KØBENHAVNS UNIVERSITET

ITS - Assignment 4

Anders Friis Persson, Oliver Meulengracht og Mikkel Willén

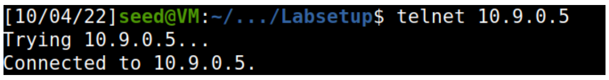
1. november 2022

Indhold

Task 2

Task 2.A

As seen in the picture below we can telnet into the server before running the specified commands.

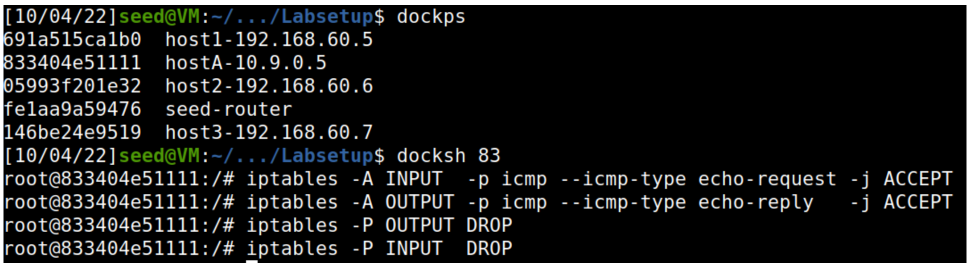


```
[10/04/22]seed@VM:~/.../Labsetup$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
```

Figur 1: Before iptables commands on the container

We type the specified commands and explain their purpose:

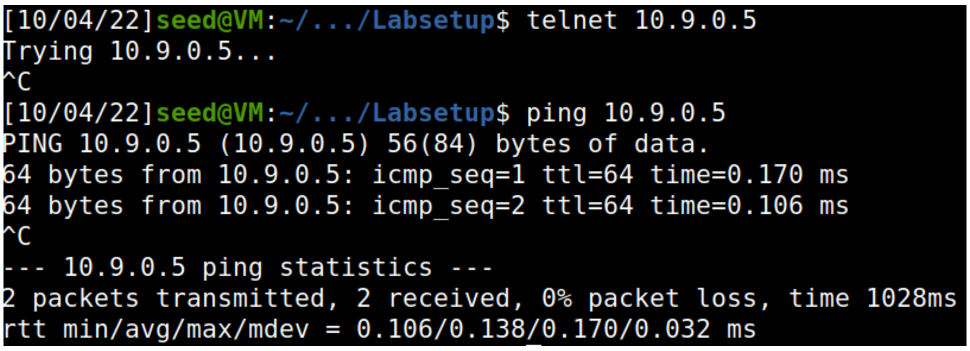
1. The first line accepts all incoming packets that are ping requests.
2. The second line accepts all outgoing packets that are ping replies.
3. The third line is a general purpose rule that drops all outgoing data that do not match the other specified rules.
4. The fourth line is another general purpose rule that drops all incoming data the do not match the other specified rules.



```
[10/04/22]seed@VM:~/.../Labsetup$ dockps
691a515ca1b0  host1-192.168.60.5
833404e51111  hostA-10.9.0.5
05993f201e32  host2-192.168.60.6
fe1aa9a59476  seed-router
146be24e9519  host3-192.168.60.7
[10/04/22]seed@VM:~/.../Labsetup$ docksh 83
root@833404e51111:/# iptables -A INPUT  -p icmp --icmp-type echo-request -j ACCEPT
root@833404e51111:/# iptables -A OUTPUT -p icmp --icmp-type echo-reply   -j ACCEPT
root@833404e51111:/# iptables -P OUTPUT DROP
root@833404e51111:/# iptables -P INPUT  DROP
```

Figur 2: Creating a shell and running iptables commands

And as a result of these rules you should be able to ping the router but not telnet to it, as seen below. This is because we only accepted traffic through pings and used the general purpose rules to block all other forms of traffic.



```
[10/04/22]seed@VM:~/.../Labsetup$ telnet 10.9.0.5
Trying 10.9.0.5...
^C
[10/04/22]seed@VM:~/.../Labsetup$ ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.170 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.106 ms
^C
--- 10.9.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1028ms
rtt min/avg/max/mdev = 0.106/0.138/0.170/0.032 ms
```

Figur 3: After iptables commands on the container

Task 2.B

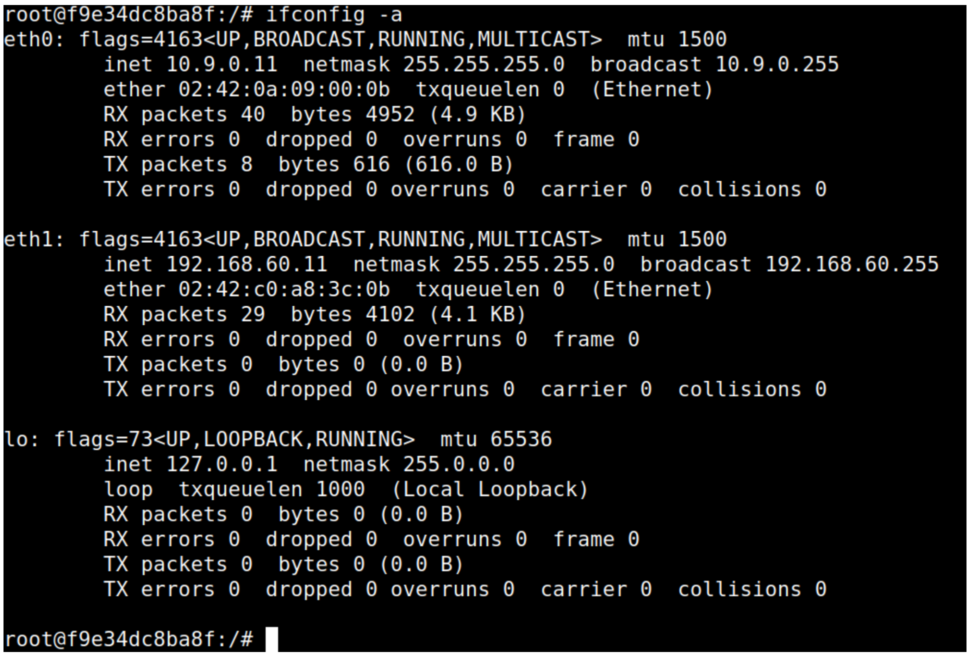
In this task we are to set up the firewall rules 1 through 4.

1. Outside hosts cannot ping internal hosts.
2. Outside hosts can ping the router.
3. Internal hosts can ping outside hosts.
4. All other packets between the internal and external networks should be blocked.

First we found out that you can find the interfaces for the shell by using the command.

```
ifconfig -a
```

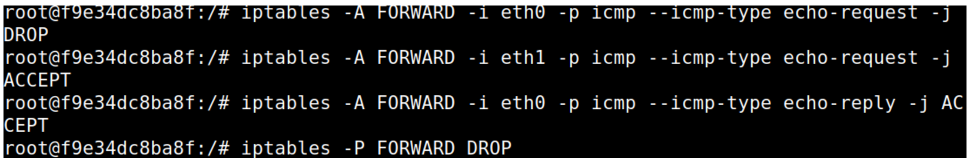
The command above is does the same as `ip addr` but gives a more detailed look at the addresses.



```
root@f9e34dc8ba8f:/# ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.11  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:0b  txqueuelen 0  (Ethernet)
        RX packets 40  bytes 4952 (4.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 616 (616.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.60.11  netmask 255.255.255.0  broadcast 192.168.60.255
        ether 02:42:c0:a8:3c:0b  txqueuelen 0  (Ethernet)
        RX packets 29  bytes 4102 (4.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@f9e34dc8ba8f:/# 
```

Figur 4: Finding the interfaces for the shell

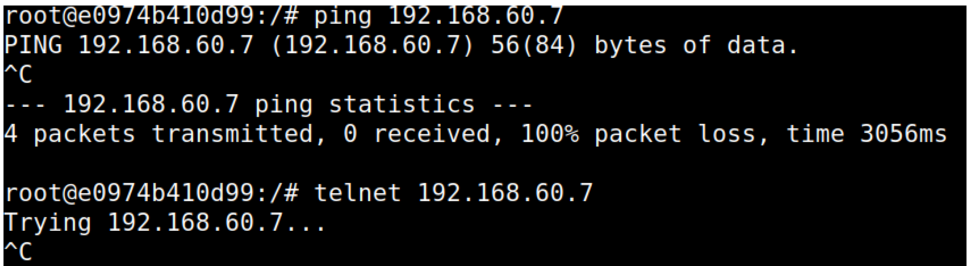
We came up with the following commands to make the firewall that respects the 4 rules.



```
root@f9e34dc8ba8f:/# iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-request -j
DROP
root@f9e34dc8ba8f:/# iptables -A FORWARD -i eth1 -p icmp --icmp-type echo-request -j
ACCEPT
root@f9e34dc8ba8f:/# iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-reply -j AC
CEPT
root@f9e34dc8ba8f:/# iptables -P FORWARD DROP
```

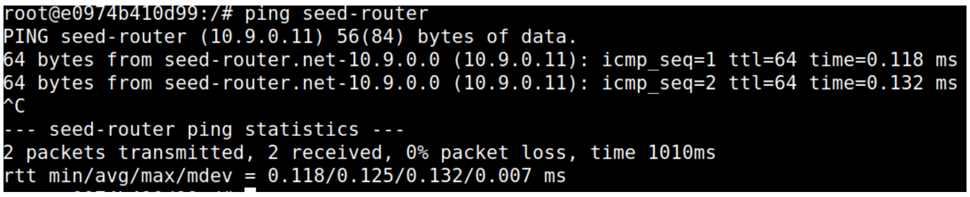
Figur 5: iptables commands run on the seed router

And here you can see that rule 1 and 4 is enforced by trying to ping the internal host3 from the external hostA, and not being able to. And also trying to telnet but this is not possible because all other packets are blocked. Rule 2 is enforced by being able to ping the seed-router from HostA-10.9.0.5



```
root@e0974b410d99:/# ping 192.168.60.7
PING 192.168.60.7 (192.168.60.7) 56(84) bytes of data.
^C
--- 192.168.60.7 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3056ms

root@e0974b410d99:/# telnet 192.168.60.7
Trying 192.168.60.7...
^C
```

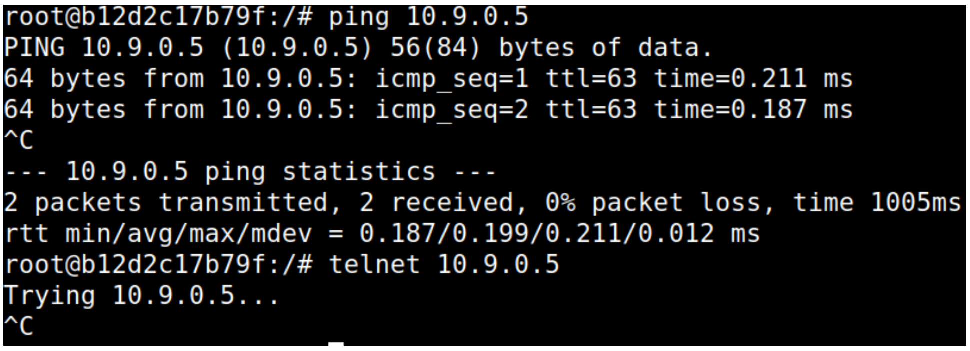
Figur 6: Host A not able to send packages to internal hosts



```
root@e0974b410d99:/# ping seed-router
PING seed-router (10.9.0.11) 56(84) bytes of data.
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=1 ttl=64 time=0.118 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.132 ms
^C
--- seed-router ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.118/0.125/0.132/0.007 ms
```

Figur 7: Host A able to ping the router

We show that rules 3 and 4 are being enforced by showing that the internal host can ping the external HostA, but not telnet to it.



```
root@b12d2c17b79f:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.211 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.187 ms
^C
--- 10.9.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 0.187/0.199/0.211/0.012 ms
root@b12d2c17b79f:/# telnet 10.9.0.5
Trying 10.9.0.5...
^C
```

Figur 8: Host1 able to ping HostA, but not able to telnet to HostA

Task 2.C

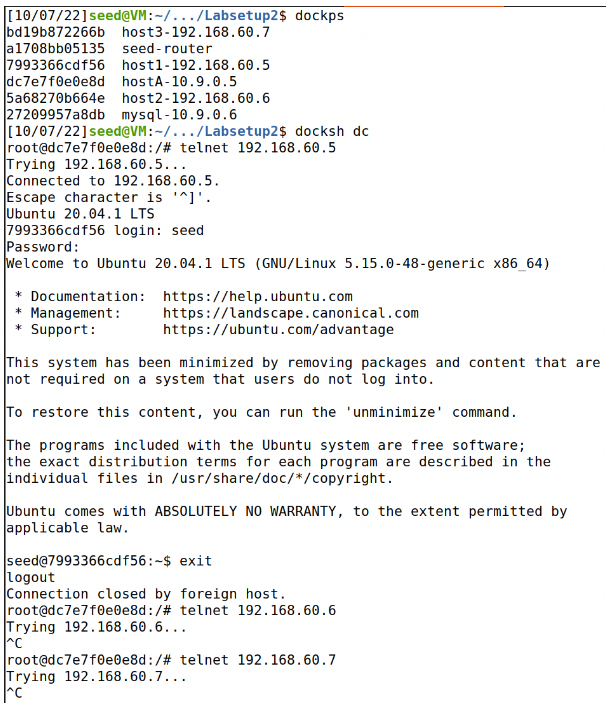
In this task we are asked to achieve the following objectives.

1. All the internal hosts run a telnet server (listening to port 23). Outside hosts can only access the telnet server on 192.168.60.5, not the other internal hosts.
2. Outside hosts cannot access other internal servers.
3. Internal hosts can access all the internal servers
4. Internal hosts cannot access external servers.
5. In this task, the connection tracking mechanism is not allowed. It will be used in a later task.

To achieve these we used the following commands on seed-router

```
$ iptables -A FORWARD -i eth0 -d 192.168.60.5 -p tcp --dport 23 -j ACCEPT
$ iptables -A FORWARD -i eth1 -s 192.168.60.5 -p tcp --sport 23 -j ACCEPT
$ iptables -P FORWARD DROP
```

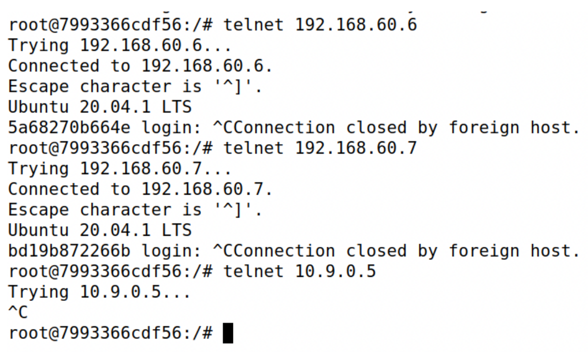
Rule 1 and 2 is shown to be inforced below, since Host A itn't able to connect to Host 2 and Host 3, but is able to connect to Host 1 Below we show, that rules 3 and 4 are enforced, since



```
[10/07/22]seed@VM:~/.../Labsetup2$ dockps
bd19b872266b  host3-192.168.60.7
a1708bb05135  seed-router
7993366cdf56  host1-192.168.60.5
dc7e7f0e0e8d  hostA-10.9.0.5
5a68270b664e  host2-192.168.60.6
27209957a8db  mysql-10.9.0.6
[10/07/22]seed@VM:~/.../Labsetup2$ docksh dc
root@dc7e7f0e0e8d:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
7993366cdf56 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-48-generic x86_64)

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

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

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

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

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

seed@7993366cdf56:~$ exit
logout
Connection closed by foreign host.
root@dc7e7f0e0e8d:/# telnet 192.168.60.6
Trying 192.168.60.6...
^C
root@dc7e7f0e0e8d:/# telnet 192.168.60.7
Trying 192.168.60.7...
^C
```

Figur 9: HostA can't connect to the other internal hosts via telnet

Host 1 is able to connect to Host 2 and 3, but not able to connect to Host A.



```
root@7993366cdf56:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
5a68270b664e login: ^CConnection closed by foreign host.
root@7993366cdf56:/# telnet 192.168.60.7
Trying 192.168.60.7...
Connected to 192.168.60.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
bd19b872266b login: ^CConnection closed by foreign host.
root@7993366cdf56:/# telnet 10.9.0.5
Trying 10.9.0.5...
^C
root@7993366cdf56:/#
```

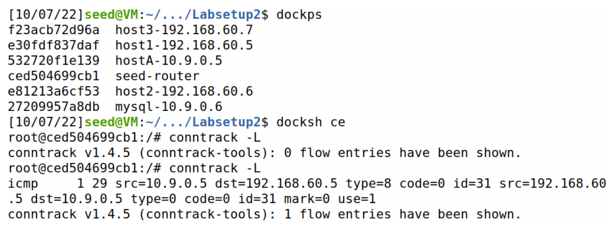
Figur 10: Internal hosts can access the other internal hosts via telnet, and the internal hosts can't access the external HostA

Task 3

Task 3A

We conducted experiements for each of the 3 protocols

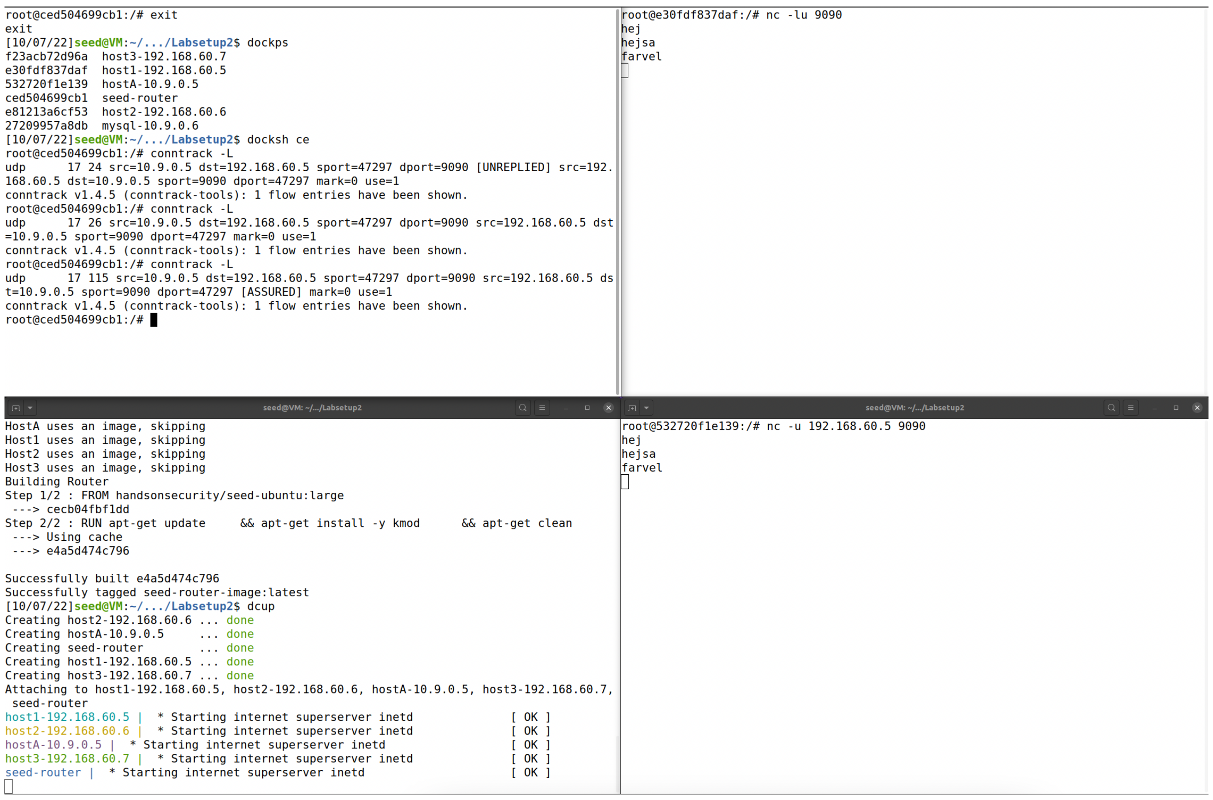
- For ICMP we pinged the internal host1 from hostA and then used conntrack on the seed-router to track the connection. We observe that the ICMP connection is withheld for up to 30 seconds after each ping, the timer is being reset after each ping.



```
[10/07/22]seed@VM:~/.../Labsetup2$ dockps
f23acb72d96a  host3-192.168.60.7
e30fdf837daf  host1-192.168.60.5
532720f1e139  hostA-10.9.0.5
ced504699cb1  seed-router
e81213a6cf53  host2-192.168.60.6
27209957a8db  mysql-10.9.0.6
[10/07/22]seed@VM:~/.../Labsetup2$ docksh ce
root@ced504699cb1:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
root@ced504699cb1:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=31 src=192.168.60
.5 dst=10.9.0.5 type=0 code=0 id=31 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

Figur 11: Lists our active docker containers. Also shows connection state for ICMP which is closed immediately after sending a singular packet

- For UDP opened a netcat UDP server on the internal host1, and then sent packets from the external hostA. We observe that as long as the connection isnt assured (meaning both ends havent replied yet) the connection is withheld for 30 seconds, and then when the connection is ASSURED the connection is held for 120 seconds. (See Figure 12)
- For TCP we opened a netcat server again on the internal Host1 and sent out packets from the external HostA. Now we observe that TCP tracks the connection 432000 seconds or 120 hours. The timer is updated for each reply. With TCP the connections is ASSURED from the start, this is because the TCP protocol utilizes a 3-way handshake to establish a reliable connection. (See Figure 13)



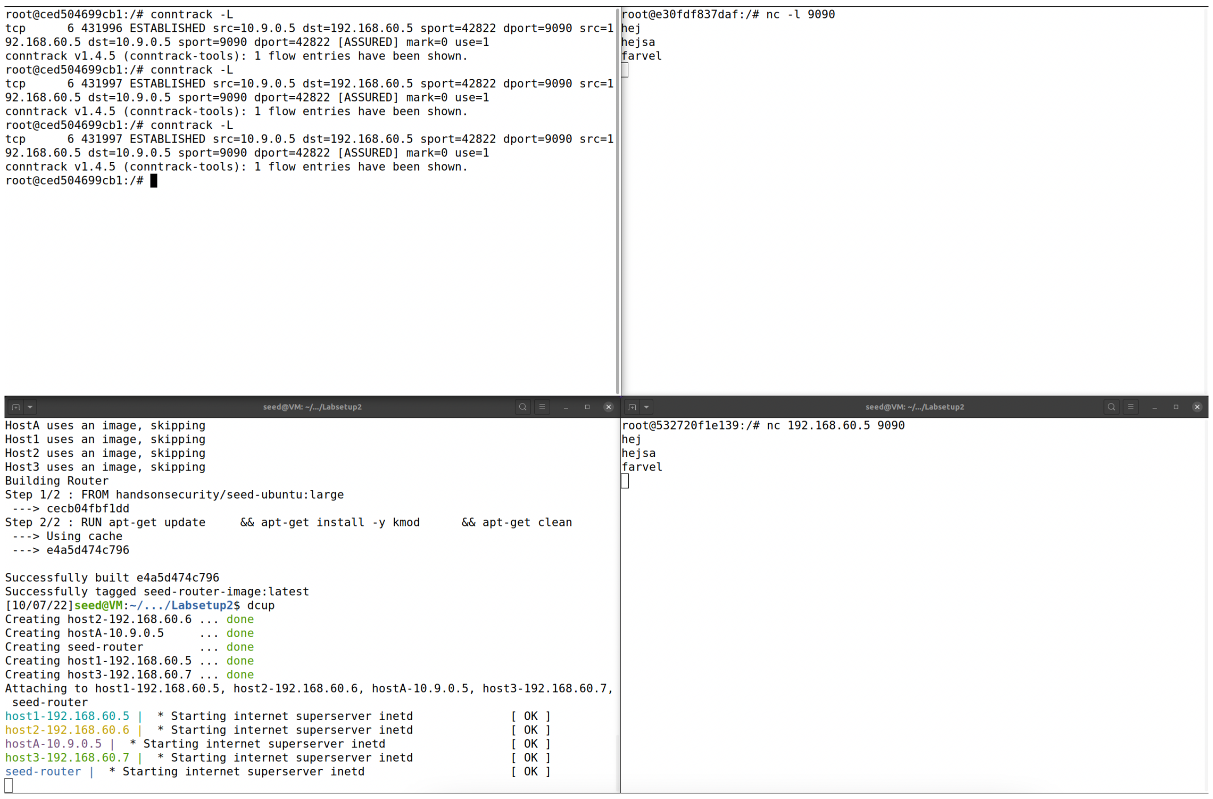
```
root@ced504699cb1:/# exit
exit
[10/07/22]seed@VM:~/.../Labsetup2$ dockps
f23acb72d96a  host3-192.168.60.7
e30fdf837daf  host1-192.168.60.5
532720f1e139  hostA-10.9.0.5
ced504699cb1  seed-router
e81213a6cf53  host2-192.168.60.6
27209957a8db  mysql-10.9.0.6
[10/07/22]seed@VM:~/.../Labsetup2$ docksh ce
root@ced504699cb1:/# conntrack -L
udp      17 24 src=10.9.0.5 dst=192.168.60.5 sport=47297 dport=9090 [UNREPLIED] src=192.
168.60.5 dst=10.9.0.5 sport=9090 dport=47297 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@ced504699cb1:/# conntrack -L
udp      17 26 src=10.9.0.5 dst=192.168.60.5 sport=47297 dport=9090 src=192.168.60.5 dst
=10.9.0.5 sport=9090 dport=47297 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@ced504699cb1:/# conntrack -L
udp      17 115 src=10.9.0.5 dst=192.168.60.5 sport=47297 dport=9090 src=192.168.60.5 ds
t=10.9.0.5 sport=9090 dport=47297 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@ced504699cb1:/#
```

```
root@e30fdf837daf:/# nc -lu 9090
hej
hejsa
farvel
```

```
HostA uses an image, skipping
Host1 uses an image, skipping
Host2 uses an image, skipping
Host3 uses an image, skipping
Building Router
Step 1/2 : FROM handsonsecurity/seed-ubuntu:large
 ---> cecb04fbf1dd
Step 2/2 : RUN apt-get update     && apt-get install -y kmod      && apt-get clean
 ---> Using cache
 ---> e4a5d474c796

Successfully built e4a5d474c796
Successfully tagged seed-router-image:latest
[10/07/22]seed@VM:~/.../Labsetup2$ dcup
Creating host2-192.168.60.6 ... done
Creating hostA-10.9.0.5     ... done
Creating seed-router        ... done
Creating host1-192.168.60.5 ... done
Creating host3-192.168.60.7 ... done
Attaching to host1-192.168.60.5, host2-192.168.60.6, hostA-10.9.0.5, host3-192.168.60.7,
 seed-router
host1-192.168.60.5 |  * Starting internet superserver inetd              [ OK ]
host2-192.168.60.6 |  * Starting internet superserver inetd              [ OK ]
hostA-10.9.0.5 |  * Starting internet superserver inetd                  [ OK ]
host3-192.168.60.7 |  * Starting internet superserver inetd              [ OK ]
seed-router |  * Starting internet superserver inetd                     [ OK ]
```

```
root@532720f1e139:/# nc -u 192.168.60.5 9090
hej
hejsa
farvel
```

Figur 12: Shows connection state for UDP before and after sending and receiving packets



```
root@ced504699cb1:/# conntrack -L
tcp      6 431996 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=42822 dport=9090 src=1
92.168.60.5 dst=10.9.0.5 sport=9090 dport=42822 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@ced504699cb1:/# conntrack -L
tcp      6 431997 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=42822 dport=9090 src=1
92.168.60.5 dst=10.9.0.5 sport=9090 dport=42822 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@ced504699cb1:/# conntrack -L
tcp      6 431997 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=42822 dport=9090 src=1
92.168.60.5 dst=10.9.0.5 sport=9090 dport=42822 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@ced504699cb1:/#
```

```
root@e30fdf837daf:/# nc -l 9090
hej
hejsa
farvel
```

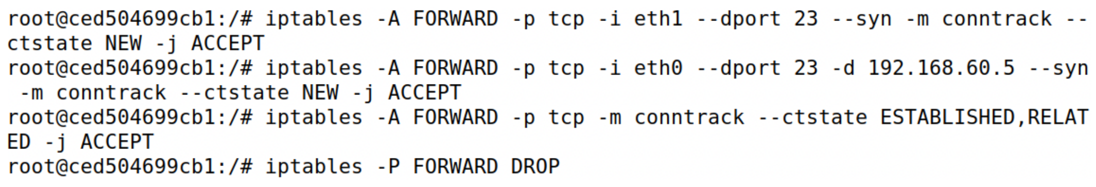
```
HostA uses an image, skipping
Host1 uses an image, skipping
Host2 uses an image, skipping
Host3 uses an image, skipping
Building Router
Step 1/2 : FROM handsonsecurity/seed-ubuntu:large
 ---> cecb04fbf1dd
Step 2/2 : RUN apt-get update     && apt-get install -y kmod      && apt-get clean
 ---> Using cache
 ---> e4a5d474c796

Successfully built e4a5d474c796
Successfully tagged seed-router-image:latest
[10/07/22]seed@VM:~/.../Labsetup2$ dcup
Creating host2-192.168.60.6 ... done
Creating hostA-10.9.0.5     ... done
Creating seed-router        ... done
Creating host1-192.168.60.5 ... done
Creating host3-192.168.60.7 ... done
Attaching to host1-192.168.60.5, host2-192.168.60.6, hostA-10.9.0.5, host3-192.168.60.7,
 seed-router
host1-192.168.60.5 |  * Starting internet superserver inetd              [ OK ]
host2-192.168.60.6 |  * Starting internet superserver inetd              [ OK ]
hostA-10.9.0.5 |  * Starting internet superserver inetd                  [ OK ]
host3-192.168.60.7 |  * Starting internet superserver inetd              [ OK ]
seed-router |  * Starting internet superserver inetd                     [ OK ]
```

```
root@532720f1e139:/# nc 192.168.60.5 9090
hej
hejsa
farvel
```

Figur 13: This connection is ESTABLISHED, which means it is kept open

Task 3B

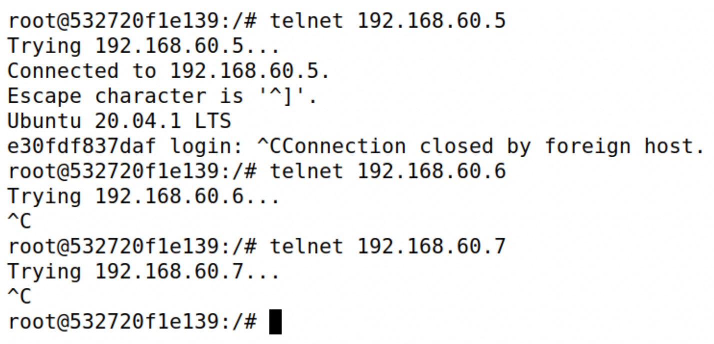
We implemented the firewall from task 2.C with the commands below: It allows connections



```
root@ced504699cb1:/# iptables -A FORWARD -p tcp -i eth1 --dport 23 --syn -m conntrack --
ctstate NEW -j ACCEPT
root@ced504699cb1:/# iptables -A FORWARD -p tcp -i eth0 --dport 23 -d 192.168.60.5 --syn
 -m conntrack --ctstate NEW -j ACCEPT
root@ced504699cb1:/# iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISHED,RELAT
ED -j ACCEPT
root@ced504699cb1:/# iptables -P FORWARD DROP
```

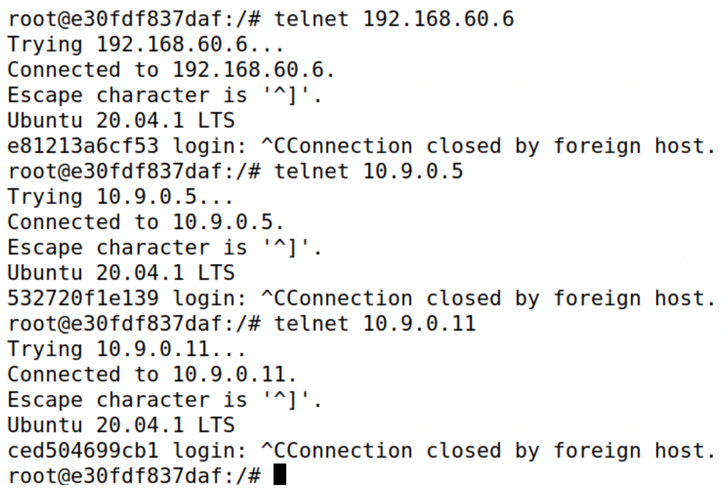
Figur 14: The commands used for solving 3B

through port 23 from the internal network or through port 23 to the specific address 192.168.60.5 from the external network in the first two rules typed above. Rule 3 permits all traffic through established or related connections, while Rule 4 prohibits all other traffic through the firewall.



```
root@532720f1e139:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
e30fdf837daf login: ^CConnection closed by foreign host.
root@532720f1e139:/# telnet 192.168.60.6
Trying 192.168.60.6...
^C
root@532720f1e139:/# telnet 192.168.60.7
Trying 192.168.60.7...
^C
root@532720f1e139:/# █
```

Figur 15: External hosts can only access the telnet server on 192.168.60.5



```
root@e30fdf837daf:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
e81213a6cf53 login: ^CConnection closed by foreign host.
root@e30fdf837daf:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
532720f1e139 login: ^CConnection closed by foreign host.
root@e30fdf837daf:/# telnet 10.9.0.11
Trying 10.9.0.11...
Connected to 10.9.0.11.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
ced504699cb1 login: ^CConnection closed by foreign host.
root@e30fdf837daf:/# █
```

Figur 16: Internal hosts can access other internal hosts as well as external hosts.

We would use the same commands as in 2C, but also add another command, which allows outgoing connections from our internal network. An advantages with the stateful firewall, is that packages are being assessed together, and that later packages can be accepted, if they are part of a previous block of packages.

Short Question - 1

A private network is one that is only accessible to trusted users. It is based on a network architecture that provides physical isolation. This could include anything from local area networks to private home networks. A VPN, on the other hand, is a private network that connects physically distant users or sub-networks. Not only is it physically isolated, but it is also protected by encrypted tunnels and protocols, software, and hardware. The term "virtual"refers to the use of shared public links, whereas private networks used expensive network links to physically connect remote networks. To summarize briefly, a VPN is essentially: A VPN stands for virtual private network. It's a service that safeguards your internet connection and privacy. The two main use cases are 'site-to-site VPNs', which bridge private networks across a public channel. The final type of VPN is a'remote access VPN', which allows authorized clients to connect to a private network from their home.

Short Question - 2

Yes, a host firewall can block malware from communicating from an infected computer back to its controller on the internet. An example of this could be that the user's firewall is blocking a wide range of IP addresses, and the malware's controller is among those IP addresses. You could also in theory block the communication if you were aware of the communication port in which the malware communicated with it's host.

Short Question - 3

We need to know what IPsec is in order to determine whether it is secure or not. IPsec is a protocol that allows for secure communication by providing network-layer security services that are inherited by all transport and application layer protocols. IPsec enables VPNs by delivering a suite of security services via three protocols. IKE is used for key management, AH is used for authentication, and ESP is used for encryption and additional authentication.

A buffer is only a temporary storage space, a program may forget the buffer location and thus overwrite adjacent memory locations. This is therefore only possible if there is a buffer overflow in the code area. In theory, the attacker could send UDP packets to the vulnerable system/router, perform remote code execution, and gain control of the system.

Short Question - 4

IPsec has two modes of operation: tunnel mode and transport mode. To provide an end-to-end VPN from one host to another, IPsec transport mode is used. The IPsec header is inserted between the original IP header and the original IP payload in transport mode. When used with ESP, the original IP payload is encrypted as the 'IPsec payload' and is transported here rather than tunneled.

However, IPsec has two VPN use cases in tunnel mode: network-to-network VPNs and host-to-network VPNs. The VPN terminates at security gateways to an enterprise network on each side in the first use-case. In terms of AH or ESP protection, each security gateway is an endpoint. The packets are unprotected for the rest of their journey through the enterprise network. As a result, end-to-end security is not provided. After the outer IP header and IPsec header are consumed by the gateway, the inner packet is delivered to its destination using the remaining IP header. The VPN functionality built into the remote host software serves as an in-host network gateway in the second use-case.