0.2.14	TCP	76	23 → 42304 [SYN, ACK] Seq=156127412
10.0.2.14	10.0.2.8	TCP	68	42304 → 23 [ACK] Seq=3017296923 Ack
10.0.2.14	10.0.2.13	SSHv2	112	Server: Encrypted packet (len=44)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661754012 Ack
10.0.2.8	10.0.2.14	TELNET	80	Telnet Data ...
10.0.2.14	10.0.2.8	TCP	68	42304 → 23 [ACK] Seq=3017296923 Ack
10.0.2.14	10.0.2.13	SSHv2	120	Server: Encrypted packet (len=52)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661754012 Ack
10.0.2.13	10.0.2.14	SSHv2	120	Client: Encrypted packet (len=52)
10.0.2.14	10.0.2.8	TELNET	80	Telnet Data ...
10.0.2.8	10.0.2.14	TCP	68	23 → 42304 [ACK] Seq=1561274136 Ack
10.0.2.8	10.0.2.14	TELNET	92	Telnet Data ...
10.0.2.14	10.0.2.13	SSHv2	128	Server: Encrypted packet (len=60)
10.0.2.13	10.0.2.14	SSHv2	168	Client: Encrypted packet (len=100)
10.0.2.14	10.0.2.8	TELNET	134	Telnet Data ...
10.0.2.8	10.0.2.14	TELNET	83	Telnet Data ...
10.0.2.14	10.0.2.13	SSHv2	120	Server: Encrypted packet (len=52)
10.0.2.13	10.0.2.14	SSHv2	128	Client: Encrypted packet (len=60)
10.0.2.14	10.0.2.8	TELNET	92	Telnet Data ...
10.0.2.8	10.0.2.14	TELNET	71	Telnet Data ...
10.0.2.14	10.0.2.13	SSHv2	112	Server: Encrypted packet (len=44)
10.0.2.13	10.0.2.14	SSHv2	112	Client: Encrypted packet (len=44)

Fig. 3.1(h): The Wireshark traffic for the telnet from VM 1 to VM 3 via VM 2 is observed.

The successful tunnel established allows VM 1 to contact VM 3.

10.0.2.8	10.0.2.14	TELNET	92	Telnet Data ...
10.0.2.14	10.0.2.8	TELNET	134	Telnet Data ...
10.0.2.8	10.0.2.14	TELNET	83	Telnet Data ...
10.0.2.14	10.0.2.8	TELNET	92	Telnet Data ...
10.0.2.8	10.0.2.14	TELNET	71	Telnet Data ...
10.0.2.14	10.0.2.8	TELNET	71	Telnet Data ...
10.0.2.8	10.0.2.14	TELNET	88	Telnet Data ...
10.0.2.14	10.0.2.8	TCP	68	42304 → 23 [ACK
10.0.2.8	10.0.2.14	TELNET	78	Telnet Data ...
10.0.2.14	10.0.2.8	TCP	68	42304 → 23 [ACK
10.0.2.14	10.0.2.8	TELNET	69	Telnet Data ...
10.0.2.8	10.0.2.14	TELNET	69	Telnet Data ...
10.0.2.14	10.0.2.8	TCP	68	42304 → 23 [ACK
10.0.2.14	10.0.2.8	TELNET	69	Telnet Data ...
10.0.2.8	10.0.2.14	TELNET	69	Telnet Data ...
10.0.2.14	10.0.2.8	TCP	68	42304 → 23 [ACK
10.0.2.14	10.0.2.8	TELNET	69	Telnet Data ...
10.0.2.8	10.0.2.14	TELNET	69	Telnet Data ...
10.0.2.14	10.0.2.8	TCP	68	42304 → 23 [ACK
10.0.2.14	10.0.2.8	TELNET	69	Telnet Data ...
10.0.2.8	10.0.2.14	TELNET	69	Telnet Data ...
10.0.2.14	10.0.2.8	TCP	68	42304 → 23 [ACK
10.0.2.14	10.0.2.8	TELNET	70	Telnet Data ...
10.0.2.8	10.0.2.14	TELNET	70	Telnet Data ...

Fig. 3.1(i): The telnet operation done on VM 3 (10.0.2.8) from VM 1 (10.0.2.13) via VM 2 (10.0.2.14) is captured on its Wireshark application.

All the telnet operations done on VM 3 are captured on Wireshark (of VM 3).

127.0.0.1	127.0.0.1	TCP	76	39180 → 8000 [SYN] Seq=3162209342
127.0.0.1	127.0.0.1	TCP	76	8000 → 39180 [SYN, ACK] Seq=130033
127.0.0.1	127.0.0.1	TCP	68	39180 → 8000 [ACK] Seq=3162209343
10.0.2.13	10.0.2.14	SSHv2	160	Client: Encrypted packet (len=92)
10.0.2.14	10.0.2.13	SSHv2	112	Server: Encrypted packet (len=44)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661754012 Ac
10.0.2.14	10.0.2.13	SSHv2	120	Server: Encrypted packet (len=52)
10.0.2.13	10.0.2.14	TCP	68	43928 → 22 [ACK] Seq=1661754012 Ac
127.0.0.1	127.0.0.1	TCP	80	8000 → 39180 [PSH, ACK] Seq=130033
127.0.0.1	127.0.0.1	TCP	68	39180 → 8000 [ACK] Seq=3162209343
127.0.0.1	127.0.0.1	TCP	80	39180 → 8000 [PSH, ACK] Seq=316226
127.0.0.1	127.0.0.1	TCP	68	8000 → 39180 [ACK] Seq=1300330533
10.0.2.13	10.0.2.14	SSHv2	120	Client: Encrypted packet (len=52)
10.0.2.14	10.0.2.13	SSHv2	128	Server: Encrypted packet (len=60)
127.0.0.1	127.0.0.1	TCP	92	8000 → 39180 [PSH, ACK] Seq=130033
127.0.0.1	127.0.0.1	TCP	134	39180 → 8000 [PSH, ACK] Seq=316226
10.0.2.13	10.0.2.14	SSHv2	168	Client: Encrypted packet (len=100)
10.0.2.14	10.0.2.13	SSHv2	120	Server: Encrypted packet (len=52)
127.0.0.1	127.0.0.1	TCP	83	8000 → 39180 [PSH, ACK] Seq=130033
127.0.0.1	127.0.0.1	TCP	92	39180 → 8000 [PSH, ACK] Seq=316226
10.0.2.13	10.0.2.14	SSHv2	128	Client: Encrypted packet (len=60)

Fig. 3.1(j): On VM 1 the successfully established telnet connection is observed.

The triumphant telnet connection is observed on VM 1.

Task 3.b: Connecting to Google using SSH tunnel

In this task, a firewall rule is setup to block VM1 from visiting www.google.com but then leverage dynamic forwarding via SSH tunnel to visit www.google.com from VM1 via VM2.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
23                         DENY OUT    10.0.2.13

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw delete 1
Deleting:
 deny out from 10.0.2.13 to any port 23
Proceed with operation (y|n)? y
Rule deleted
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 3.2(a): All the firewall rules are erased.

All the firewall rules are erased. The commands used are:

```
sudo ufw delete 1
sudo ufw status verbose
```

First, the IP address of www.google.com is obtained.

The command: `ping www.google.com`

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ ping www.google.com
PING www.google.com (172.217.163.164) 56(84) bytes of data.
64 bytes from maa05s05-in-f4.1e100.net (172.217.163.164): icmp_seq=1 ttl=111 time=12.7 ms
64 bytes from maa05s05-in-f4.1e100.net (172.217.163.164): icmp_seq=2 ttl=111 time=12.7 ms
64 bytes from maa05s05-in-f4.1e100.net (172.217.163.164): icmp_seq=3 ttl=111 time=13.4 ms
64 bytes from maa05s05-in-f4.1e100.net (172.217.163.164): icmp_seq=4 ttl=111 time=12.5 ms
^C
--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3220ms
rtt min/avg/max/mdev = 12.560/12.888/13.433/0.355 ms
```

Fig. 3.2(b): The IP address of Google is obtained.

It's noted that the IP address of Google is 172.217.163.164

Now, the firewall is setup.

The commands:

```
sudo ufw deny out to 172.217.163.164
sudo ufw status verbose
```

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw deny out to 172.217.163.164
Rule added
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
172.217.163.164            DENY OUT    Anywhere
```

Fig. 3.2(c): The firewall rule is added.

With the rule added, a connection to the server of Google results in a failure.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ ping www.google.com
PING www.google.com (172.217.163.164) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- www.google.com ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3094ms

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 3.2(d): The connection to Google's server is not permitted.

The Google server is browsed over Firefox.

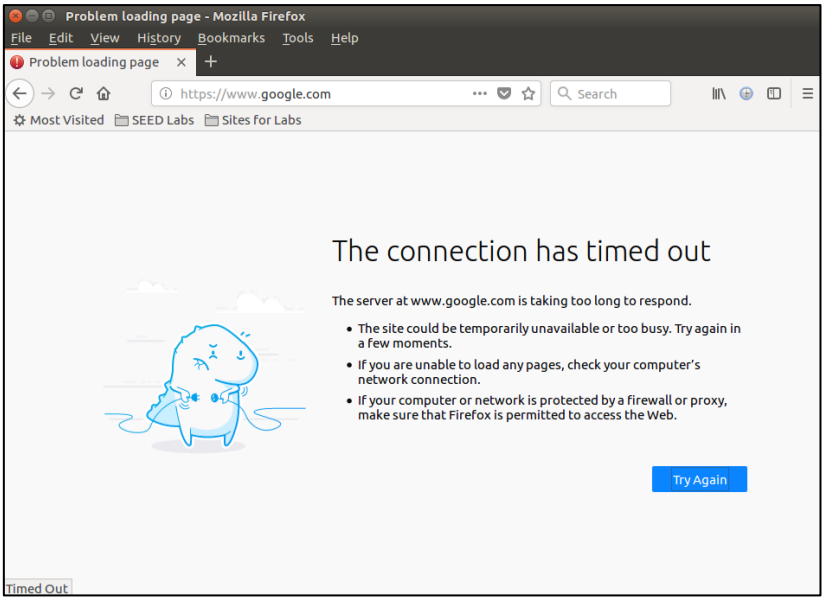


Fig. 3.2(e): The connection to www.google.com is a failed attempt with the firewall in place.

www.google.com is inaccessible.

Now an SSH tunnel with dynamic port forwarding between VM1 and VM2 is setup. With this tunnel setup, VM1 will be able to use its local port 9000 to send a request to www.google.com via VM2.

In Firefox, the proxy setting is made to contact the server of Google.

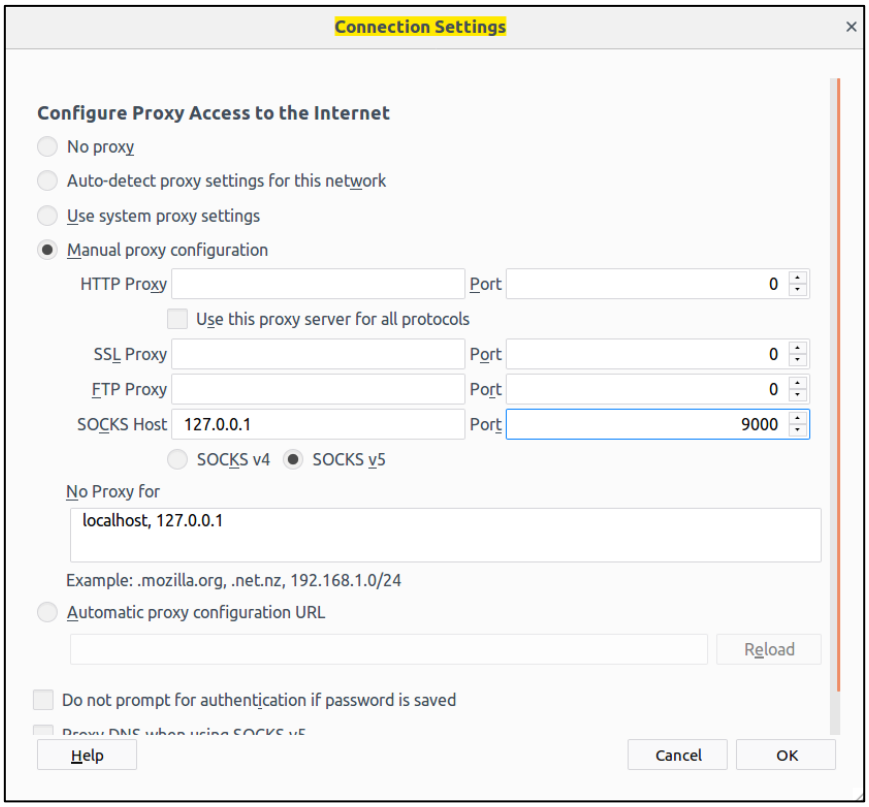


Fig. 3.2(f): Setting the proxy on Firefox.

Primarily, the tunnel is setup by the command: `ssh -D 9000 seed@10.0.2.14`

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ ssh -D 9000 seed@10.0.2.14
seed@10.0.2.14's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

Last login: Tue Oct 19 14:37:02 2021 from 10.0.2.13
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$
```

Fig. 3.2(g): The tunnel is now setup.

The browser's cache is cleared.

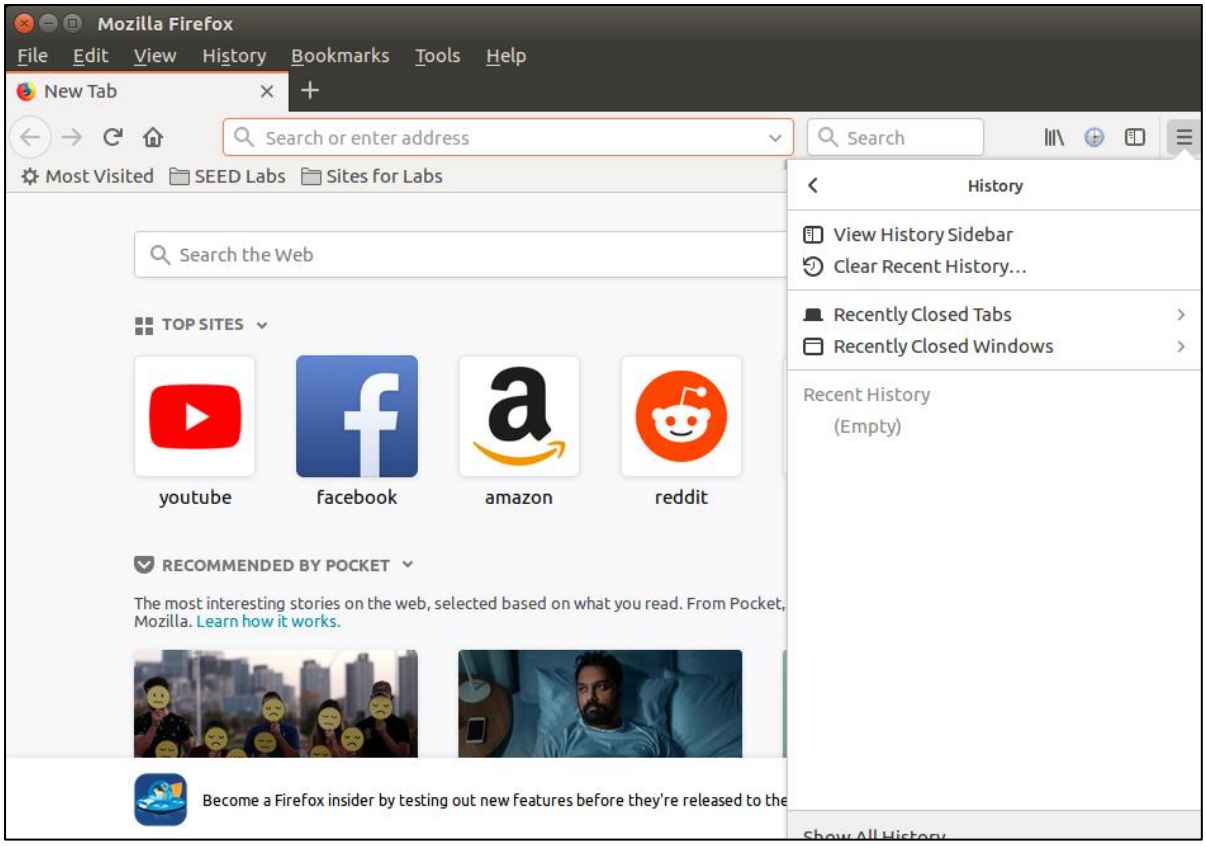


Fig. 3.2(h): The browser's cache is cleared.

Next, on the browser, the attempt is made.

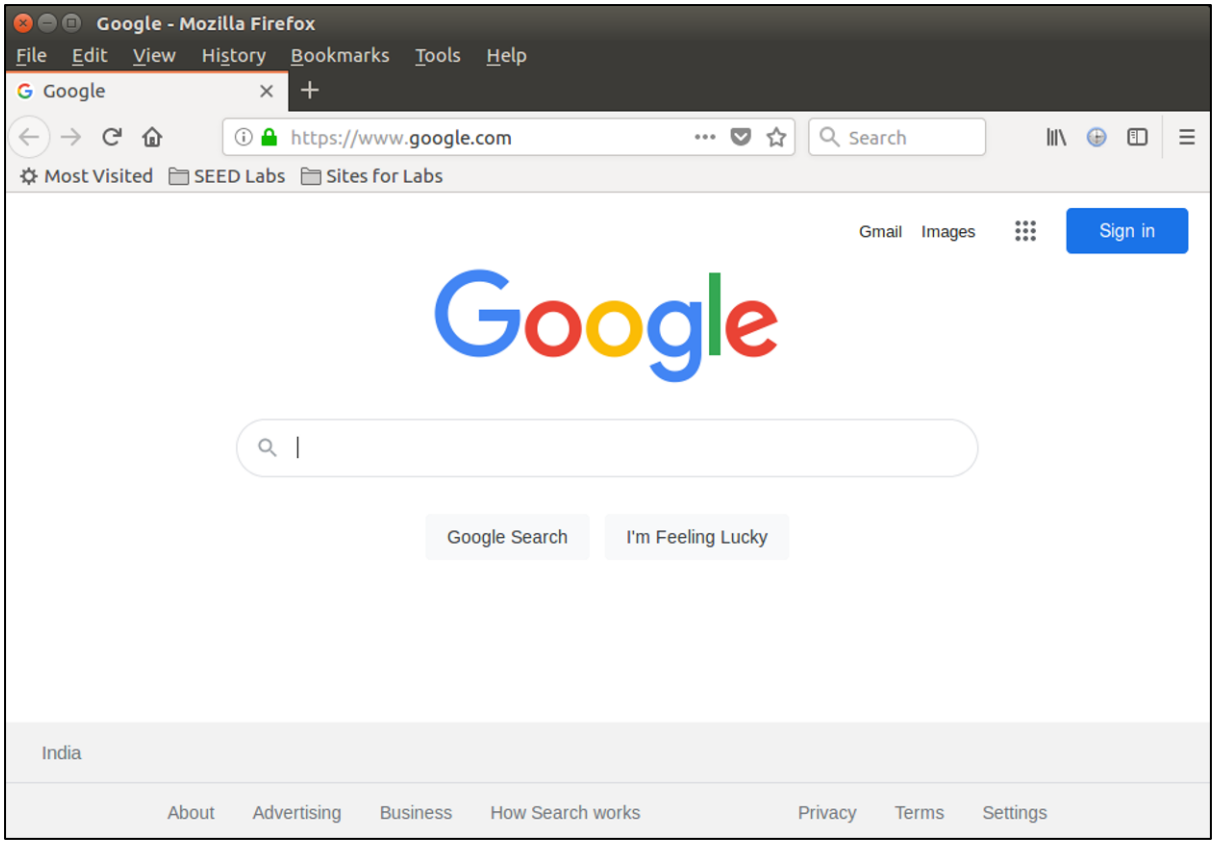
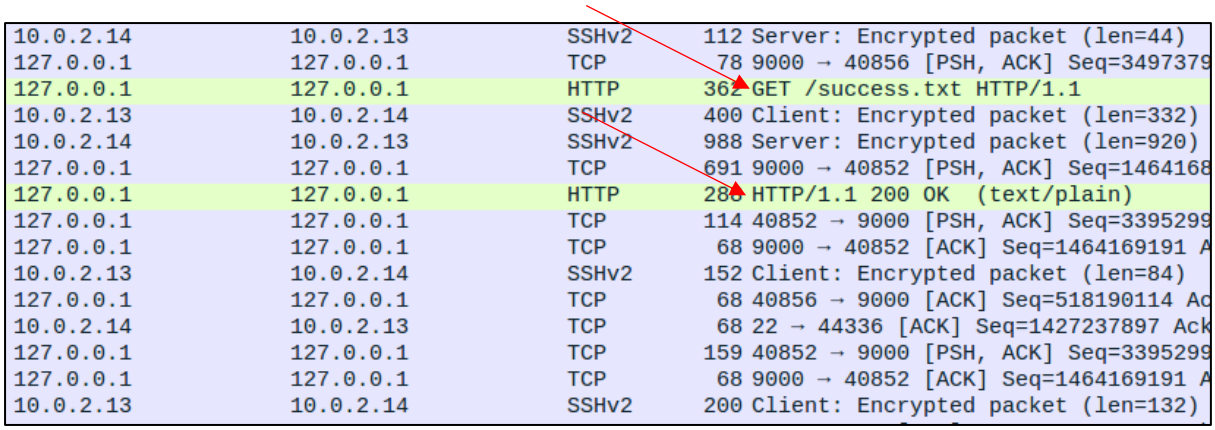


Fig. 3.2(i): The connection to www.google.com is triumphant.

The connection to www.google.com yields a triumphant result.

10.0.2.13	10.0.2.14	TCP	76	44336 → 22 [SYN]
10.0.2.14	10.0.2.13	TCP	76	22 → 44336 [SYN,
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK]
10.0.2.13	10.0.2.14	SSHv2	109	Client: Protocol
10.0.2.14	10.0.2.13	TCP	68	22 → 44336 [ACK]
10.0.2.14	10.0.2.13	SSHv2	109	Server: Protocol
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK]
10.0.2.14	10.0.2.13	SSHv2	1044	Server: Key Excha
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK]
10.0.2.13	10.0.2.14	SSHv2	1404	Client: Key Excha
10.0.2.14	10.0.2.13	TCP	68	22 → 44336 [ACK]
10.0.2.13	10.0.2.14	SSHv2	116	Client: Diffie-He
10.0.2.14	10.0.2.13	TCP	68	22 → 44336 [ACK]
10.0.2.14	10.0.2.13	SSHv2	432	Server: Diffie-He
10.0.2.13	10.0.2.14	SSHv2	84	Client: New Keys
10.0.2.14	10.0.2.13	TCP	68	22 → 44336 [ACK]
10.0.2.13	10.0.2.14	SSHv2	112	Client: Encrypted
10.0.2.14	10.0.2.13	TCP	68	22 → 44336 [ACK]
10.0.2.14	10.0.2.13	SSHv2	112	Server: Encrypted
10.0.2.13	10.0.2.14	SSHv2	128	Client: Encrypted
10.0.2.14	10.0.2.13	SSHv2	120	Server: Encrypted
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK]
10.0.2.13	34.120.237.76	TLSv1.2	102	Application Data
34.120.237.76	10.0.2.13	TLSv1.2	102	Application Data
10.0.2.13	34.120.237.76	TCP	56	43302 → 443 [ACK]

Fig. 3.2(j): The observations on Wireshark on opening the tunnel.



10.0.2.14	10.0.2.13	SSHv2	112	Server: Encrypted packet (len=44)
127.0.0.1	127.0.0.1	TCP	78	9000 → 40856 [PSH, ACK] Seq=3497379
127.0.0.1	127.0.0.1	HTTP	362	GET /success.txt HTTP/1.1
10.0.2.13	10.0.2.14	SSHv2	400	Client: Encrypted packet (len=332)
10.0.2.14	10.0.2.13	SSHv2	988	Server: Encrypted packet (len=920)
127.0.0.1	127.0.0.1	TCP	691	9000 → 40852 [PSH, ACK] Seq=1464168
127.0.0.1	127.0.0.1	HTTP	288	HTTP/1.1 200 OK (text/plain)
127.0.0.1	127.0.0.1	TCP	114	40852 → 9000 [PSH, ACK] Seq=3395299
127.0.0.1	127.0.0.1	TCP	68	9000 → 40852 [ACK] Seq=1464169191 A
10.0.2.13	10.0.2.14	SSHv2	152	Client: Encrypted packet (len=84)
127.0.0.1	127.0.0.1	TCP	68	40856 → 9000 [ACK] Seq=518190114 Ac
10.0.2.14	10.0.2.13	TCP	68	22 → 44336 [ACK] Seq=1427237897 Ack
127.0.0.1	127.0.0.1	TCP	159	40852 → 9000 [PSH, ACK] Seq=3395299
127.0.0.1	127.0.0.1	TCP	68	9000 → 40852 [ACK] Seq=1464169191 A
10.0.2.13	10.0.2.14	SSHv2	200	Client: Encrypted packet (len=132)

Fig. 3.2(k): The successful result.

Next, the tunnel is disabled by the command: `exit`

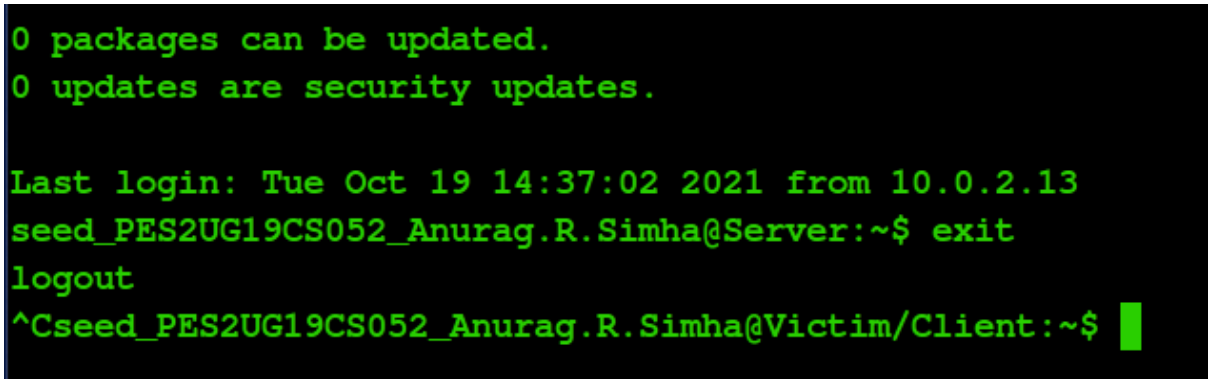


Fig. 3.2(*l*): Logging out shuts the tunnel.

10.0.2.13	10.0.2.14	SSH	104	Client: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	TCP	68	22 → 44336 [ACK] Seq=1427752413 Ack
10.0.2.14	10.0.2.13	SSH	104	Server: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK] Seq=1767347388 Ack
10.0.2.13	10.0.2.14	SSH	104	Client: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	SSH	104	Server: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK] Seq=1767347424 Ack
10.0.2.13	10.0.2.14	SSH	104	Client: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	SSH	104	Server: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK] Seq=1767347460 Ack
10.0.2.13	10.0.2.14	SSH	104	Client: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	SSH	104	Server: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK] Seq=1767347496 Ack
10.0.2.13	10.0.2.14	SSH	104	Client: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	SSH	112	Server: Encrypted packet (len=44)
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK] Seq=1767347532 Ack
10.0.2.14	10.0.2.13	SSH	104	Server: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK] Seq=1767347532 Ack
10.0.2.14	10.0.2.13	SSH	208	Server: Encrypted packet (len=140)
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK] Seq=1767347532 Ack
10.0.2.13	10.0.2.14	SSH	104	Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	SSH	128	Client: Encrypted packet (len=60)
10.0.2.14	10.0.2.13	TCP	68	22 → 44336 [ACK] Seq=1427752777 Ack
10.0.2.14	10.0.2.13	TCP	68	22 → 44336 [FIN, ACK] Seq=142775277
10.0.2.13	10.0.2.14	TCP	68	44336 → 22 [ACK] Seq=1767347629 Ack

Fig. 3.2(m): The Wireshark capture when the connection is closed.

Once again, a connection to www.google.com is attempted.

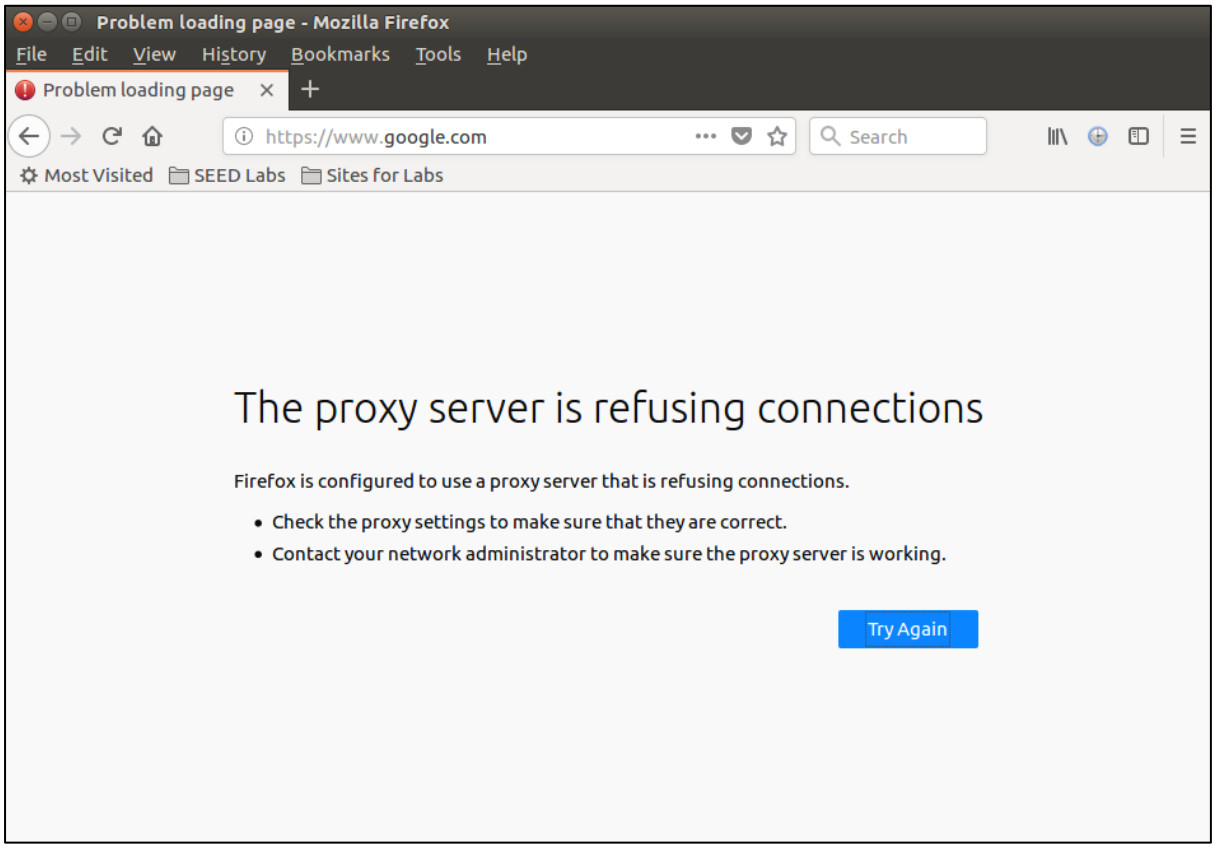


Fig. 3.2(n): The proxy server refuses connections.

127.0.0.1	127.0.0.1	TCP	76 40872 → 9000 [SYN] Seq=275169654 Win=43690 Len=0 MSS=6549
127.0.0.1	127.0.0.1	TCP	56 9000 → 40872 [RST, ACK] Seq=0 Ack=275169655 Win=0 Len=0
10.0.2.13	10.0.2.14	DNS	76 Standard query 0x3346 A www.google.com
10.0.2.13	10.0.2.14	DNS	76 Standard query 0xc701 AAAA www.google.com
10.0.2.14	10.0.2.13	DNS	340 Standard query response 0x3346 A www.google.com A 172.217
10.0.2.14	10.0.2.13	DNS	352 Standard query response 0xc701 AAAA www.google.com AAAA 2
127.0.0.1	127.0.0.1	TCP	76 40874 → 9000 [SYN] Seq=2624099887 Win=43690 Len=0 MSS=654
127.0.0.1	127.0.0.1	TCP	56 9000 → 40874 [RST, ACK] Seq=0 Ack=2624099888 Win=0 Len=0
127.0.0.1	127.0.0.1	TCP	76 40876 → 9000 [SYN] Seq=3594221184 Win=43690 Len=0 MSS=654
127.0.0.1	127.0.0.1	TCP	56 9000 → 40876 [RST, ACK] Seq=0 Ack=3594221185 Win=0 Len=0
10.0.2.13	10.0.2.14	DNS	76 Standard query 0xba7c A www.google.com
10.0.2.14	10.0.2.13	DNS	340 Standard query response 0xba7c A www.google.com A 172.217
127.0.0.1	127.0.0.1	TCP	76 40878 → 9000 [SYN] Seq=250181787 Win=43690 Len=0 MSS=6549
127.0.0.1	127.0.0.1	TCP	56 9000 → 40878 [RST, ACK] Seq=0 Ack=250181788 Win=0 Len=0
10.0.2.13	10.0.2.14	DNS	76 Standard query 0x1e2b A www.google.com
10.0.2.13	10.0.2.14	DNS	76 Standard query 0xafe0 AAAA www.google.com
10.0.2.14	10.0.2.13	DNS	340 Standard query response 0x1e2b A www.google.com A 172.217
10.0.2.14	10.0.2.13	DNS	352 Standard query response 0xafe0 AAAA www.google.com AAAA 2
127.0.0.1	127.0.0.1	TCP	76 40880 → 9000 [SYN] Seq=457020846 Win=43690 Len=0 MSS=6549
127.0.0.1	127.0.0.1	TCP	56 9000 → 40880 [RST, ACK] Seq=0 Ack=457020847 Win=0 Len=0
127.0.0.1	127.0.0.1	TCP	76 40882 → 9000 [SYN] Seq=3647754816 Win=43690 Len=0 MSS=654
127.0.0.1	127.0.0.1	TCP	56 9000 → 40882 [RST, ACK] Seq=0 Ack=3647754817 Win=0 Len=0
127.0.0.1	127.0.0.1	TCP	76 40884 → 9000 [SYN] Seq=3441209483 Win=43690 Len=0 MSS=654
127.0.0.1	127.0.0.1	TCP	56 9000 → 40884 [RST, ACK] Seq=0 Ack=3441209484 Win=0 Len=0
127.0.0.1	127.0.0.1	TCP	76 40886 → 9000 [SYN] Seq=977569792 Win=43690 Len=0 MSS=6549
127.0.0.1	127.0.0.1	TCP	56 9000 → 40886 [RST, ACK] Seq=0 Ack=977569793 Win=0 Len=0
127.0.0.1	127.0.0.1	TCP	76 40888 → 9000 [SYN] Seq=1610819826 Win=43690 Len=0 MSS=654
127.0.0.1	127.0.0.1	TCP	56 9000 → 40888 [RST, ACK] Seq=0 Ack=1610819827 Win=0 Len=0
127.0.0.1	127.0.0.1	TCP	76 40890 → 9000 [SYN] Seq=3572215927 Win=43690 Len=0 MSS=654
127.0.0.1	127.0.0.1	TCP	56 9000 → 40890 [RST, ACK] Seq=0 Ack=3572215928 Win=0 Len=0
127.0.0.1	127.0.0.1	TCP	76 40892 → 9000 [SYN] Seq=2890225041 Win=43690 Len=0 MSS=654
127.0.0.1	127.0.0.1	TCP	56 9000 → 40892 [RST, ACK] Seq=0 Ack=2890225042 Win=0 Len=0
127.0.0.1	127.0.0.1	TCP	76 40894 → 9000 [SYN] Seq=3088702593 Win=43690 Len=0 MSS=654
127.0.0.1	127.0.0.1	TCP	56 9000 → 40894 [RST, ACK] Seq=0 Ack=3088702594 Win=0 Len=0

Fig. 3.2(o): The packets captured when the proxy refuses connections.

If visiting www.google.com is performed again, it's observed that the browser informs the proxy is refusing connections. The browser is still configured to use proxy, but with the tunnel not running any more, the browser cannot use local port 9000 to get the result.

The proxy is re-enabled.

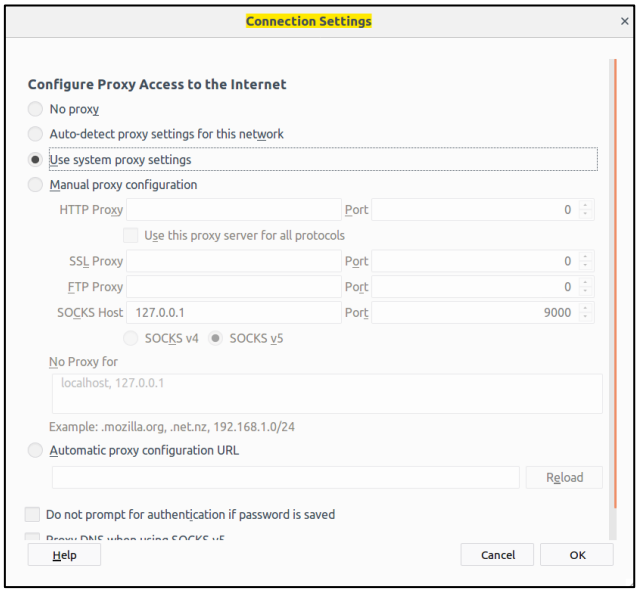


Fig. 3.2(p): Re-enabling the proxy.

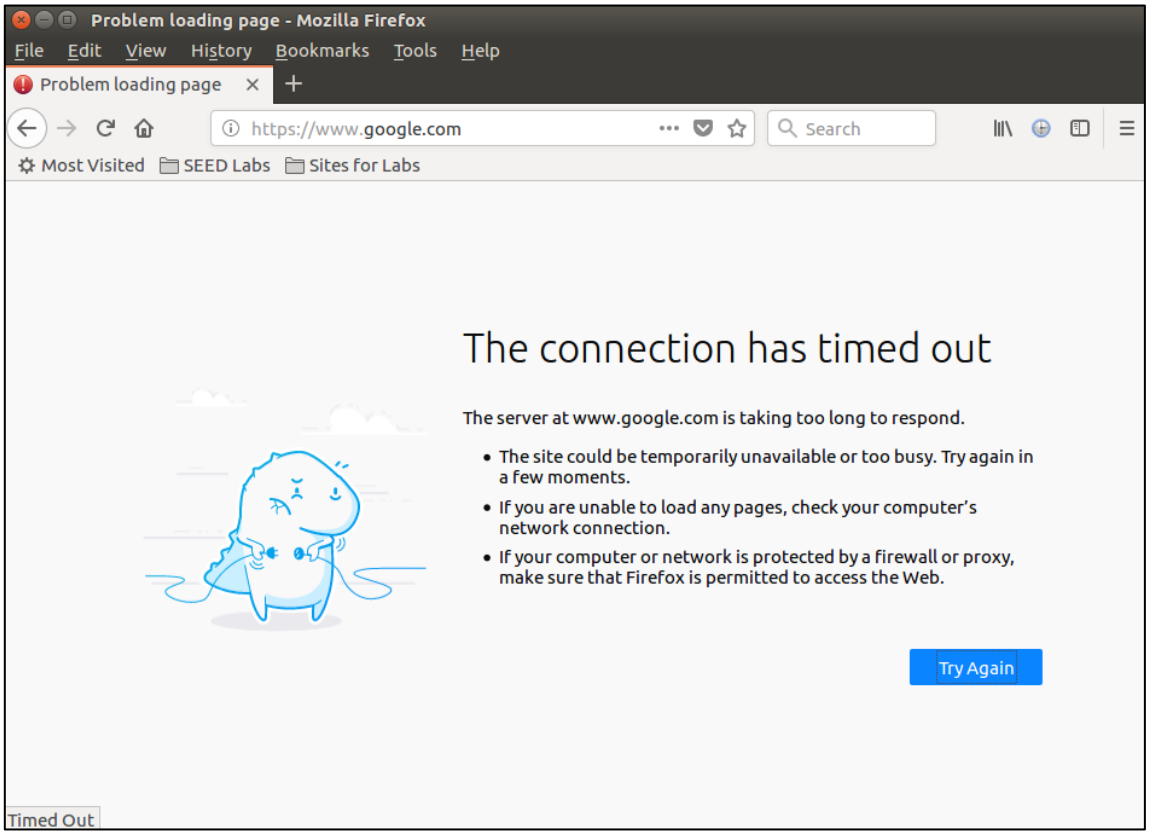
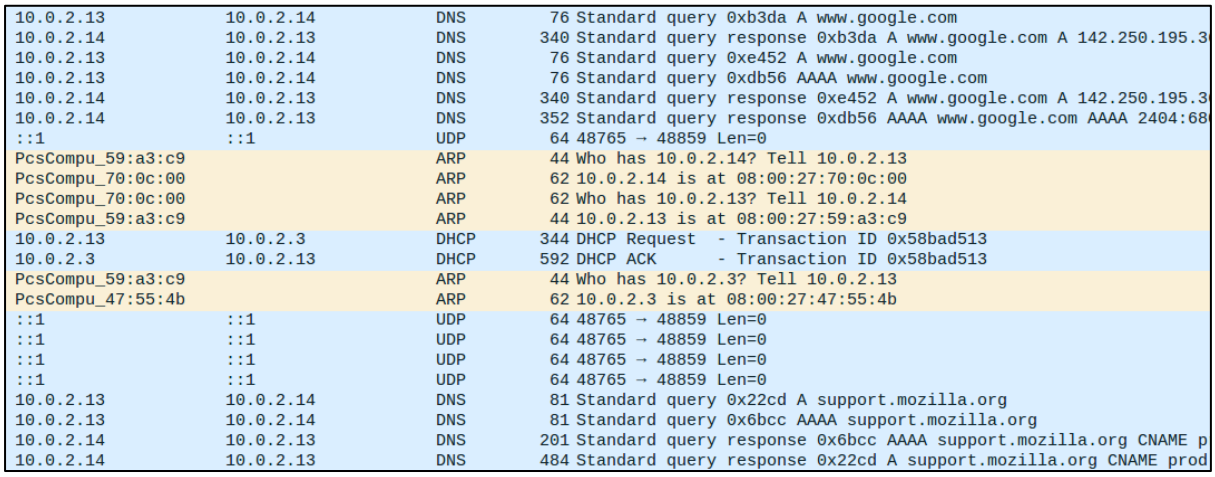


Fig. 3.2(q): Re-enabling the proxy denies the connection to www.google.com

The desired result (failed browse attempt) is obtained.



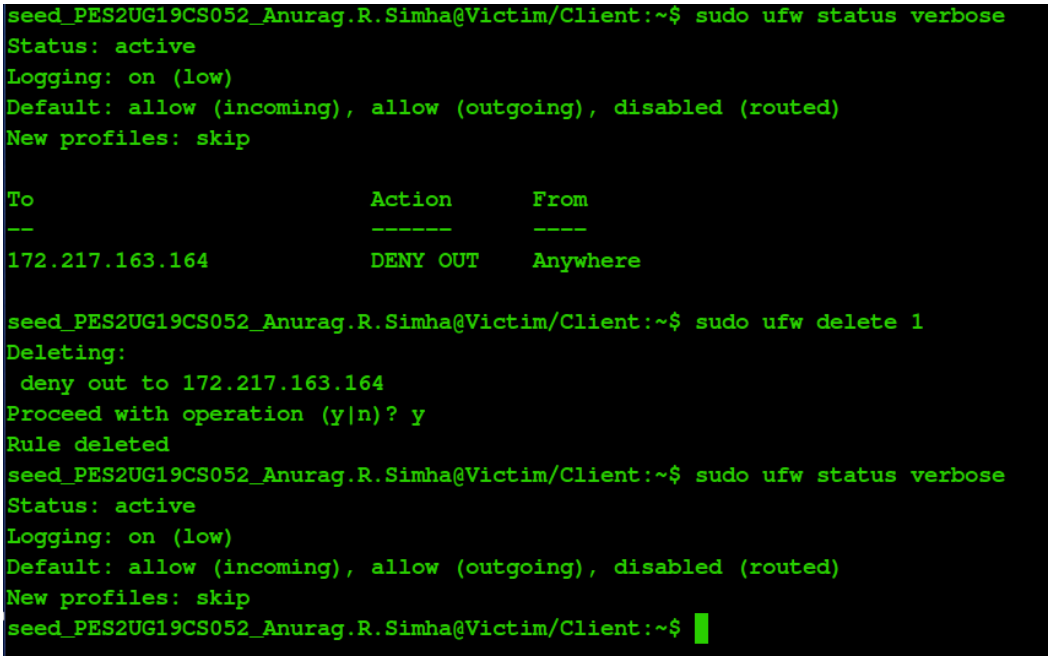
10.0.2.13	10.0.2.14	DNS	76	Standard query 0xb3da A www.google.com
10.0.2.14	10.0.2.13	DNS	340	Standard query response 0xb3da A www.google.com A 142.250.195.3
10.0.2.13	10.0.2.14	DNS	76	Standard query 0xe452 A www.google.com
10.0.2.13	10.0.2.14	DNS	76	Standard query 0xdb56 AAAA www.google.com
10.0.2.14	10.0.2.13	DNS	340	Standard query response 0xe452 A www.google.com A 142.250.195.3
10.0.2.14	10.0.2.13	DNS	352	Standard query response 0xdb56 AAAA www.google.com AAAA 2404:68
::1	::1	UDP	64	48765 → 48859 Len=0
PcsCompu_59:a3:c9		ARP	44	Who has 10.0.2.14? Tell 10.0.2.13
PcsCompu_70:0c:00		ARP	62	10.0.2.14 is at 08:00:27:70:0c:00
PcsCompu_70:0c:00		ARP	62	Who has 10.0.2.13? Tell 10.0.2.14
PcsCompu_59:a3:c9		ARP	44	10.0.2.13 is at 08:00:27:59:a3:c9
10.0.2.13	10.0.2.3	DHCP	344	DHCP Request - Transaction ID 0x58bad513
10.0.2.3	10.0.2.13	DHCP	592	DHCP ACK - Transaction ID 0x58bad513
PcsCompu_59:a3:c9		ARP	44	Who has 10.0.2.3? Tell 10.0.2.13
PcsCompu_47:55:4b		ARP	62	10.0.2.3 is at 08:00:27:47:55:4b
::1	::1	UDP	64	48765 → 48859 Len=0
::1	::1	UDP	64	48765 → 48859 Len=0
::1	::1	UDP	64	48765 → 48859 Len=0
::1	::1	UDP	64	48765 → 48859 Len=0
10.0.2.13	10.0.2.14	DNS	81	Standard query 0x22cd A support.mozilla.org
10.0.2.13	10.0.2.14	DNS	81	Standard query 0x6bcc AAAA support.mozilla.org
10.0.2.14	10.0.2.13	DNS	201	Standard query response 0x6bcc AAAA support.mozilla.org CNAME p
10.0.2.14	10.0.2.13	DNS	484	Standard query response 0x22cd A support.mozilla.org CNAME prod

Fig. 3.2(r): The connections are back to normal.

Task 4: Evade Ingress Filtering

In this task, all the incoming port 80 and port 22 connections on VM1 are blocked. But still, it'd be possible to access a web page on the web server in VM1 from VM2 by using a reverse SSH tunnel.

The previous firewall rules are erased.



```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
172.217.163.164            DENY OUT    Anywhere

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw delete 1
Deleting:
 deny out to 172.217.163.164
Proceed with operation (y|n)? y
Rule deleted
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 4(a): Deleting all the firewall rules.

The commands used are:

```
sudo ufw delete 1

sudo ufw status verbose
```

The goal is to access a secret page on VM1 (test.html) from VM2. The content of the page is shown below:

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo nano /var/www/html/test.html
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ cat /var/www/html/test.html
<HTML>
<BODY>
This is a page on VM 1 (10.0.2.13)
</BODY>
</HTML>

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 4(b): The targetted page to access.

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 4(c): The rules applied are none.

With no firewall rules setup, this page is accessed from VM 2.

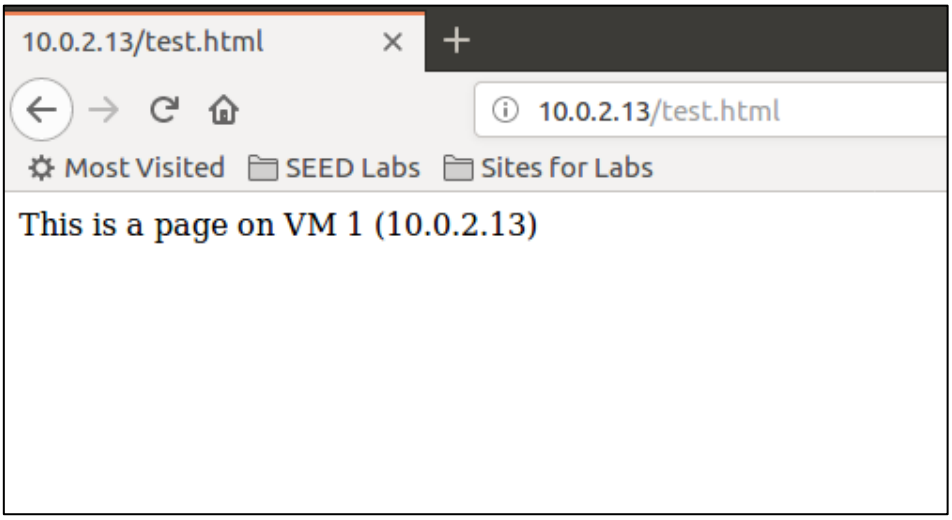


Fig. 4(d): The website can be accessed.

10.0.2.14	10.0.2.13	TCP	76	43892 → 80 [SYN] Seq=1217706574 Win
10.0.2.13	10.0.2.14	TCP	76	80 → 43892 [SYN, ACK] Seq=955073300
10.0.2.14	10.0.2.13	TCP	68	43892 → 80 [ACK] Seq=1217706575 Ack
10.0.2.14	10.0.2.13	HTTP	392	GET /test.html HTTP/1.1
10.0.2.13	10.0.2.14	TCP	68	80 → 43892 [ACK] Seq=955073301 Ack=
10.0.2.13	10.0.2.14	HTTP	417	HTTP/1.1 200 OK (text/html)
10.0.2.14	10.0.2.13	TCP	68	43892 → 80 [ACK] Seq=1217706899 Ack
10.0.2.14	10.0.2.13	HTTP	364	GET /favicon.ico HTTP/1.1
10.0.2.13	10.0.2.14	HTTP	568	HTTP/1.1 404 Not Found (text/html)
10.0.2.14	10.0.2.13	TCP	68	43892 → 80 [ACK] Seq=1217707195 Ack

Fig. 4(e): The Wireshark results.

An SSH connection is also attempted with the command, `ssh seed@10.0.2.13`

```
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$ ssh seed@10.0.2.13
seed@10.0.2.13's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

Last login: Tue Oct 19 15:38:29 2021 from 10.0.2.14
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ cd /var/www/html
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:.../html$ ls
index.html  test.html
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:.../html$ cat test.html
<HTML>
<BODY>
This is a page on VM 1 (10.0.2.13)
</BODY>
</HTML>

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:.../html$
```

Fig. 4(f): The result is a success.

10.0.2.14	10.0.2.13	SSHv2	104	Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	SSHv2	104	Server: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	TCP	68	52028 → 22 [ACK] Seq=373346809 Ack
10.0.2.14	10.0.2.13	SSHv2	104	Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	SSHv2	104	Server: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	TCP	68	52028 → 22 [ACK] Seq=373346845 Ack
10.0.2.14	10.0.2.13	SSHv2	104	Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	SSHv2	128	Server: Encrypted packet (len=60)
10.0.2.14	10.0.2.13	TCP	68	52028 → 22 [ACK] Seq=373346881 Ack
10.0.2.13	10.0.2.14	SSHv2	160	Server: Encrypted packet (len=92)
10.0.2.14	10.0.2.13	TCP	68	52028 → 22 [ACK] Seq=373346881 Ack
10.0.2.14	10.0.2.13	SSHv2	104	Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	SSHv2	104	Server: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	TCP	68	52028 → 22 [ACK] Seq=373346917 Ack
10.0.2.14	10.0.2.13	SSHv2	104	Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	SSHv2	104	Server: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	TCP	68	52028 → 22 [ACK] Seq=373346953 Ack
10.0.2.14	10.0.2.13	SSHv2	104	Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	SSHv2	104	Server: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	TCP	68	52028 → 22 [ACK] Seq=373346989 Ack
10.0.2.14	10.0.2.13	SSHv2	104	Client: Encrypted packet (len=36)

Fig. 4(g): The Wireshark packet capture results.

Next, the incoming requests on port 80 and port 22 on VM1 are blocked.

The commands:

```
sudo ufw deny in from any to 10.0.2.13 port 80
sudo ufw deny in from any to 10.0.2.13 port 22
sudo ufw status verbose
```

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw deny in from any to 10.0.2.13 port 80
Rule added
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw deny in from any to 10.0.2.13 port 22
Rule added
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
10.0.2.13 80               DENY IN     Anywhere
10.0.2.13 22               DENY IN     Anywhere

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 4(h): The rules are set.

Given the firewall rules in place on VM1, it's checked if the page is still accessible. The browser cache is also cleared.

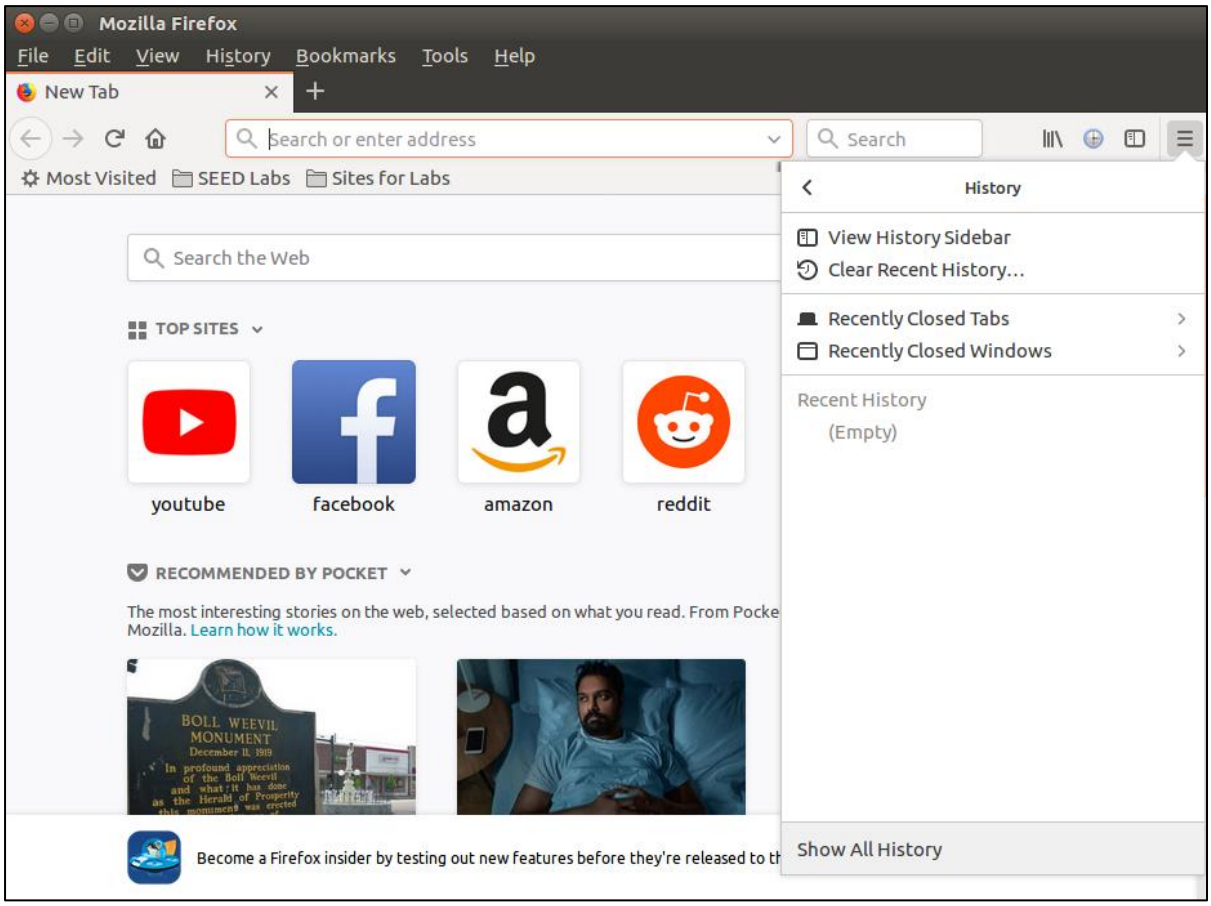


Fig. 4(i): The browser cache is cleared.

When the page is accessed again from VM2, it's no longer accessible.

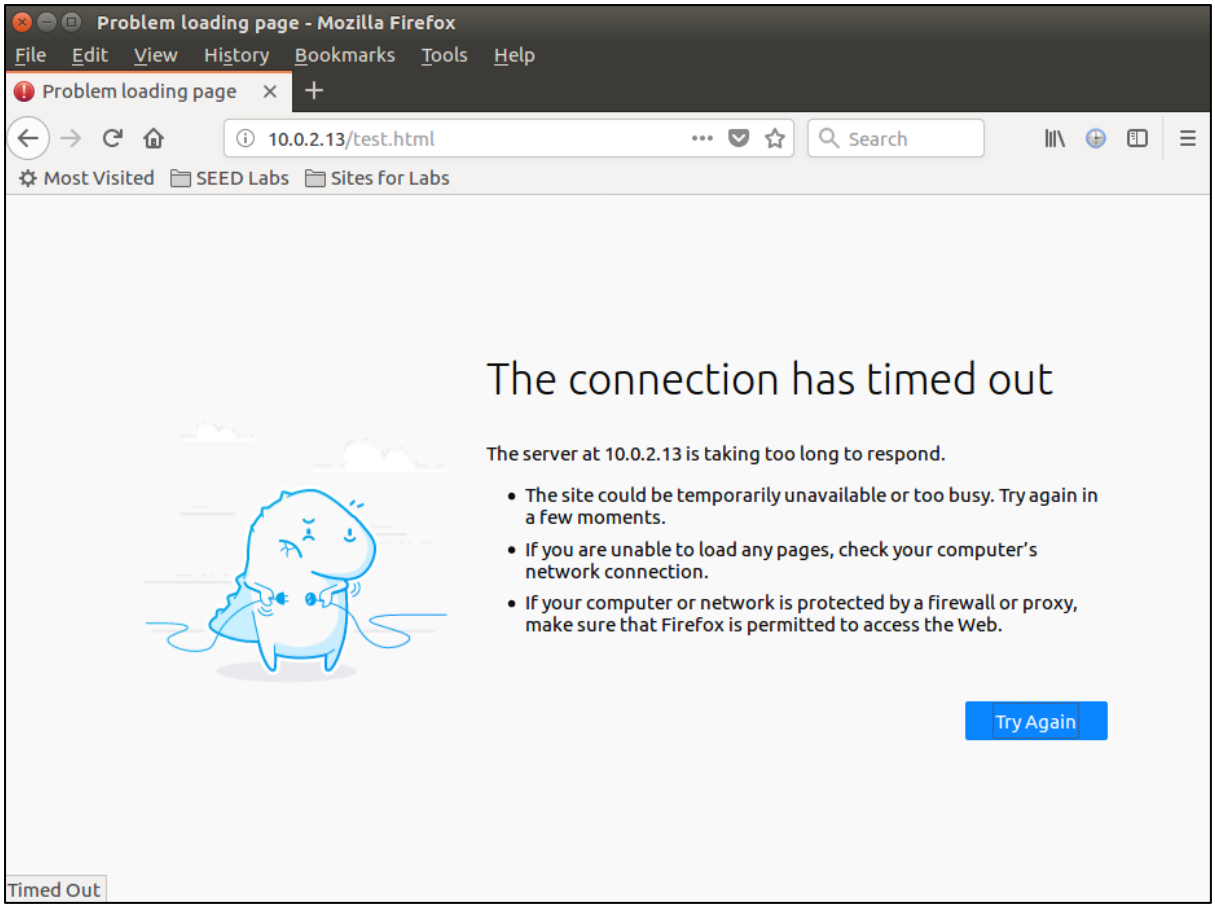


Fig. 4(j): The page is inaccessible.

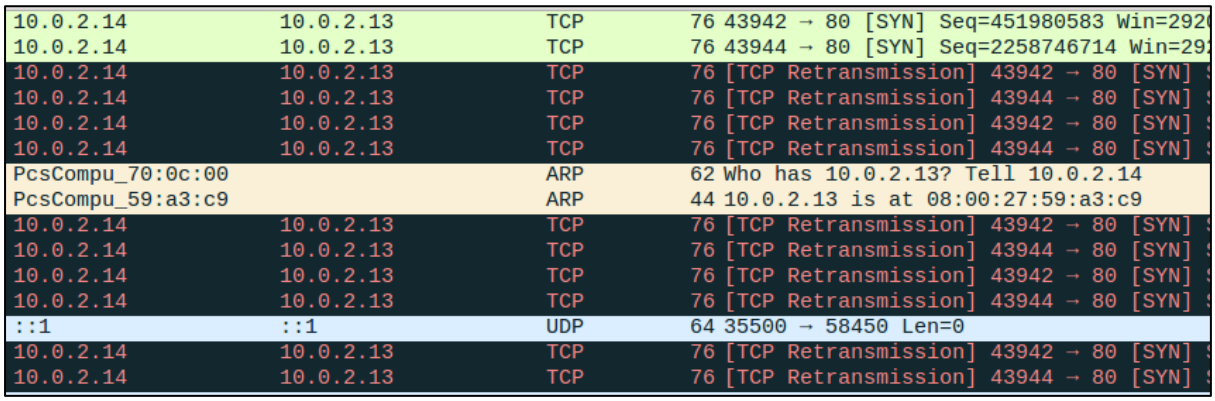


Fig. 4(k): The observations on Wireshark.

Next, an SSH connection is attempted.

The command: `ssh seed@10.0.2.13`

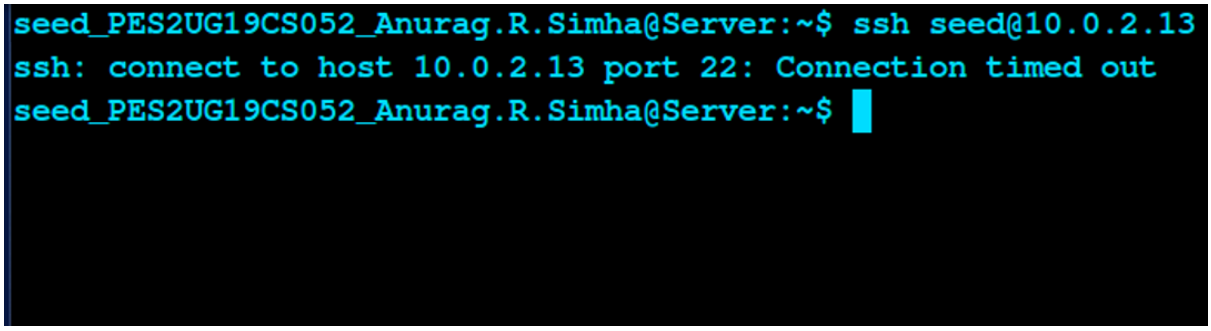


Fig. 4(*l*): The SSH connection is a failed attempt.

SSH is not contacted due to the presence of a firewall.

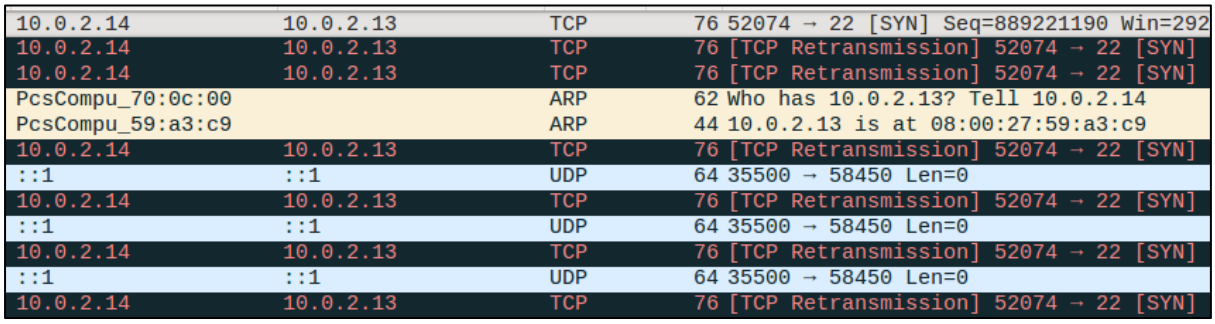


Fig. 4(m): The results on Wireshark.

Next, a reverse tunnel is setup. Using this VM2 can use its local port 8000 to access port 80 on VM1.

The command: `ssh -R 9000:10.0.2.13:80 10.0.2.14`

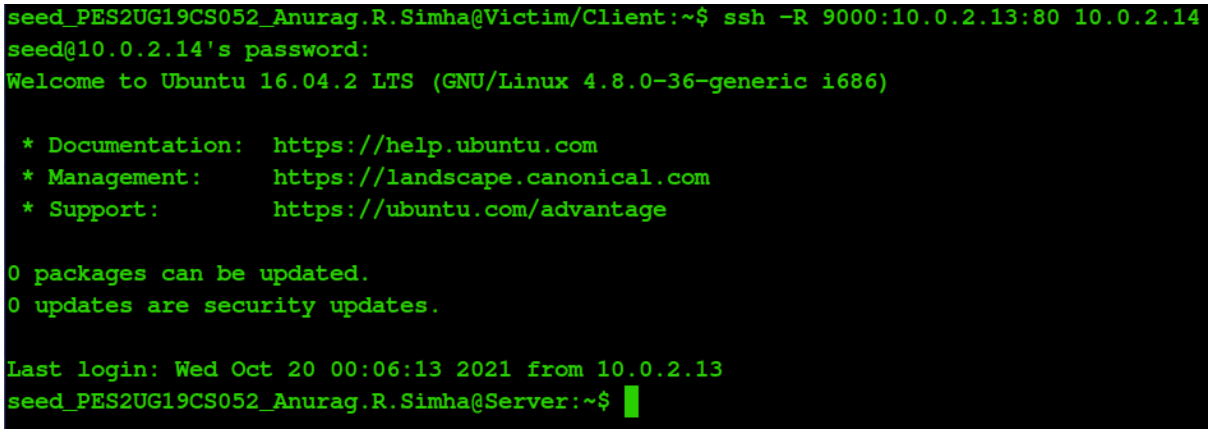


Fig. 4(n): The tunnel is opened.

Next, on the browser, the page is accessed. `localhost:9000/test.html` is browsed.

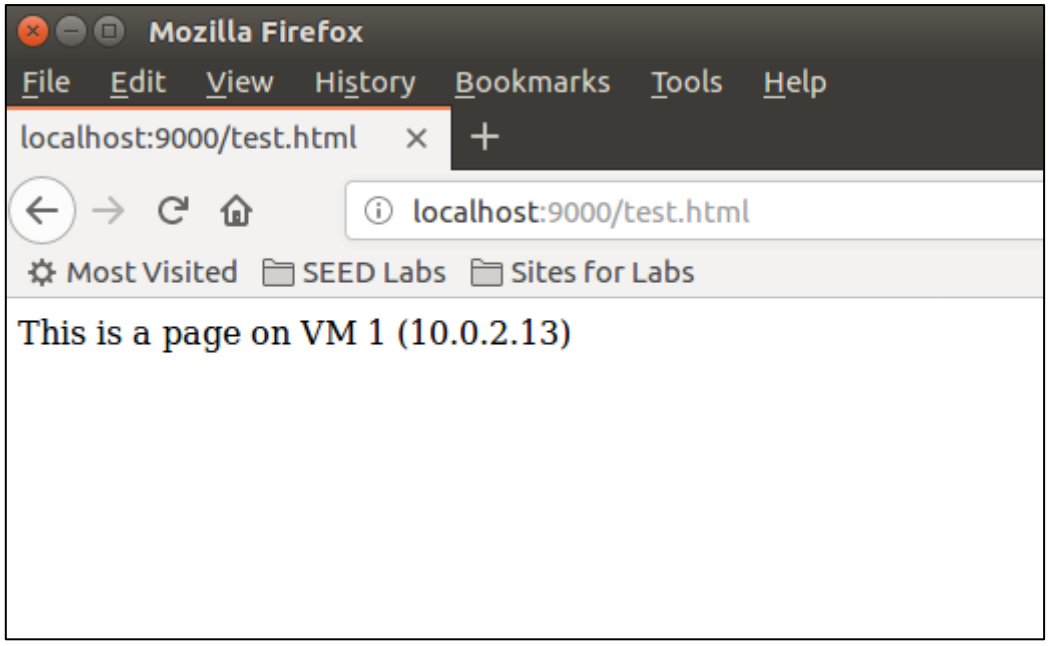


Fig. 4(o): A successful access to the page on VM 1.

When the address is typed, since the tunnel is open, localhost contacts 10.0.2.13 over port 9000. Afterwards, the desired page is fetched and returned to VM 2 from VM 1. This was the goal.

Below are the packets captured on Wireshark.

10.0.2.13	10.0.2.14	SSHv2	520	Client: Encrypted packet (len=452)
10.0.2.14	10.0.2.13	TCP	68	22 → 40188 [ACK] Seq=2351395359 Ac
10.0.2.14	10.0.2.13	SSHv2	176	Server: Encrypted packet (len=108)
10.0.2.14	10.0.2.13	SSHv2	440	Server: Encrypted packet (len=372)
10.0.2.13	10.0.2.14	TCP	68	40188 → 22 [ACK] Seq=3757651449 Ac
10.0.2.14	10.0.2.13	SSHv2	152	Server: Encrypted packet (len=84)
10.0.2.13	10.0.2.14	TCP	68	40188 → 22 [ACK] Seq=3757651449 Ac
10.0.2.14	10.0.2.13	SSHv2	160	Server: Encrypted packet (len=92)
10.0.2.13	10.0.2.14	TCP	68	40188 → 22 [ACK] Seq=3757651449 Ac
10.0.2.13	10.0.2.13	TCP	76	44632 → 80 [SYN] Seq=1056971960 Wi
10.0.2.13	10.0.2.13	TCP	76	80 → 44632 [SYN, ACK] Seq=20229839
10.0.2.13	10.0.2.13	TCP	68	44632 → 80 [ACK] Seq=1056971961 Ac
10.0.2.13	10.0.2.14	SSHv2	112	Client: Encrypted packet (len=44)
10.0.2.14	10.0.2.13	SSHv2	432	Server: Encrypted packet (len=364)
10.0.2.13	10.0.2.13	HTTP	397	GET /test.html HTTP/1.1
10.0.2.13	10.0.2.13	TCP	68	80 → 44632 [ACK] Seq=202298391 Ack
10.0.2.13	10.0.2.13	HTTP	417	HTTP/1.1 200 OK (text/html)
10.0.2.13	10.0.2.13	TCP	68	44632 → 80 [ACK] Seq=1056972290 Ac
10.0.2.13	10.0.2.14	SSHv2	456	Client: Encrypted packet (len=388)
10.0.2.14	10.0.2.13	TCP	68	22 → 40188 [ACK] Seq=2351396379 Ac
10.0.2.13	10.0.2.13	TCP	68	80 → 44632 [FIN, ACK] Seq=202298740
10.0.2.13	10.0.2.14	SSHv2	104	Client: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	TCP	68	22 → 40188 [ACK] Seq=2351396379 Ac
10.0.2.14	10.0.2.13	SSHv2	140	Server: Encrypted packet (len=72)
10.0.2.13	10.0.2.13	TCP	68	44632 → 80 [FIN, ACK] Seq=10569722
10.0.2.13	10.0.2.13	TCP	68	80 → 44632 [ACK] Seq=202298741 Ack

Fig. 4(p): The capture on Wireshark.

It's noticed that (in the green region) the address of both, the source and the destination are the same. For, the contacted address was 'localhost'.

Next, the tunnel is shut.

The command, `exit` logs out, hence shutting the tunnel.

```
Last login: Tue Oct 19 23:31:55 2021 from 10.0.2.13
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$ exit
logout
Connection to 10.0.2.14 closed.
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 4(q): Shutting the tunnel.

10.0.2.13	10.0.2.14	SSH	104 Client: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	SSH	104 Server: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	TCP	68 40194 → 22 [ACK] Seq=546193276 Ack
10.0.2.13	10.0.2.14	SSH	104 Client: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	SSH	104 Server: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	TCP	68 40194 → 22 [ACK] Seq=546193312 Ack
10.0.2.13	10.0.2.14	SSH	104 Client: Encrypted packet (len=36)
10.0.2.14	10.0.2.13	SSH	112 Server: Encrypted packet (len=44)
10.0.2.13	10.0.2.14	TCP	68 40194 → 22 [ACK] Seq=546193348 Ack
10.0.2.14	10.0.2.13	SSH	244 Server: Encrypted packet (len=176)
10.0.2.13	10.0.2.14	TCP	68 40194 → 22 [ACK] Seq=546193348 Ack
10.0.2.13	10.0.2.14	SSH	104 Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.14	SSH	128 Client: Encrypted packet (len=60)
10.0.2.13	10.0.2.14	TCP	68 40194 → 22 [FIN, ACK] Seq=54619344
10.0.2.14	10.0.2.13	TCP	68 22 → 40194 [ACK] Seq=3774291350 Ac
10.0.2.14	10.0.2.13	TCP	68 22 → 40194 [FIN, ACK] Seq=37742913
10.0.2.13	10.0.2.14	TCP	68 40194 → 22 [ACK] Seq=546193445 Ack

Fig. 4(r): The capture on Wireshark.

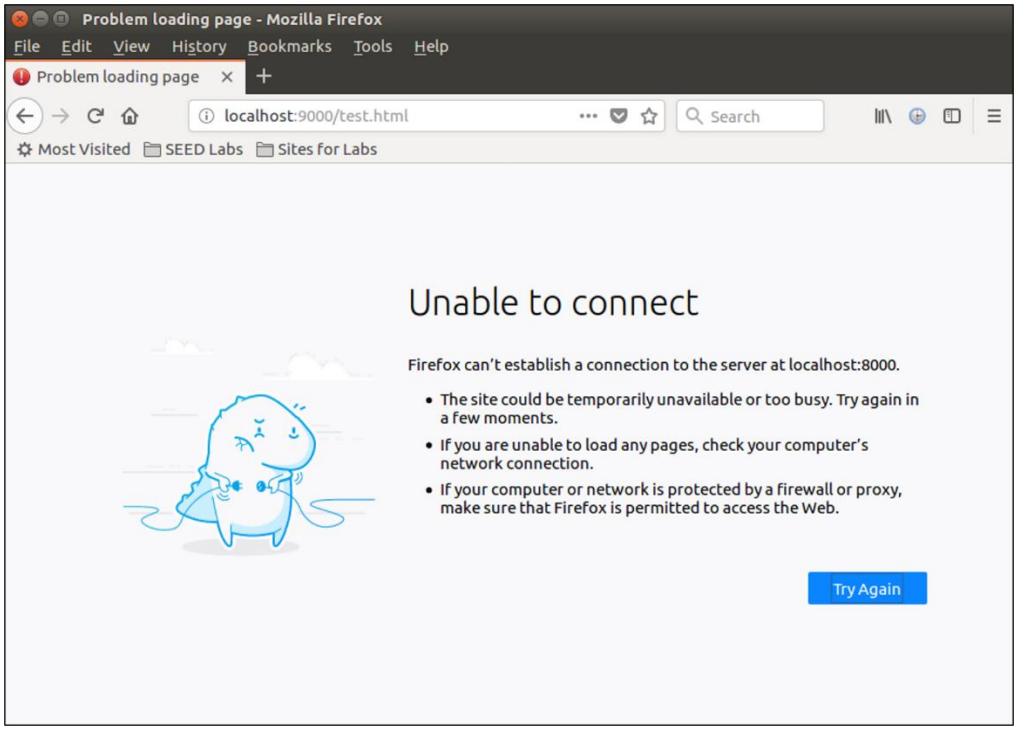


Fig. 4(s): The page is inaccessible.

With the tunnel broken, the page on VM1 is no longer accessible from VM2.

(Additional work)

Now, a connection to the SSH server on VM 1 from VM 2 is tried to access.

The command:

`ssh -R 9000:10.0.2.13:22 seed@10.0.2.14`

```
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ ssh -R 9000:10.0.2.13:22 seed@10.0.2.14
seed@10.0.2.14's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

Last login: Sun Oct 24 05:50:18 2021 from 10.0.2.13
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$
```

Fig. 4A(a): Opening the tunnel to contact SSH.

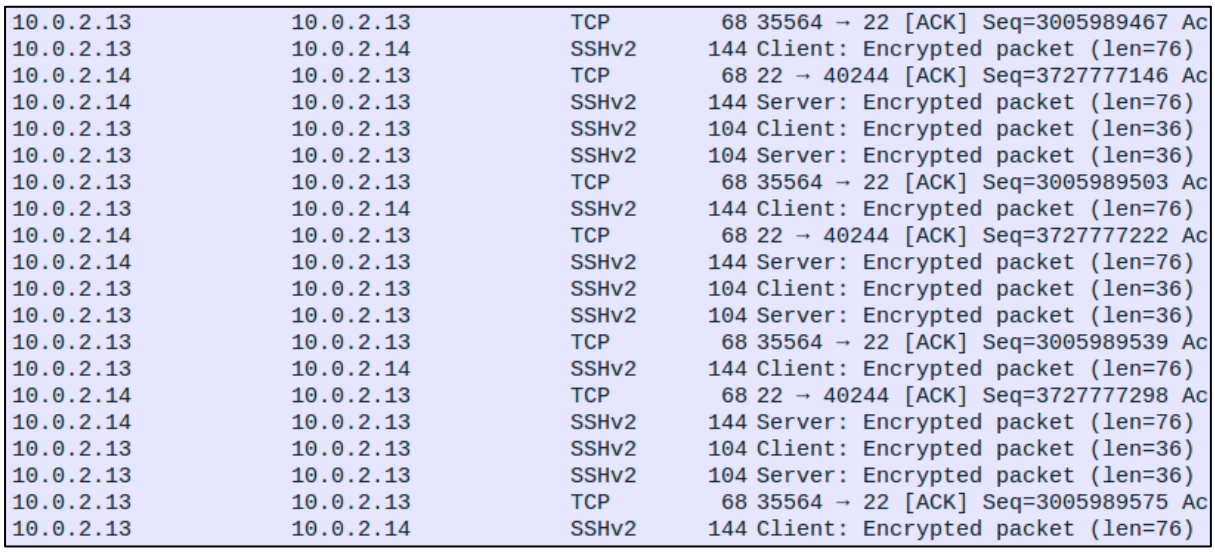
On VM 2, the command, `ssh localhost -p 9000` gives access to VM 1.

```
seed_PES2UG19CS052_Anurag.R.Simha@Server:~$ ssh localhost -p 9000
seed@localhost's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

0 packages can be updated.
0 updates are security updates.

Last login: Wed Oct 20 00:06:24 2021 from 10.0.2.13
seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:59:a3:c9
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::5f33:85f1:5546:41d0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:176 errors:0 dropped:0 overruns:0 frame:0
          TX packets:175 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:29603 (29.6 KB)  TX bytes:26358 (26.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:209 errors:0 dropped:0 overruns:0 frame:0
          TX packets:209 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:45483 (45.4 KB)  TX bytes:45483 (45.4 KB)

seed_PES2UG19CS052_Anurag.R.Simha@Victim/Client:~$
```

Fig. 4A(b): The successful SSH connection to 10.0.2.13 (VM 1) from 10.0.2.14 (VM 2).

When the localhost is contacted, port forwarding occurs and 10.0.2.13 is contacted over port 9000.

Below are the packets captured on Wireshark.



10.0.2.13	10.0.2.13	TCP	68	35564 → 22 [ACK] Seq=3005989467 Ac
10.0.2.13	10.0.2.14	SSHv2	144	Client: Encrypted packet (len=76)
10.0.2.14	10.0.2.13	TCP	68	22 → 40244 [ACK] Seq=3727777146 Ac
10.0.2.14	10.0.2.13	SSHv2	144	Server: Encrypted packet (len=76)
10.0.2.13	10.0.2.13	SSHv2	104	Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.13	SSHv2	104	Server: Encrypted packet (len=36)
10.0.2.13	10.0.2.13	TCP	68	35564 → 22 [ACK] Seq=3005989503 Ac
10.0.2.13	10.0.2.14	SSHv2	144	Client: Encrypted packet (len=76)
10.0.2.14	10.0.2.13	TCP	68	22 → 40244 [ACK] Seq=3727777222 Ac
10.0.2.14	10.0.2.13	SSHv2	144	Server: Encrypted packet (len=76)
10.0.2.13	10.0.2.13	SSHv2	104	Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.13	SSHv2	104	Server: Encrypted packet (len=36)
10.0.2.13	10.0.2.13	TCP	68	35564 → 22 [ACK] Seq=3005989539 Ac
10.0.2.13	10.0.2.14	SSHv2	144	Client: Encrypted packet (len=76)
10.0.2.14	10.0.2.13	TCP	68	22 → 40244 [ACK] Seq=3727777298 Ac
10.0.2.14	10.0.2.13	SSHv2	144	Server: Encrypted packet (len=76)
10.0.2.13	10.0.2.13	SSHv2	104	Client: Encrypted packet (len=36)
10.0.2.13	10.0.2.13	SSHv2	104	Server: Encrypted packet (len=36)
10.0.2.13	10.0.2.13	TCP	68	35564 → 22 [ACK] Seq=3005989575 Ac
10.0.2.13	10.0.2.14	SSHv2	144	Client: Encrypted packet (len=76)

Fig. 4A(c): The Wireshark capture.

Henceforth, the firewall rules were evaded while accessing both the ports (80 and 22).

****