Iptables

HW8 - CNS Sapienza

Valerio Coretti 1635747

2020-01-02

1 Introduction

Today every computer is connected to the internet and continuously exchanges data with other hosts. Data from the outside world can be *good* data, such as a page of a site that we have requested, a friend who contacts us in chat or a file that we are downloading with a filesharing program, and *bad* data, such as a cracker that contacts us in various ways and without our knowledge to try to get into our computer.

The job of the firewall is to recognize good traffic and to block bad traffic. To do this it needs our help, with rules for carrying out checks on data.

The term *Iptables* is used to define the Linux kernel firewall, part of the Netfilter project and the userspace tool used to configure this firewall. The Netfilter framework provides a set of features used by iptables to perform operations on packets.

Iptables groups all the controls on incoming traffic, in the Chain INPUT, and on outgoing traffic, in the Chain OUTPUT. The Chain FORWARD is used for example when the data traffic is not addressed to us but still passes through our computer. Each of these chains has a policy, that is, a predefined action to be performed when all the other checks in the chain have failed to recognize whether the data was good or not.

In this homework, we make some experiments with the iptables firewall and to do this we simulate a network connection between two virtual machines with some rules specificated.

2 Setting Up

In this section we introduce all the technology we will use to try experiments with Iptables. First of all we use two *Docker* virtual machines with *Ubuntu* as Linux distrubution and we setup Iptables:

Download and run Ubuntu VMs:

```
docker pull ubuntu
docker run --cap-add=NET_ADMIN -it ubuntu
```

Setup sudo and Iptables:

```
apt update -y
apt-get install iptables sudo -y
```

Setup a user, user1, and added it to the sudo group:

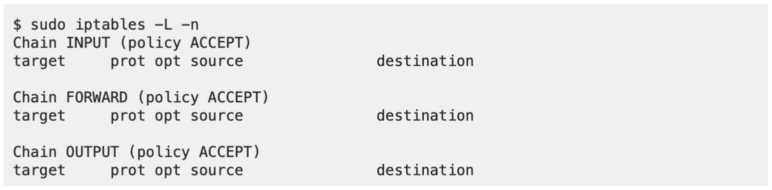
```
adduser user1
adduser user1 sudo
```

Set user to user1:

```
su - user1
```

Check if user1 can access iptables via sudo:

```
sudo iptables -L -n
```



```
$ sudo iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

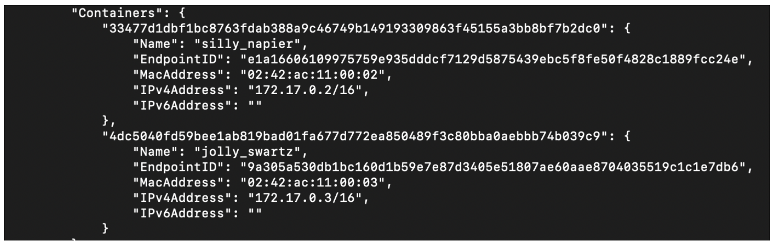
Now we setup the main network tool:

```
sudo apt-get install netcat telnet ftp ping ssh
```

All we need now is the configuration of the network and with docker, this is very easy:

```
docker inspect network bridge
```

The following is only the output part relative to the containers:



```
        "Containers": {
            "33477d1dbf1bc8763fdab388a9c46749b149193309863f45155a3bb8bf7b2dc0": {
                "Name": "silly_napier",
                "EndpointID": "e1a16606109975759e935dddcf7129d5875439ebc5f8fe50f4828c1889fcc24e",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            },
            "4dc5040fd59bee1ab819bad01fa677d772ea850489f3c80bba0aebbb74b039c9": {
                "Name": "jolly_swartz",
                "EndpointID": "9a305a530db1bc160d1b59e7e87d3405e51807ae60aae8704035519c1c1e7db6",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            }
```

2.1 NGINX Mini Web Server

If we want to establish a connection between two machines, for example with telnet, we need a web server in one of the two. For this scope, we use the Nginx service. There is a very fast guide to creating a web server with a simple Html home page on our VM here:
`https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04`

At the end of the configuration we have only to run this command:

```
sudo service nginx start
```

3 Experimentation

In this section, we will try an experiment where, given a sequence of commands *seq*, we have to incrementally allow, with proper rules, a command $c \in seq$. Therefore the first thing we do is to block all the traffic in the network:

Flush the rules out of the chains:

```
sudo iptables -F
```

Change the policy of the chains:

```
sudo iptables -P INPUT DROP
sudo iptables -P OUTPUT DROP
sudo iptables -P FORWARD DROP
```

Now we can start our test. First of all we create a file *"seq.sh"* with a sequence of commands that at the beginning will be all blocked:

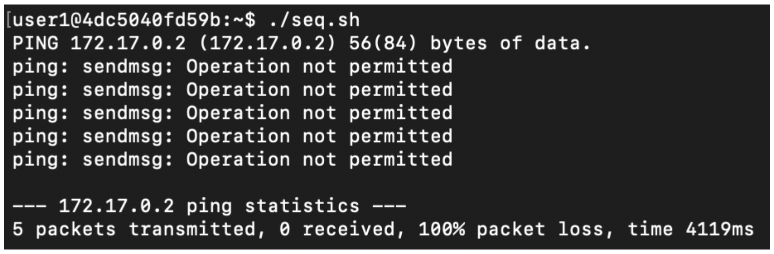
```
ping -c 5 [ip]
telnet [ip] [port]
ssh [ip]
ftp [ip]
netcat [ip] [port]
```

3.1 ICMP protocol

The first test is on the icmp protocol, therefore we run the command:

```
ping -c 5 [ip]
```

this commands try to exachange five packets with icmp. We will receive this output:



```
[user1@4dc5040fd59b:~$ ./seq.sh
PING 172.17.0.2 (172.17.0.2) 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

--- 172.17.0.2 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4119ms
```

In fact at this moment we have all the machines blocked in Input and Output. With the following rules we allow the traffic in both sender and receiver:

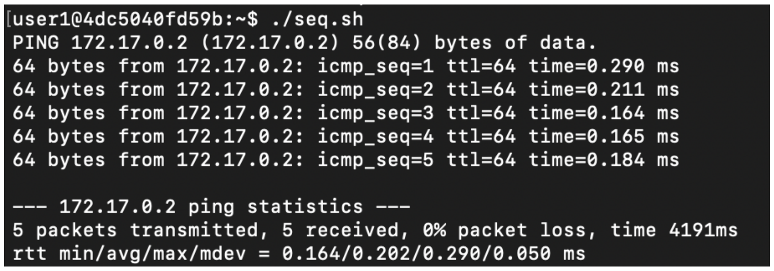
Accept ICMP request and reply from a specific host:

```
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p icmp
--icmp-type echo-request -j ACCEPT
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p icmp
--icmp-type echo-reply -j ACCEPT
```

Allow to send ICMP request and reply to a specific host:

```
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p icmp
--icmp-type echo-request -j ACCEPT
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p icmp
--icmp-type echo-reply -j ACCEPT
```

this rules must be setted also in the other virtual machine to allow a bidirectional channel for ICMP protocol. Now our two virtual machines can exchange only the icmp messagge infact the current output will be:



```
[user1@4dc5040fd59b:~$ ./seq.sh
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.290 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.211 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.164 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.165 ms
64 bytes from 172.17.0.2: icmp_seq=5 ttl=64 time=0.184 ms

--- 172.17.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4191ms
rtt min/avg/max/mdev = 0.164/0.202/0.290/0.050 ms
```

3.2 Telnet connection

The second command of the seq is a telnet connection, the running command is:

```
telnet [ip] [port]
```

The current output of the sequence is nothing. The reason is that machine A tries to establish a Telnet connection to B, but it has all the traffic in INPUT and OUTPUT blocked except for ICMP protocol. Therefore we set the rules to allow telnet connection in both direction, remembering that telnet use port 23, but the response of Nginx web server run in port 80:

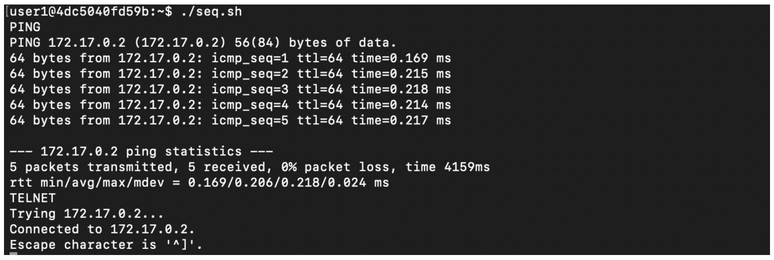
Accept replies from a specific Nginx web server:

```
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p tcp
--sport 80 -j ACCEPT
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p tcp
--dport 23 -j ACCEPT
```

Allow Telnet to establish a connection with a specific web server:

```
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p tcp
--dport 80 -j ACCEPT
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p tcp
--sport 23 -j ACCEPT
```

this rules must be setted also in the other virtual machine to allow a bidirectional channel. Now if everything went well we have the following output:



```
user1@4dc5040fd59b:~$ ./seq.sh
PING
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.169 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.215 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.218 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.214 ms
64 bytes from 172.17.0.2: icmp_seq=5 ttl=64 time=0.217 ms

--- 172.17.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4159ms
rtt min/avg/max/mdev = 0.169/0.206/0.218/0.024 ms
TELNET
Trying 172.17.0.2...
Connected to 172.17.0.2.
Escape character is '^]'.
```

3.3 SSH connection

Now we are ready to establish a SSH connection. This step is very simple, in fact we have to run the following command:

```
ssh [ip]
```

As we expect the output is nothing because the sender cannot establish this type of connection. SSH run in port 22 and therefore we go to allow it with the following iptables rules:

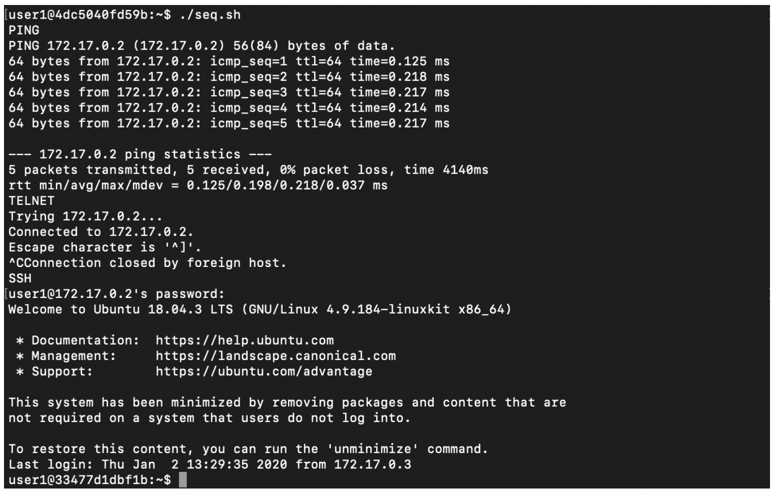
Accept replies from a specific SSH server:

```
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p tcp
--dport 22 -j ACCEPT
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p tcp
--sport 22 -j ACCEPT
```

Allow an ip to establish a SSH connection:

```
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p tcp
--dport 22 -j ACCEPT
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p tcp
--sport 22 -j ACCEPT
```

for a bidirectional channel we set this rules also in the other host.
Now if everything went well we have the following output:



```
user1@4dc5040fd59b:~$ ./seq.sh
PING
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.125 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.218 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.217 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.214 ms
64 bytes from 172.17.0.2: icmp_seq=5 ttl=64 time=0.217 ms

--- 172.17.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4140ms
rtt min/avg/max/mdev = 0.125/0.198/0.218/0.037 ms
TELNET
Trying 172.17.0.2...
Connected to 172.17.0.2.
Escape character is '^]'.
^CConnection closed by foreign host.
SSH
user1@172.17.0.2's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.9.184-linuxkit x86_64)

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

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

To restore this content, you can run the 'unminimize' command.
Last login: Thu Jan  2 13:29:35 2020 from 172.17.0.3
user1@33477d1dbf1b:~$
```

3.4 FTP connection

First of all we set a *FTP Server* in one of the two machine in this way:

```
sudo apt-get install vsftpd
sudo service vsftpd start
```

Now we can set the rules to establish an ftp connection between the two host. We have to remember that FTP run on two port 21 (for control) and 20 (for data):

Accept replies from a specific FTP server:

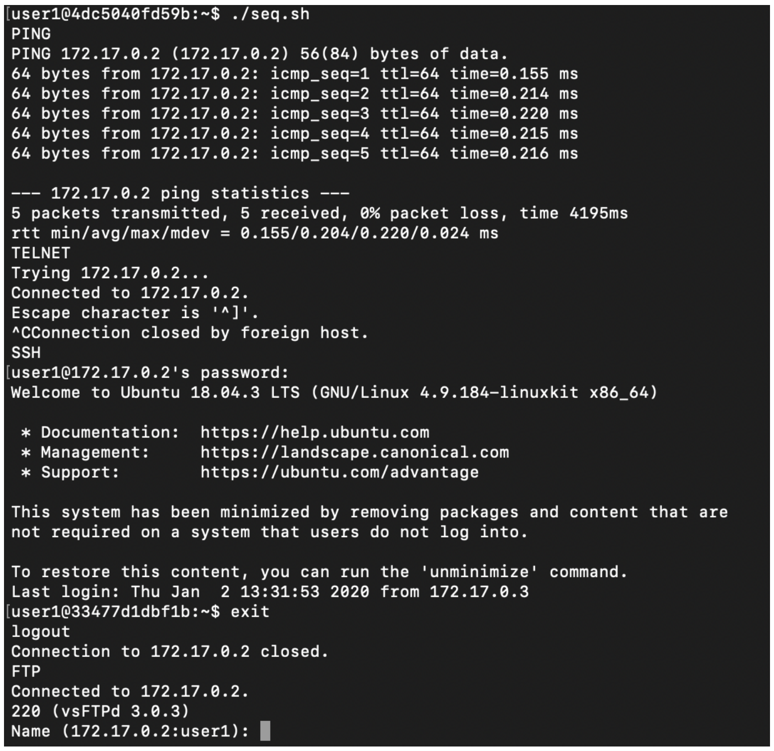
```
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p tcp
--dport 21 -j ACCEPT
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p tcp
--sport 21 -j ACCEPT
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p tcp
--dport 20 -j ACCEPT
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p tcp
--sport 20 -j ACCEPT
```

Allow FTP to establish a connection with a specific ftp server:

```
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p tcp
--dport 21 -j ACCEPT
```

```
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p tcp
--sport 21 -j ACCEPT
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p tcp
--dport 20 -j ACCEPT
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p tcp
--sport 20 -j ACCEPT
```

this rules must be setted also in the other virtual machine to allow a bidirectional channel. Now if everything went well we have the following output:



```
user1@4dc5040fd59b:~$ ./seq.sh
PING
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.155 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.214 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.220 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.215 ms
64 bytes from 172.17.0.2: icmp_seq=5 ttl=64 time=0.216 ms

--- 172.17.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4195ms
rtt min/avg/max/mdev = 0.155/0.204/0.220/0.024 ms
TELNET
Trying 172.17.0.2...
Connected to 172.17.0.2.
Escape character is '^]'.
^CConnection closed by foreign host.
SSH
user1@172.17.0.2's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.9.184-linuxkit x86_64)

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

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

To restore this content, you can run the 'unminimize' command.
Last login: Thu Jan  2 13:31:53 2020 from 172.17.0.3
user1@33477d1dbf1b:~$ exit
logout
Connection to 172.17.0.2 closed.
FTP
Connected to 172.17.0.2.
220 (vsFTPd 3.0.3)
Name (172.17.0.2:user1):
```

3.5 NETCAT connection

In this section, we use a tool called Netcat to establish a TCP connection on a port chosen by us. First of all in one of the two machines we have to wait, on a chosen port, for a connection (like a server):

```
netcat -l -p [chosen port]
```

In the other side, we run

```
netcat [ip] [chosen port]
```

to establish the connection, but as we know before we have to set the rules on iptables firewall:

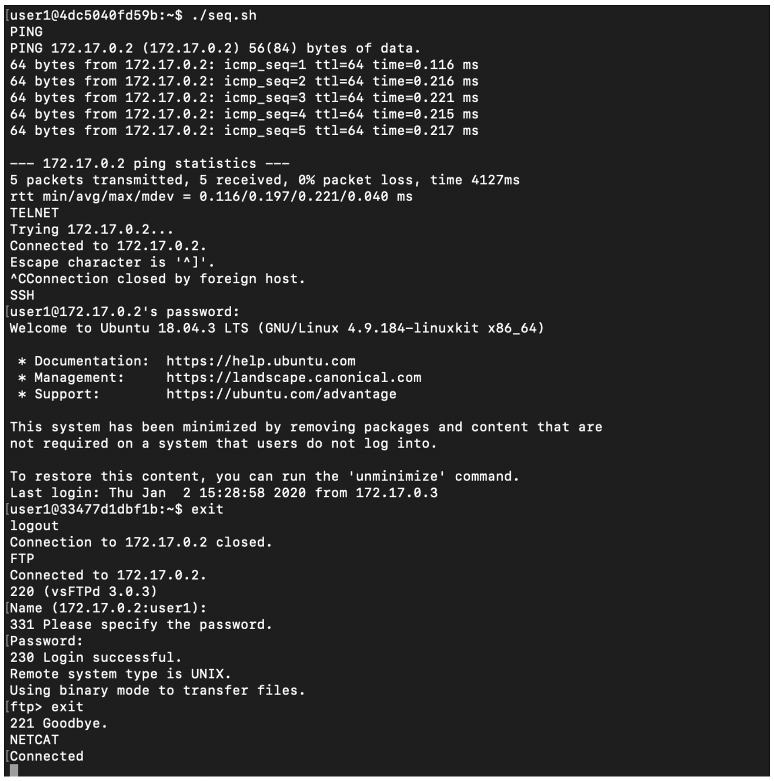
Accept replies from a specific Netcat server:

```
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p tcp
--dport [chosen port] -j ACCEPT
sudo iptables -A INPUT -s [SourceIP] -d [DestIP] -p tcp
--sport [chosen port] -j ACCEPT
```

Allow an ip to establish a connection with a specific Netcat server: `sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p tcp --dport [chosen port] -j ACCEPT`

```
sudo iptables -A OUTPUT -s [SourceIP] -d [DestIP] -p tcp
--sport [chosen port] -j ACCEPT
```

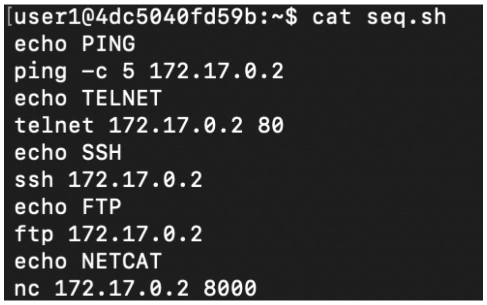
this rules must be setted also in the other virtual machine to allow a bidirectional channel. Now if everything went well we have the following output:



```
[user1@4dc5040fd59b:~$ ./seq.sh
PING
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.116 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.216 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.221 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.215 ms
64 bytes from 172.17.0.2: icmp_seq=5 ttl=64 time=0.217 ms

--- 172.17.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4127ms
rtt min/avg/max/mdev = 0.116/0.197/0.221/0.040 ms
TELNET
Trying 172.17.0.2...
Connected to 172.17.0.2.
Escape character is '^]'.
^CConnection closed by foreign host.
SSH
[user1@172.17.0.2's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.9.184-linuxkit x86_64)

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

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

To restore this content, you can run the 'unminimize' command.
Last login: Thu Jan  2 15:28:58 2020 from 172.17.0.3
[user1@33477d1dbf1b:~$ exit
logout
Connection to 172.17.0.2 closed.
FTP
Connected to 172.17.0.2.
220 (vsFTPd 3.0.3)
[Name (172.17.0.2:user1):
331 Please specify the password.
[Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
[ftp> exit
221 Goodbye.
NETCAT
[Connected
```

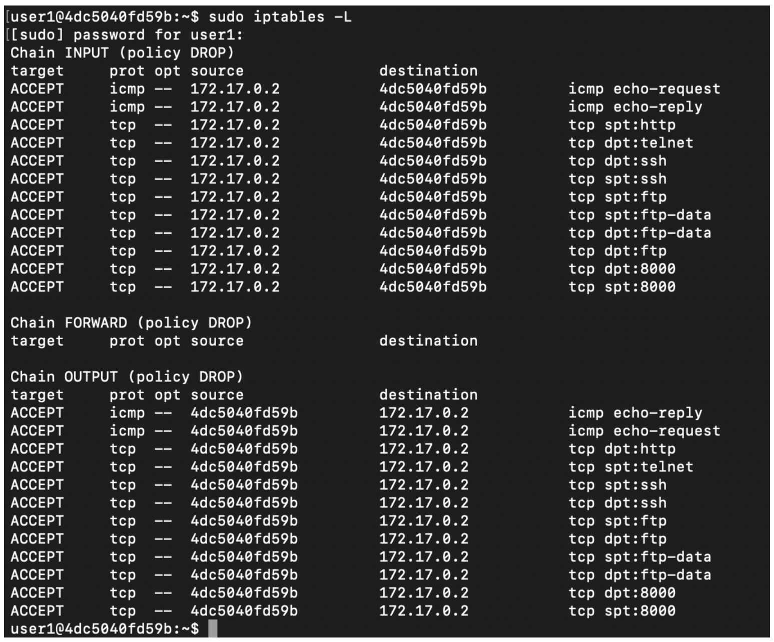
4 Conclusion

We stop here our experiment with iptables and finally we show our *seq.sh* file that can be run with the command: `./seq.sh`



```
[user1@4dc5040fd59b:~$ cat seq.sh
echo PING
ping -c 5 172.17.0.2
echo TELNET
telnet 172.17.0.2 80
echo SSH
ssh 172.17.0.2
echo FTP
ftp 172.17.0.2
echo NETCAT
nc 172.17.0.2 8000
```

and moreover, the following picture shows our final configuration of iptables firewall. Obviously, we can add more rules relative to the protocol we want to use.



```
[user1@4dc5040fd59b:~$ sudo iptables -L
[[sudo] password for user1:
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  172.17.0.2           4dc5040fd59b         icmp echo-request
ACCEPT     icmp --  172.17.0.2           4dc5040fd59b         icmp echo-reply
ACCEPT     tcp  --  172.17.0.2           4dc5040fd59b         tcp spt:http
ACCEPT     tcp  --  172.17.0.2           4dc5040fd59b         tcp dpt:telnet
ACCEPT     tcp  --  172.17.0.2           4dc5040fd59b         tcp dpt:ssh
ACCEPT     tcp  --  172.17.0.2           4dc5040fd59b         tcp spt:ssh
ACCEPT     tcp  --  172.17.0.2           4dc5040fd59b         tcp spt:ftp
ACCEPT     tcp  --  172.17.0.2           4dc5040fd59b         tcp spt:ftp-data
ACCEPT     tcp  --  172.17.0.2           4dc5040fd59b         tcp dpt:ftp-data
ACCEPT     tcp  --  172.17.0.2           4dc5040fd59b         tcp dpt:ftp
ACCEPT     tcp  --  172.17.0.2           4dc5040fd59b         tcp dpt:8000
ACCEPT     tcp  --  172.17.0.2           4dc5040fd59b         tcp spt:8000

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  4dc5040fd59b         172.17.0.2           icmp echo-reply
ACCEPT     icmp --  4dc5040fd59b         172.17.0.2           icmp echo-request
ACCEPT     tcp  --  4dc5040fd59b         172.17.0.2           tcp dpt:http
ACCEPT     tcp  --  4dc5040fd59b         172.17.0.2           tcp spt:telnet
ACCEPT     tcp  --  4dc5040fd59b         172.17.0.2           tcp spt:ssh
ACCEPT     tcp  --  4dc5040fd59b         172.17.0.2           tcp dpt:ssh
ACCEPT     tcp  --  4dc5040fd59b         172.17.0.2           tcp spt:ftp
ACCEPT     tcp  --  4dc5040fd59b         172.17.0.2           tcp dpt:ftp
ACCEPT     tcp  --  4dc5040fd59b         172.17.0.2           tcp spt:ftp-data
ACCEPT     tcp  --  4dc5040fd59b         172.17.0.2           tcp dpt:ftp-data
ACCEPT     tcp  --  4dc5040fd59b         172.17.0.2           tcp dpt:8000
ACCEPT     tcp  --  4dc5040fd59b         172.17.0.2           tcp spt:8000
user1@4dc5040fd59b:~$
```

References

[1] *Iptables*
`https://wiki.archlinux.org/index.php/Iptables`

[2] *Ping and ICMP messages*
`http://wwwcdf.pd.infn.it/AppuntiLinux/messaggi_icmp.htm`

[3] *How to use SSH*
`https://phoenixnap.com/kb/ssh-to-connect-to-remote-server-linux-or-windows`

[4] *Iptable HOWTO*
`https://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.txt`

[5] *Server Ftp*
`https://wiki.ubuntu-it.org/Server/Ftp`