

NSSII Exercise 2

Manavjeet Singh, 2018295

VMs	IP address	Comments
VM1	10.0.0.1/24	Client in Task 2 and 3
VM2	10.0.0.2/24	Server in Task 2 and 3
VM3	10.0.0.3/24	-
VM4	10.0.0.4/24	-
VM5	10.0.0.5/24	-

Part 1

Creating Bridge:

- `sudo ifconfig eth1 0.0.0.0`
- `sudo ifconfig eth2 0.0.0.0`
- `sudo brctl addbr br0`
- `sudo brctl addif br0 eth1 eth2`
- `sudo ip addr add 10.0.0.2/24 dev br0`
- `sudo ip link set dev br0 up`

```
user@artixVM:~  
user@artixVM:~ 108x27  
inet 127.0.0.1/8 scope host lo  
    valid_lft forever preferred_lft forever  
inet6 ::1/128 scope host  
    valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,DYNAMIC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 08:00:27:ef:d5:49 brd ff:ff:ff:ff:ff:ff  
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0  
        valid_lft forever preferred_lft forever  
    inet6 fe80::a00:27ff:feef:d549/64 scope link  
        valid_lft forever preferred_lft forever  
3: eth1: <BROADCAST,MULTICAST,DYNAMIC,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0 state UP group default  
    qlen 1000  
    link/ether 08:00:27:76:3c:cd brd ff:ff:ff:ff:ff:ff  
    inet6 fe80::a00:27ff:fe76:3ccd/64 scope link  
        valid_lft forever preferred_lft forever  
4: eth2: <BROADCAST,MULTICAST,DYNAMIC,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0 state UP group default  
    qlen 1000  
    link/ether 08:00:27:5b:29:f1 brd ff:ff:ff:ff:ff:ff  
    inet6 fe80::a00:27ff:fe5b:29f1/64 scope link  
        valid_lft forever preferred_lft forever  
5: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000  
    link/ether 08:00:27:5b:29:f1 brd ff:ff:ff:ff:ff:ff  
    inet 10.0.0.2/24 scope global br0  
        valid_lft forever preferred_lft forever  
    inet6 fe80::a00:27ff:fe5b:29f1/64 scope link  
        valid_lft forever preferred_lft forever  
user@artixVM
```

1 | to IPv6 | W: down | E: 10.0.2.15 (1000 Mbit/s) | FULL 100.00% | 41.0 GiB | 0.00 | MEMORY < 735.1 MiB | 2021-03-15 13:15:30

Bridge settings

```
user@artixVM:~ user@artixVM:~ 97x18  
rtt min/avg/max/ndev = 1.573/1.630/1.687/0.057 ms  
user@artixVM ping 10.0.0.5  
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.  
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=1.89 ms  
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=1.73 ms  
^C  
--- 10.0.0.5 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 100ms  
rtt min/avg/max/ndev = 1.731/1.888/1.885/0.077 ms  
user@artixVM ping 10.0.0.2  
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.  
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.954 ms  
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.00 ms  
^C  
--- 10.0.0.2 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 100ms  
rtt min/avg/max/ndev = 0.954/0.978/1.003/0.024 ms  
user@artixVM ping 10.0.0.1  
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.  
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=1.69 ms  
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.69 ms  
^C  
--- 10.0.0.1 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 100ms  
rtt min/avg/max/ndev = 1.693/1.693/1.693/0.000 ms  
user@artixVM ping 10.0.0.2  
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.  
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.16 ms  
^C  
--- 10.0.0.2 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/ndev = 1.161/1.161/1.161/0.000 ms  
user@artixVM
```

1 | down | E: 10.0.2.15 (1000 Mbit/s) | FULL 100.00% | 39.8 GiB | 0.06 | MEMORY < 740.8 MiB | 2021-03-15 13:14:25

```
user@artixVM:~ user@artixVM:~ 97x18  
valid_lft forever preferred_lft forever  
user@artixVM sudo ip addr add 10.0.0.4/24 dev enp0s8  
user@artixVM ping 10.0.0.1  
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.  
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=2.90 ms  
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.76 ms  
^C  
--- 10.0.0.1 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 100ms  
rtt min/avg/max/ndev = 1.761/2.370/2.979/0.609 ms  
user@artixVM ping 10.0.0.2  
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.  
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.84 ms  
^C  
--- 10.0.0.2 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/ndev = 1.836/1.836/1.836/0.000 ms  
user@artixVM
```

1 | down | E: 10.0.2.15 (1000 Mbit/s) | FULL 100.00% | 41.1 GiB | 0.02 | MEMORY < 731.5 MiB | 2021-03-15 13:14:25

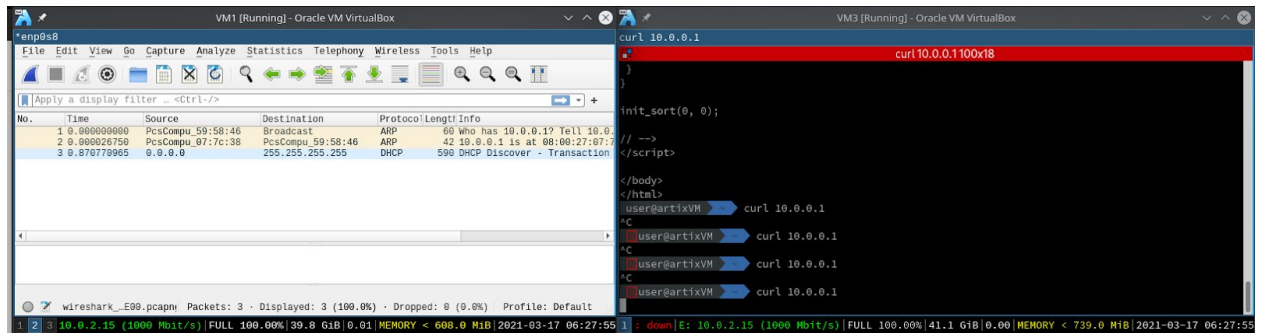
```
user@artixVM:~ user@artixVM:~ 97x18  
user@artixVM ping 10.0.0.1  
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.  
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=3.37 ms  
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.69 ms  
^C  
--- 10.0.0.1 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 100ms  
rtt min/avg/max/ndev = 1.689/2.529/3.370/0.840 ms  
user@artixVM ping 10.0.0.2  
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.  
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.04 ms  
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.983 ms  
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.847 ms  
^C  
--- 10.0.0.2 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2003ms  
rtt min/avg/max/ndev = 0.847/1.289/2.039/0.532 ms  
user@artixVM
```

1 | down | E: 10.0.2.15 (1000 Mbit/s) | FULL 100.00% | 41.1 GiB | 0.01 | MEMORY < 732.6 MiB | 2021-03-15 13:14:25

Machines are able to ping each other

1. To block VM1 from VM3: `sudo ebtables -I FORWARD -s 08:00:27:59:58:46 -d 08:00:27:07:7c:38 -j DROP`

When accessing server on VM1 from VM3:



When accessing server on VM1 from VM4:

VM1 [Running] - Oracle VM VirtualBox

*enp0s8

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	0.0.0.0	255.255.255.255	DHCP	590	DHCP Discover - Trans
2	1.105047220	PcsCompu_b1:d9:65	Broadcast	ARP	60	Who has 10.0.0.1? Tell
3	1.105080063	PcsCompu_07:7c:38	PcsCompu_b1:d9:65	ARP	42	10.0.0.1 is at 08:00:2
4	1.106308444	10.0.0.4	10.0.0.1	TCP	74	43342 → 80 [SYN] Seq=0
5	1.106340353	10.0.0.1	10.0.0.4	TCP	74	80 → 43342 [SYN, ACK]
6	1.107458926	10.0.0.4	10.0.0.1	TCP	66	43342 → 80 [ACK] Seq=1
7	1.107459231	10.0.0.4	10.0.0.1	HTTP	138	GET / HTTP/1.1
8	1.107500074	10.0.0.1	10.0.0.4	TCP	66	80 → 43342 [ACK] Seq=1
9	1.107761455	10.0.0.1	10.0.0.4	HTTP	6086	HTTP/1.1 200 OK (text
10	1.108547963	10.0.0.4	10.0.0.1	TCP	66	43342 → 80 [ACK] Seq=7

Frame 1: 590 bytes on wire (4720 bits), 590 bytes captured (4720 bits) on interface enp0s8, id 0
 Ethernet II, Src: PcsCompu_fe:f2:84 (08:00:27:fe:f2:84), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
 User Datagram Protocol, Src Port: 68, Dst Port: 67

Internet ... 20 byte: Packets: 13 · Displayed: 13 (100.0%) · Dropped: 0 (0.0%) · Profile: Default

1 2 3 10.0.2.15 (1000 Mbit/s) | FULL 100.00% | 39.8 GiB | 0.00 | MEMORY < 611.0 MiB | 2021-03-17 06:30:40

Right Ctrl

VM4 [Running] - Oracle VM VirtualBox

user@artixVM:~

```

user@artixVM:~ 97x18
var lnk = row[init_sort_column].firstChild;
if (ascending) {
  var span = lnk.childNodes[1];
  span.setAttribute('sortdir','down');
}
resort(lnk);
//}
}
}
init_sort(0, 0);

// -->
</script>

</body>
</html>
user@artixVM

```

1 down | E: 10.0.2.15 (1000 Mbit/s) | FULL 100.00% | 41.1 GiB | 0.06 | MEMORY < 742.6 MiB | 2021-03-17 06:30:35

Right Ctrl

We can see that no IP4 packets are received from VM3 on VM1.

2. Initially:

The image shows two Oracle VM VirtualBox windows. The top window, titled 'VM1 [Running] - Oracle VM VirtualBox', displays a network capture from the 'eth1' interface. The capture shows two packets: packet 3 from 10.0.0.4 to 10.0.0.1, and packet 4 from 10.0.0.1 to 10.0.0.4. The details for packet 3 are expanded, showing it is an Ethernet II frame with source MAC 08:00:27:07:7c:38 and destination MAC 08:00:27:b1:d9:65. The protocol is IPv4, and the application is ICMP. The bottom window, titled 'VM4 [Running] - Oracle VM VirtualBox', shows the network configuration for 'artixVM'. It lists three interfaces: enp0s3, enp0s8, and enp0s8. The configuration for enp0s8 is highlighted, showing its MAC address as 08:00:27:b1:d9:65 and its IP address as 10.0.0.4/24. The status bar at the bottom of the VM4 window shows network statistics: 'no IPv6 | W: down | E: 10.0.2.15 (1000 Mbit/s) | FULL 100.00% | 41.1 GiB | 0.03 | MEMORY < 730.8 MiB | 2021-03-15 13:44:10'.

VM1 [Running] - Oracle VM VirtualBox

Capturing from eth1

File Edit View Go Capture Analyze Statistics

icmp

No.	Time	Source	Destination
3	3.068419100	10.0.0.4	10.0.0.1
4	3.068459902	10.0.0.1	10.0.0.4

Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth1

Ethernet II, Src: PcsCompu_b1:d9:65 (08:00:27:b1:d9:65), Dst: PcsCompu_07:7c:38 (08:00:27:07:7c:38)

Destination: PcsCompu_07:7c:38 (08:00:27:07:7c:38)

Source: PcsCompu_b1:d9:65 (08:00:27:b1:d9:65)

Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 10.0.0.4, Dst: 10.0.0.1

Internet Control Message Protocol

user@artixVM:~

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid_lft forever preferred_lft forever

inet6 ::1/128 scope host

valid_lft forever preferred_lft forever

2: eth0: <BROADCAST,MULTICAST,DYNAMIC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000

link/ether 08:00:27:49:9e:21 brd ff:ff:ff:ff:ff:ff

inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0

valid_lft forever preferred_lft forever

inet6 fe80::a00:27ff:fe49:9e21/64 scope link

valid_lft forever preferred_lft forever

3: eth1: <BROADCAST,MULTICAST,DYNAMIC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000

link/ether 08:00:27:07:7c:38 brd ff:ff:ff:ff:ff:ff

inet 10.0.0.1/24 scope global eth1

valid_lft forever preferred_lft forever

inet 169.254.110.133/16 brd 169.254.255.255 scope global eth1

valid_lft forever preferred_lft forever

user@artixVM:~ 113x18

valid_lft forever preferred_lft forever

inet6 ::1/128 scope host

valid_lft forever preferred_lft forever

2: enp0s3: <BROADCAST,MULTICAST,DYNAMIC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000

link/ether 08:00:27:71:43:04 brd ff:ff:ff:ff:ff:ff

inet 10.0.2.15/24 brd 10.0.2.255 scope global enp0s3

valid_lft forever preferred_lft forever

inet6 fe80::a00:27ff:fe71:4304/64 scope link

valid_lft forever preferred_lft forever

3: enp0s8: <BROADCAST,MULTICAST,DYNAMIC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000

link/ether 08:00:27:b1:d9:65 brd ff:ff:ff:ff:ff:ff

inet 10.0.0.4/24 scope global enp0s8

valid_lft forever preferred_lft forever

inet 169.254.67.219/16 brd 169.254.255.255 scope global enp0s8

valid_lft forever preferred_lft forever

inet6 fe80::a00:27ff:feb1:d965/64 scope link

valid_lft forever preferred_lft forever

user@artixVM

1 no IPv6 | W: down | E: 10.0.2.15 (1000 Mbit/s) | FULL 100.00% | 41.1 GiB | 0.03 | MEMORY < 730.8 MiB | 2021-03-15 13:44:10

Right Ctrl

Using commands:

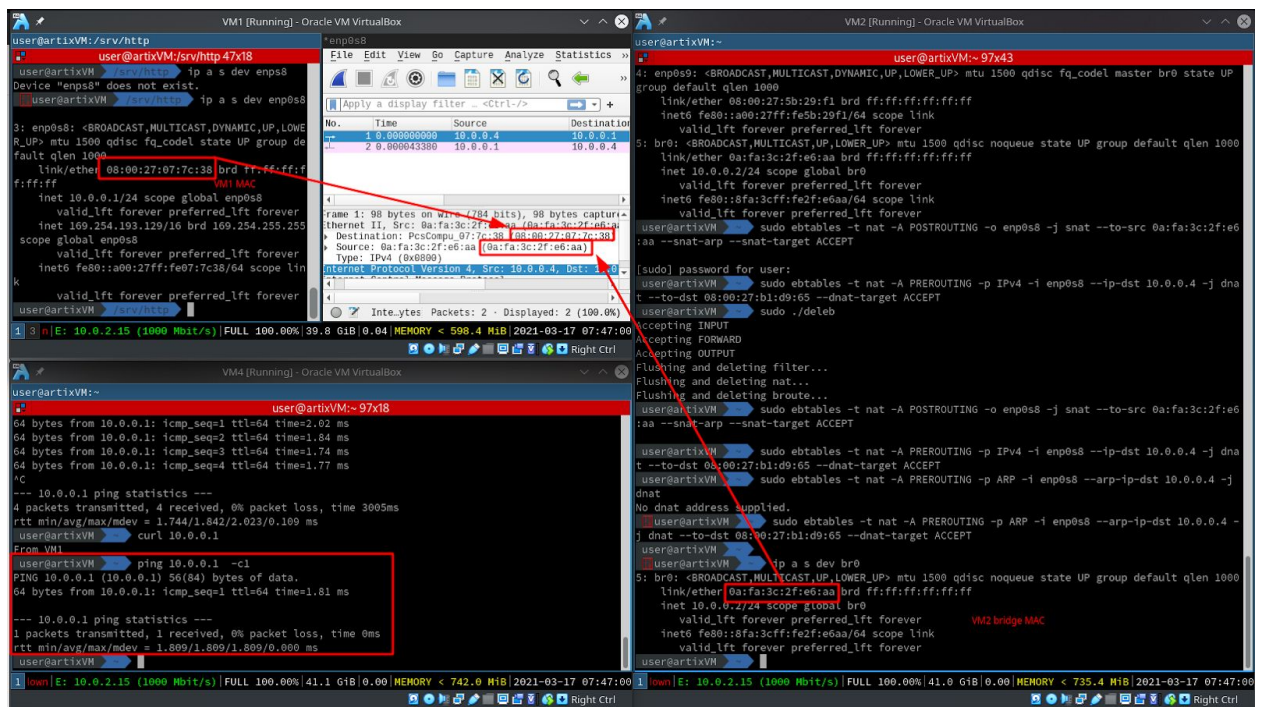
For source nat:

```
sudo ebtables -t nat -A POSTROUTING -o enp0s8 -j snat --to-src 0a:fa:3c:2f:e6:aa --snat-arp --snat-target ACCEPT
```

For destination nat, mapping between IP address and MAC for VM4:

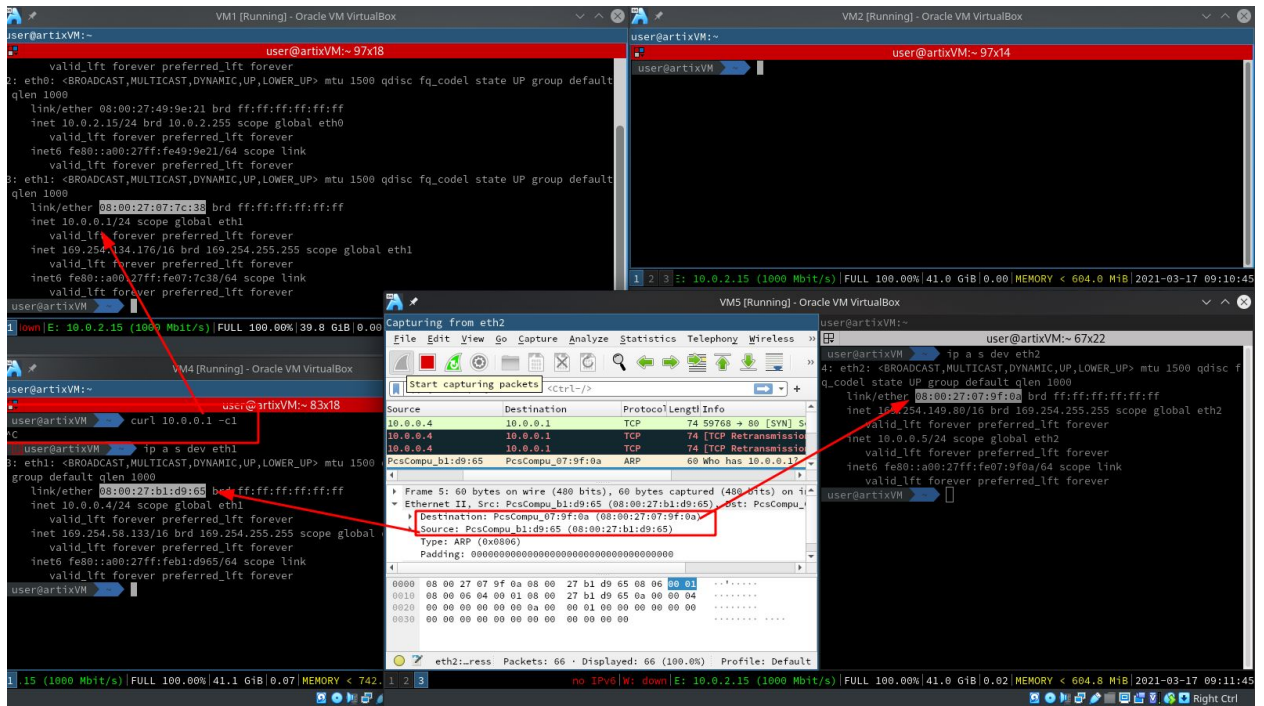
```
sudo ebtables -t nat -A PREROUTING -p IPv4 -i enp0s8 --ip-dst 10.0.0.4 -j dnat --to-dst 08:00:27:b1:d9:65 --dnat-target ACCEPT
sudo ebtables -t nat -A PREROUTING -p ARP -i enp0s8 --arp-ip-dst 10.0.0.4 -j dnat --to-dst 08:00:27:b1:d9:65 --dnat-target ACCEPT
```


Similarly this can be done for VM3 and VM5.



3. Since the source and nat-ed output should be on the different interfaces on a bridge, the following changes were made:
 - 1) Another interface was added to VM2(eth3) and VM5(eth2: 10.0.0.5/24)
 - 2) Added eth3 to existing `br0`
 Command to do MAC address NAT, all the packets of destination VM1(08:00:27:07:7c:38) will be forwarded to VM5(08:00:27:07:9f:0a) but VM5 will drop the packet since it's IP address is different from destination IP address in the packet :


```
sudo ebtables -t nat -A PREROUTING -d 08:00:27:07:7c:38 -i eth2 -j dnat --to-destination 08:00:27:07:9f:0a
```



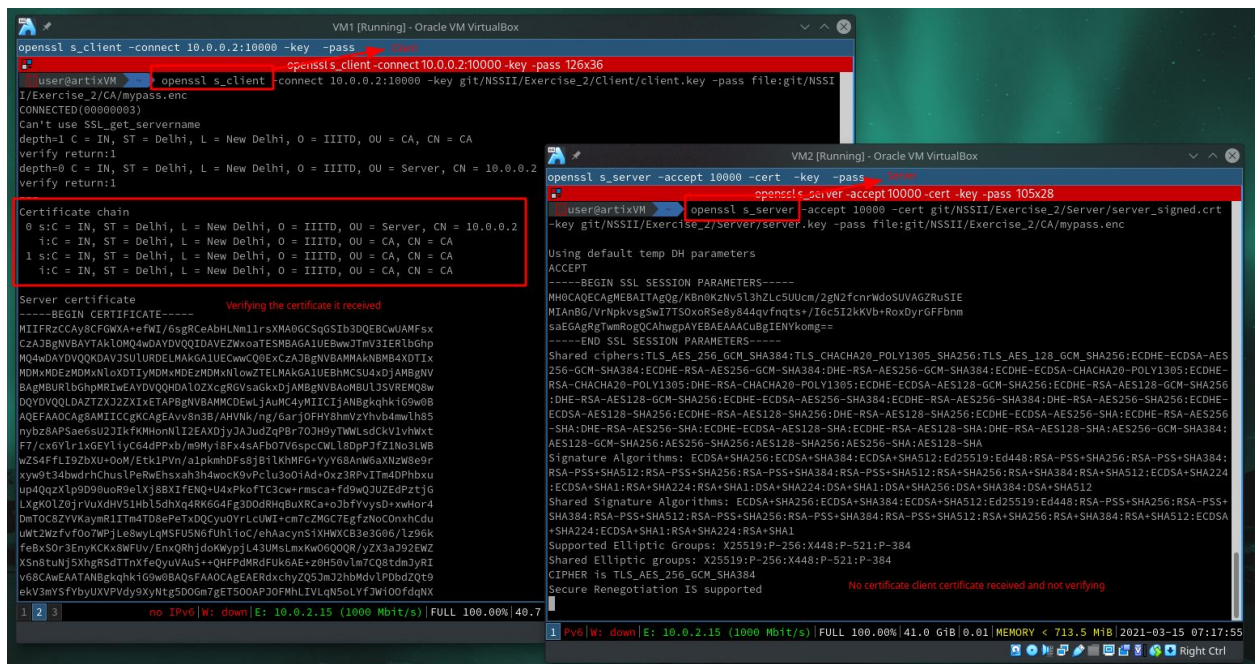
4. This thing cannot be attained by IP tables because bridge network work at Layer 2 and packet is forwarded before accessing through IP table hooks which work at Layer 3.

Part 2

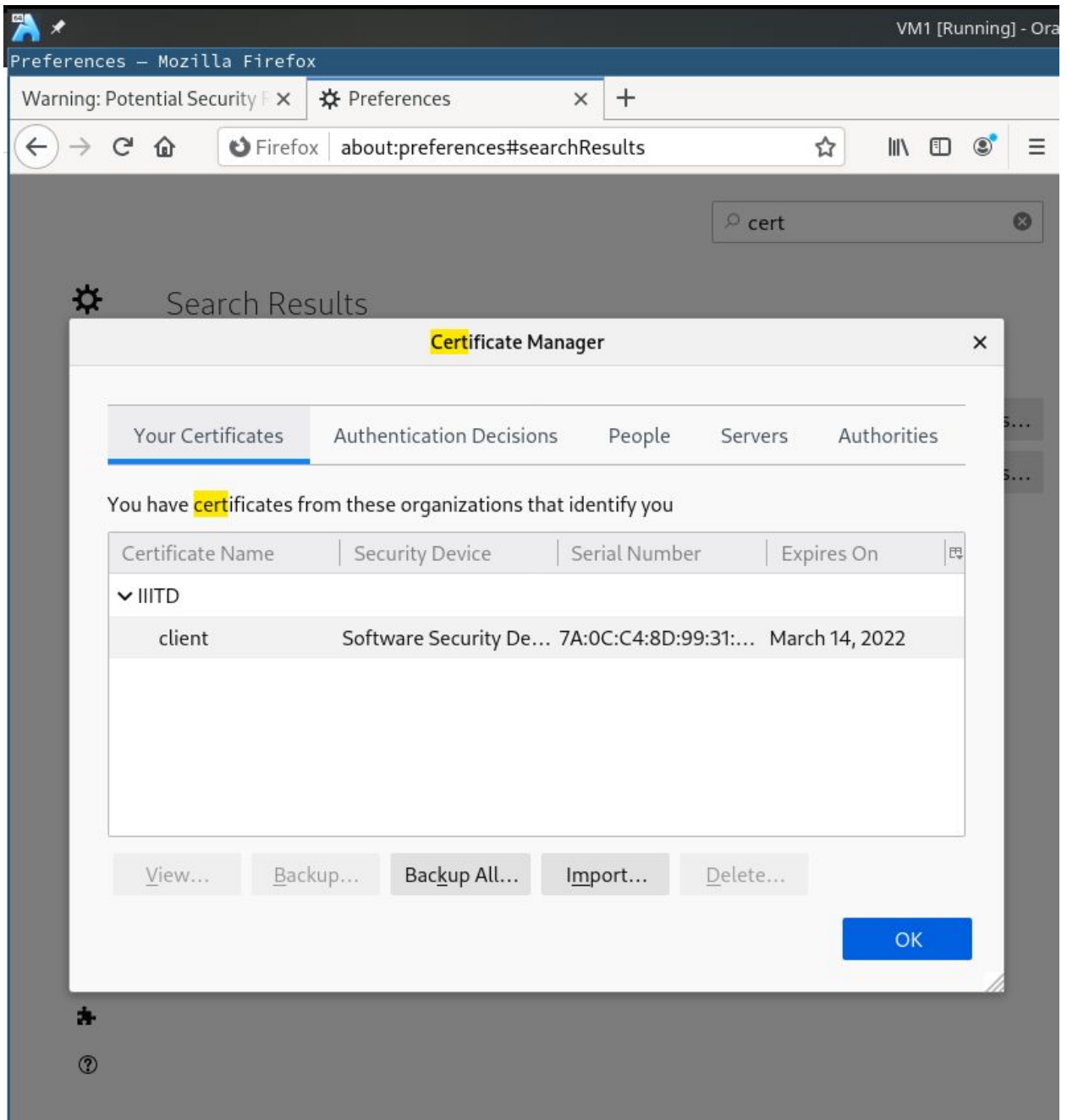
1. On the host machine:
 - a. For CA,
 - i. `openssl genrsa -out ca.key 4096`. Creates a new private key for CA.
 - ii. `openssl req -new -x509 -days 365 -key ca.key -out ca.cert.pem`. Creates a self signed certificate in x509 format for the CA.
 - b. For Server
 - i. `openssl genrsa -out server.key 4096`. Creates a new private key for the server.
 - ii. `openssl req -new -key server.key -out server.csr`. Creates a new certificate signing request for the server.
 - iii. `openssl x509 -req -days 365 -in server.csr -CA ../CA/ca.cert.pem -CAkey ../CA/ca.key -CAcreateserial -out server_signed.crt`. Creates a certificate for the server signed by the CA.
 - c. For Client
 - i. `openssl genrsa -out client.key 4096`. Creates a new private key for the client.

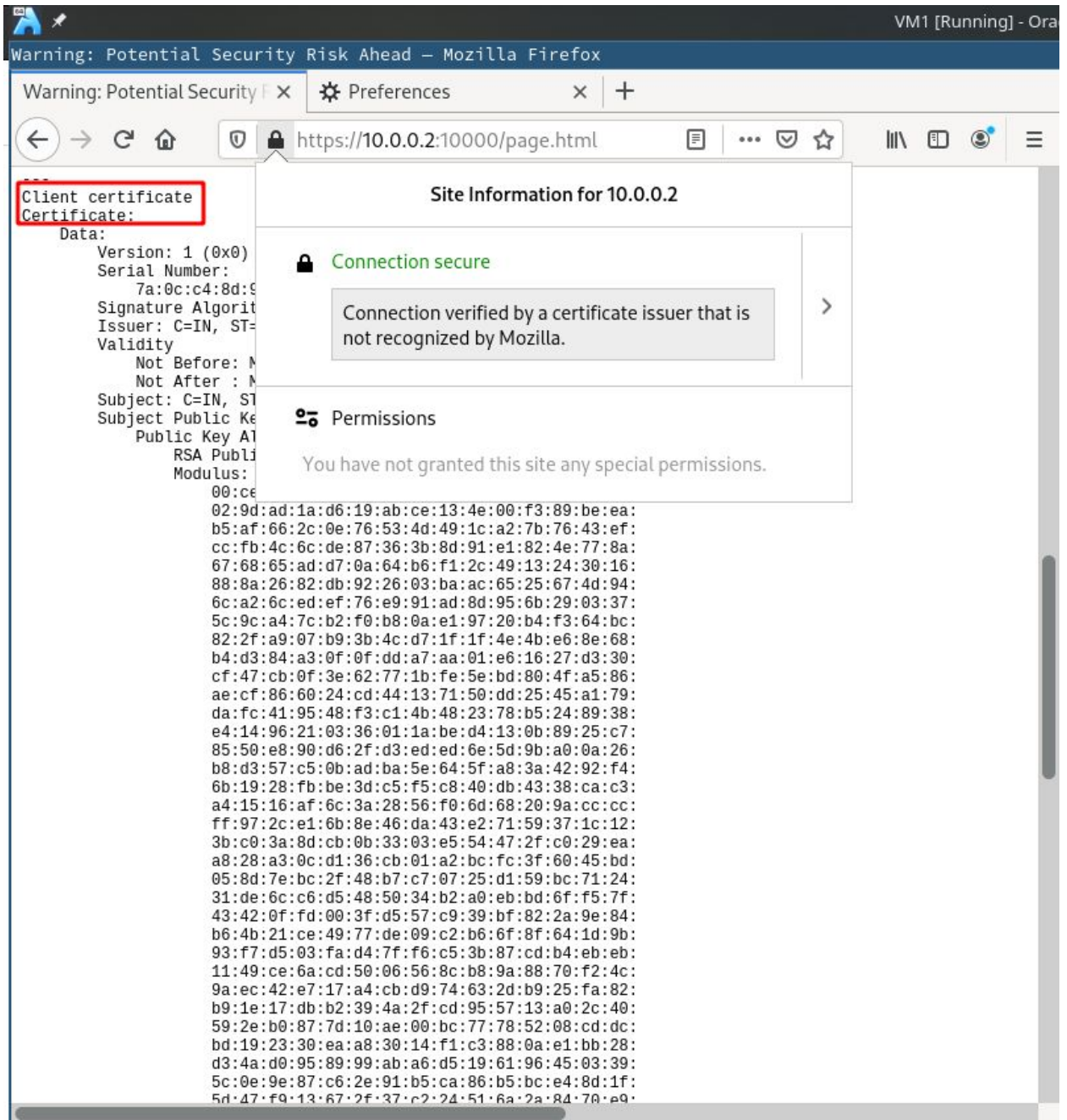
- ii. `openssl req -new -key client.key -out client.csr.`
Creates a new certificate signing request for the client.
- iii. `openssl x509 -req -days 365 -in client.csr -CA
../CA/ca.cert.pem -CAkey ../CA/ca.key -CAcreateserial
-out client_signed.crt.` Creates a certificate for the server signed
by the CA.

2. Starting the server on port 10000: `openssl s_server -accept 10000 -cert git/NSSII/Exercise_2/Server/server_signed.crt -key git/NSSII/Exercise_2/Server/server.key`
3. Connecting the server from the client without verifying the client. `openssl s_client -connect 10.0.0.2:10000 -key git/NSSII/Exercise_2/Client/client.key -pass file:git/NSSII/Exercise_2/CA/mypass.enc`



4. On the server side: `openssl s_server -accept 10000 -cert git/NSSII/Exercise_2/Server/server_signed.crt -key git/NSSII/Exercise_2/Server/server.key -pass file:git/NSSII/Exercise_2/CA/mypass.enc -Verify 2`
On the client side: `openssl s_client -connect 10.0.0.2:10000 -cert git/NSSII/Exercise_2/Client/client_signed.crt -key git/NSSII/Exercise_2/Client/client.key -pass file:git/NSSII/Exercise_2/CA/mypass.enc`
The -Verify option on the server makes it mandatory for the client to submit a certificate, and then server verifies it.

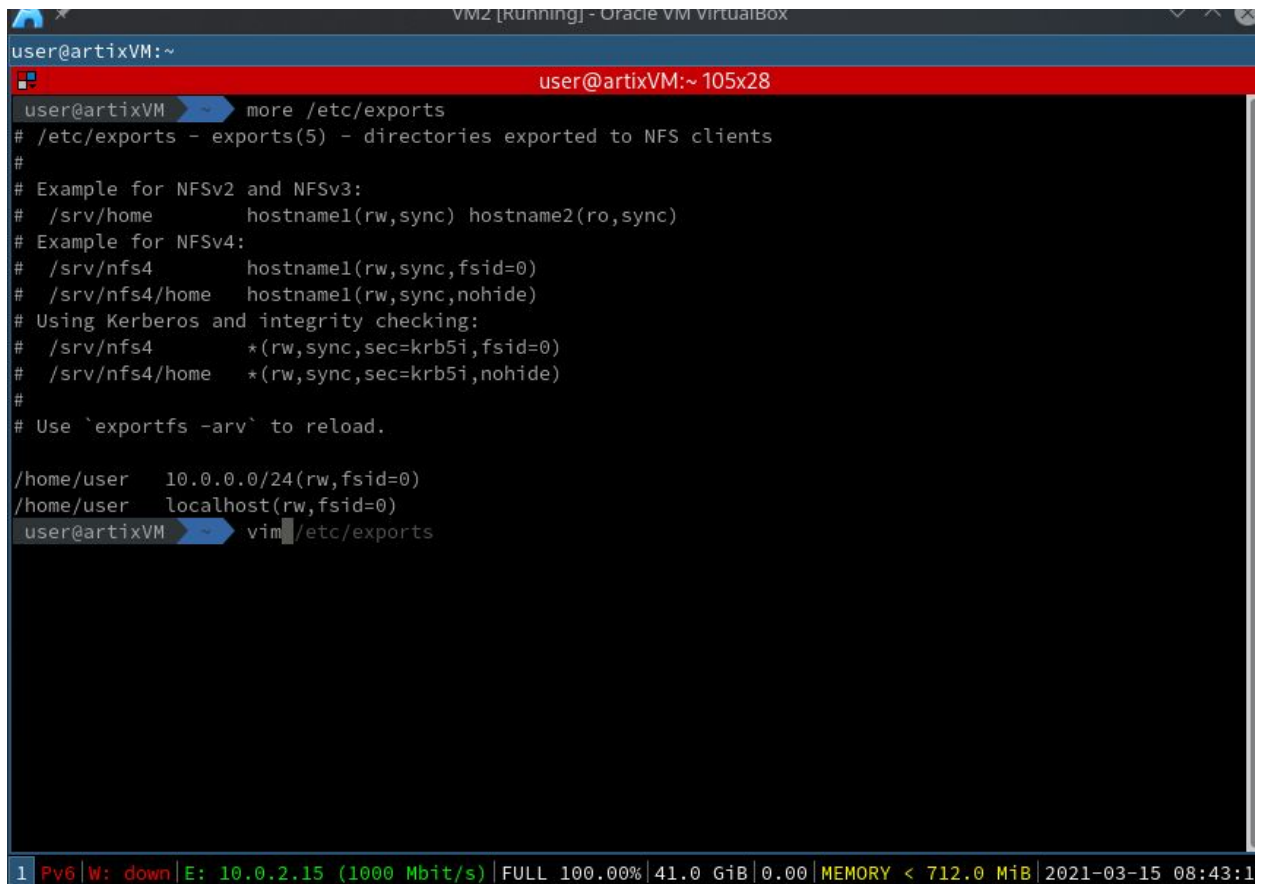




Part 3

1. Export home folder in /etc/exports, make fsid=0 to make /home/user as fake root of the export. The directory is also exported to localhost to be compatible with stunnel in next

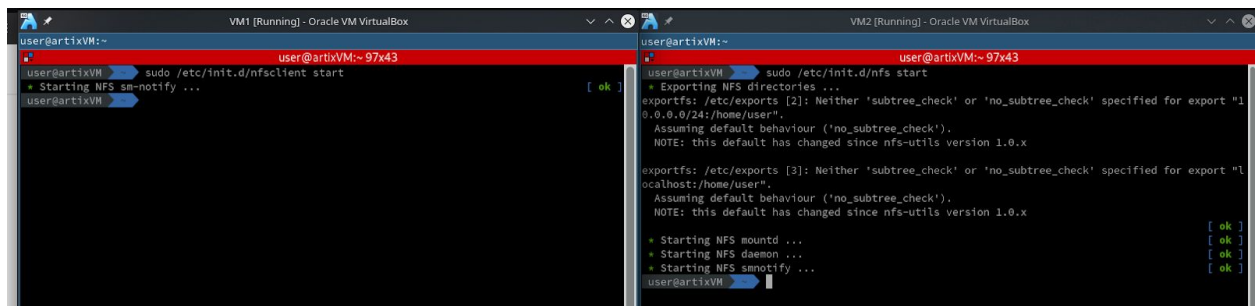
steps.



```
user@artixVM:~  
user@artixVM:~ 105x28  
user@artixVM ~ more /etc/exports  
# /etc/exports - exports(5) - directories exported to NFS clients  
#  
# Example for NFSv2 and NFSv3:  
# /srv/home      hostname1(rw, sync) hostname2(ro, sync)  
# Example for NFSv4:  
# /srv/nfs4      hostname1(rw, sync, fsid=0)  
# /srv/nfs4/home  hostname1(rw, sync, nohide)  
# Using Kerberos and integrity checking:  
# /srv/nfs4      *(rw, sync, sec=krb5i, fsid=0)  
# /srv/nfs4/home  *(rw, sync, sec=krb5i, nohide)  
#  
# Use `exportfs -arv` to reload.  
  
/home/user      10.0.0.0/24(rw, fsid=0)  
/home/user      localhost(rw, fsid=0)  
user@artixVM ~ vim /etc/exports
```

1 Pv6 | W: down | E: 10.0.2.15 (1000 Mbit/s) | FULL 100.00% | 41.0 GiB | 0.00 | MEMORY < 712.0 MiB | 2021-03-15 08:43:1

2. Export this file using `exportfs -arv`.
3. Run the nfs server using `/etc/init.d/nfs start`. On the client use `/etc/init.d/nfsclient` before mounting.



```
user@artixVM:~  
user@artixVM:~ 97x43  
user@artixVM ~ sudo /etc/init.d/nfsclient start  
* Starting NFS sm-notify ... [ ok ]  
user@artixVM ~  
  
user@artixVM:~  
user@artixVM:~ 97x43  
user@artixVM ~ sudo /etc/init.d/nfs start  
* Exporting NFS directories ...  
exportfs: /etc/exports [2]: Neither 'subtree_check' or 'no_subtree_check' specified for export "10.0.0.0/24:/home/user".  
Assuming default behaviour ('no_subtree_check').  
NOTE: this default has changed since nfs-utils version 1.0.x  
  
exportfs: /etc/exports [3]: Neither 'subtree_check' or 'no_subtree_check' specified for export "localhost:/home/user".  
Assuming default behaviour ('no_subtree_check').  
NOTE: this default has changed since nfs-utils version 1.0.x  
  
* Starting NFS mountd ... [ ok ]  
* Starting NFS daemon ... [ ok ]  
* Starting NFS smnotify ... [ ok ]  
user@artixVM ~
```


4. On client use `sudo mount 10.0.0.2:/home/user /mnt` to mount the directory.

```
user@artixVM ~$ sudo /etc/init.d/nfsclient start
* Starting NFS sm-notify ... [ ok ]
user@artixVM ~$ sudo mount 10.0.0.2:/home/user /mnt
user@artixVM ~$ ls
Downloads Exercise2_client keygen server_stunnel.cert stunnel.pem
Exercis_2 git myCA.key start-network testscript.sh
user@artixVM ~$ cd /mnt
user@artixVM /mnt$ ls
Exercise2 half-open.pcapng nssl.pcapng start-network stunnel.pem
git normal-http.pcapng server_stunnel.cert start-network-local
user@artixVM /mnt$
```

5. Configure stunnel on server(/etc/stunnel/stunnel.conf) as follows

```
* Starting NFS smnotify ...
user@artixVM ~$ cat /etc/stunnel/stunnel.conf
; BOM composed of non printable characters. It is here, before the semicolon!
setuid = stunnel
setgid = stunnel
output = /tmp/nfs.log
[nfs]
debug      = 3
verify     = 0
accept     = 12345
connect    = localhost:2049
cert       = /etc/stunnel/stunnel.pem
key        = /etc/stunnel/stunnel.pem
setuid     = stunnel
setgid     = stunnel
user@artixVM ~$
```

stunnel.pem is a file containing a key and a self signed certificate, it acts as a pre shared secret for both client and server. It is generated using command `openssl req -new -x509 -days 365 -nodes -out stunnel.pem -keyout stunnel.pem`. A screenshot from <https://www.stunnel.org/howto.html> to know the parameters.

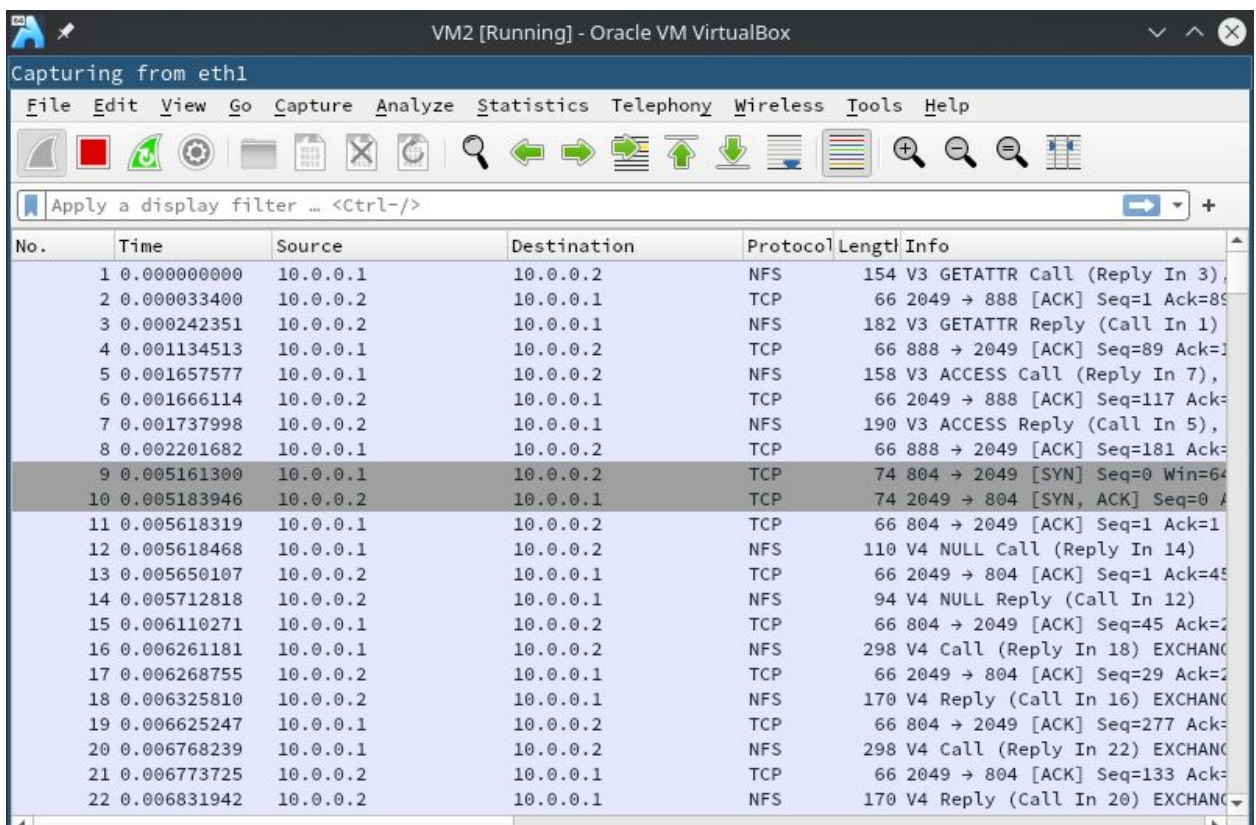
```
-days 365
    make this key valid for 1 year, after which it is not to be used any more
-new
    Generate a new key
-x509
    Generate an X509 certificate (self sign)
-nodes
    Do not put a password on this key.
-config 'stunnel.cnf'
    the OpenSSL configuration file to use
-out 'stunnel.pem'
    where to put the SSL certificate
-keyout 'stunnel.pem'
    put the key in this file
```

6. Configure stunnel on client(/etc/stunnel/stunnel.conf) as follows

```
user@artixVM ~ /mnt$ cat /etc/stunnel/stunnel.conf
; BOM composed of non printable characters. It is here, before the semicolon!
setuid = stunnel
setgid = stunnel
output = /tmp/nfs.log
[nfs]
client      = yes
accept      = localhost:20490
connect     = 10.0.0.2:12345
debug       = 3
cert        = /etc/stunnel/stunnel.pem
key         = /etc/stunnel/stunnel.pem
setuid      = stunnel
setgid      = stunnel
```

Client will interact with its port 20490, stunnel will read the data from that port and make a secure connection with port 12345 on server. Stunnel on server will decrypt the data and send it to port 2049(which is used by nfs server). This connection will work both way round.

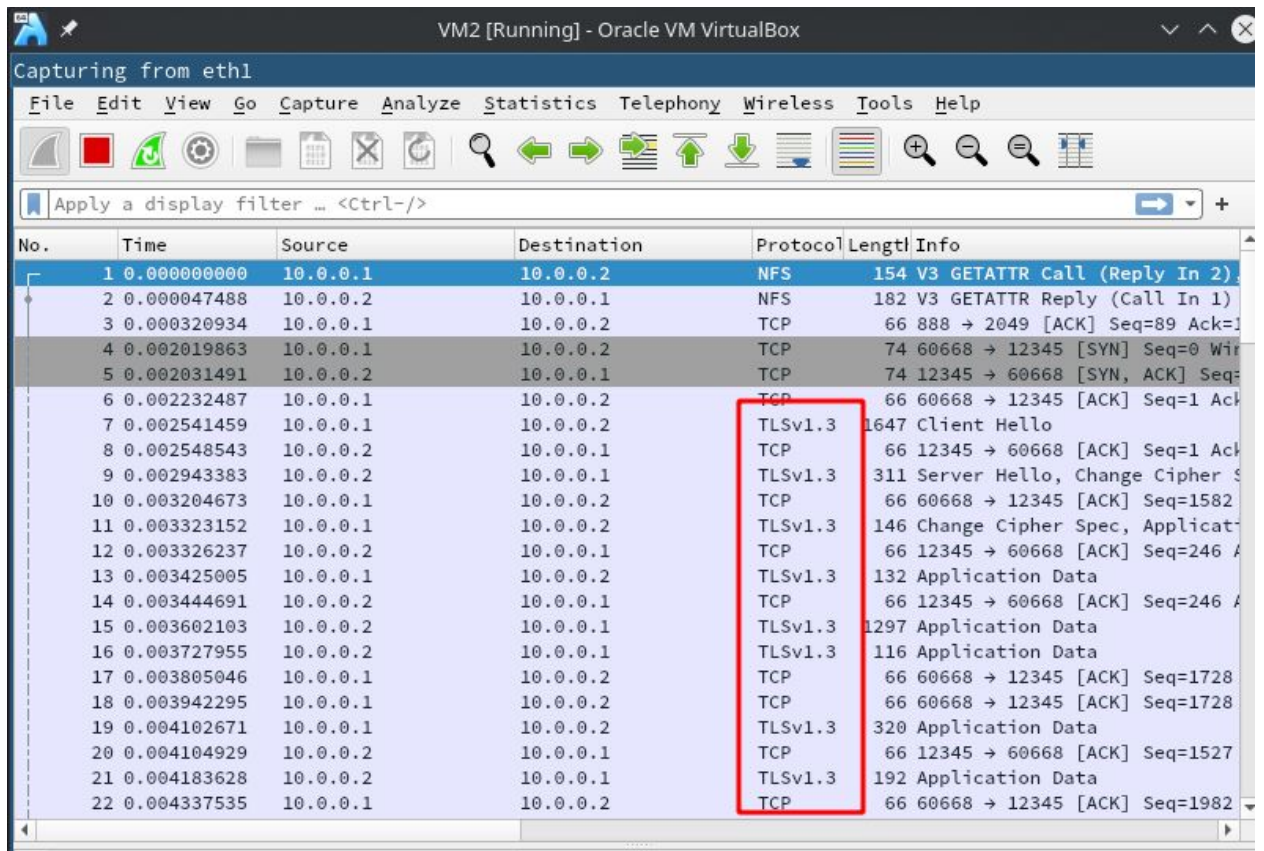
7. Mount the encrypted partition using: `sudo mount -o port=20490 localhost:/home/user /mnt`
8. Unsecure data packet capture will look as follows



The screenshot shows a Wireshark packet capture window titled "VM2 [Running] - Oracle VM VirtualBox". The capture is from the eth1 interface. The packet list table shows the following data:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	10.0.0.2	NFS	154	V3 GETATTR Call (Reply In 3)
2	0.000033400	10.0.0.2	10.0.0.1	TCP	66	2049 → 888 [ACK] Seq=1 Ack=888
3	0.000242351	10.0.0.2	10.0.0.1	NFS	182	V3 GETATTR Reply (Call In 1)
4	0.001134513	10.0.0.1	10.0.0.2	TCP	66	888 → 2049 [ACK] Seq=89 Ack=1
5	0.001657577	10.0.0.1	10.0.0.2	NFS	158	V3 ACCESS Call (Reply In 7)
6	0.001666114	10.0.0.2	10.0.0.1	TCP	66	2049 → 888 [ACK] Seq=117 Ack=
7	0.001737998	10.0.0.2	10.0.0.1	NFS	190	V3 ACCESS Reply (Call In 5)
8	0.002201682	10.0.0.1	10.0.0.2	TCP	66	888 → 2049 [ACK] Seq=181 Ack=
9	0.005161300	10.0.0.1	10.0.0.2	TCP	74	804 → 2049 [SYN] Seq=0 Win=64
10	0.005183946	10.0.0.2	10.0.0.1	TCP	74	2049 → 804 [SYN, ACK] Seq=0 A
11	0.005618319	10.0.0.1	10.0.0.2	TCP	66	804 → 2049 [ACK] Seq=1 Ack=1
12	0.005618468	10.0.0.1	10.0.0.2	NFS	110	V4 NULL Call (Reply In 14)
13	0.005650107	10.0.0.2	10.0.0.1	TCP	66	2049 → 804 [ACK] Seq=1 Ack=45
14	0.005712818	10.0.0.2	10.0.0.1	NFS	94	V4 NULL Reply (Call In 12)
15	0.006110271	10.0.0.1	10.0.0.2	TCP	66	804 → 2049 [ACK] Seq=45 Ack=2
16	0.006261181	10.0.0.1	10.0.0.2	NFS	298	V4 Call (Reply In 18) EXCHANG
17	0.006268755	10.0.0.2	10.0.0.1	TCP	66	2049 → 804 [ACK] Seq=29 Ack=2
18	0.006325810	10.0.0.2	10.0.0.1	NFS	170	V4 Reply (Call In 16) EXCHANG
19	0.006625247	10.0.0.1	10.0.0.2	TCP	66	804 → 2049 [ACK] Seq=277 Ack=
20	0.006768239	10.0.0.1	10.0.0.2	NFS	298	V4 Call (Reply In 22) EXCHANG
21	0.006773725	10.0.0.2	10.0.0.1	TCP	66	2049 → 804 [ACK] Seq=133 Ack=
22	0.006831942	10.0.0.2	10.0.0.1	NFS	170	V4 Reply (Call In 20) EXCHANG

9. TLS secure data packets will look as follows



VM2 [Running] - Oracle VM VirtualBox

Capturing from eth1

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	10.0.0.2	NFS	154	V3 GETATTR Call (Reply In 2)
2	0.000047488	10.0.0.2	10.0.0.1	NFS	182	V3 GETATTR Reply (Call In 1)
3	0.000320934	10.0.0.1	10.0.0.2	TCP	66	888 → 2049 [ACK] Seq=89 Ack=1
4	0.002019863	10.0.0.1	10.0.0.2	TCP	74	60668 → 12345 [SYN] Seq=0 Win
5	0.002031491	10.0.0.2	10.0.0.1	TCP	74	12345 → 60668 [SYN, ACK] Seq=
6	0.002232487	10.0.0.1	10.0.0.2	TCP	66	60668 → 12345 [ACK] Seq=1 Ack
7	0.002541459	10.0.0.1	10.0.0.2	TLSv1.3	1647	Client Hello
8	0.002548543	10.0.0.2	10.0.0.1	TCP	66	12345 → 60668 [ACK] Seq=1 Ack
9	0.002943383	10.0.0.2	10.0.0.1	TLSv1.3	311	Server Hello, Change Cipher S
10	0.003204673	10.0.0.1	10.0.0.2	TCP	66	60668 → 12345 [ACK] Seq=1582
11	0.003323152	10.0.0.1	10.0.0.2	TLSv1.3	146	Change Cipher Spec, Applicat
12	0.003326237	10.0.0.2	10.0.0.1	TCP	66	12345 → 60668 [ACK] Seq=246 A
13	0.003425005	10.0.0.1	10.0.0.2	TLSv1.3	132	Application Data
14	0.003444691	10.0.0.2	10.0.0.1	TCP	66	12345 → 60668 [ACK] Seq=246 A
15	0.003602103	10.0.0.2	10.0.0.1	TLSv1.3	1297	Application Data
16	0.003727955	10.0.0.2	10.0.0.1	TLSv1.3	116	Application Data
17	0.003805046	10.0.0.1	10.0.0.2	TCP	66	60668 → 12345 [ACK] Seq=1728
18	0.003942295	10.0.0.1	10.0.0.2	TCP	66	60668 → 12345 [ACK] Seq=1728
19	0.004102671	10.0.0.1	10.0.0.2	TLSv1.3	320	Application Data
20	0.004104929	10.0.0.2	10.0.0.1	TCP	66	12345 → 60668 [ACK] Seq=1527
21	0.004183628	10.0.0.2	10.0.0.1	TLSv1.3	192	Application Data
22	0.004337535	10.0.0.1	10.0.0.2	TCP	66	60668 → 12345 [ACK] Seq=1982

References

<https://www.thegeekstuff.com/2017/06/brctl-bridge/> brctl command

http://www.microhowto.info/howto/bridge_traffic_between_two_or_more_ethernet_interfaces_on_linux.html using bridge interface

https://ebtables.netfilter.org/examples/basic.html#ex_nat Destination nating

<https://wiki.archlinux.org/index.php/OpenSSL> openssl

<https://deliciousbrains.com/ssl-certificate-authority-for-local-https-development/> set up CA

<https://www.golinuxcloud.com/create-certificate-authority-root-ca-linux/>

[https://wiki.archlinux.org/index.php/User:Grawity/Adding_a_trusted_CA_certificate#System-wide_%E2%80%93_Arch._Fedora_\(p11-kit\)](https://wiki.archlinux.org/index.php/User:Grawity/Adding_a_trusted_CA_certificate#System-wide_%E2%80%93_Arch._Fedora_(p11-kit)) adding certificate

<https://security.stackexchange.com/questions/108508/how-do-i-produce-a-ca-signed-public-key>

<https://www.golinuxcloud.com/openssl-create-client-server-certificate/> signing public keys
certificates with CA key

https://www.openssl.org/docs/man1.0.2/man1/openssl-s_server.html

https://www.openssl.org/docs/man1.0.2/man1/openssl-s_client.html openssl man page

<https://wiki.archlinux.org/index.php/NFS> using nfs server

<https://www.linuxjournal.com/content/encrypting-nfsv4-stunnel-tls> nfs

<https://www.stunnel.org/howto.html> stunnel config