

Packet Filter Firewalls

Linus Bein Fahlander (linusfa@kth.se)

A public repository containing the files generated and used in this lab can be found on [GitHub](#).

Task 1 - Building a Firewall

1.2 Network Permission

UFW configuration after completing this sub task

```
Status: active
Logging: on (low)
Default: deny (incoming), deny (outgoing), deny (routed)
New profiles: skip

To Action From
--
Anywhere on eth1 ALLOW IN 10.0.20.0/24

Anywhere ALLOW OUT 10.0.20.0/24 on eth1
```

Q 1

`ufw` comes with an easy way of setting default policies, the ones needed for the desired configuration are:

```
ufw default deny incoming
ufw default deny outgoing
ufw default deny routed
```

To allow communication within the internal network I set these rules:

```
ufw allow in on eth1 from 10.0.20.0/24
ufw allow out on eth1 from 10.0.20.0/24
```

Q 2

The difference between `deny` and `reject` is that packets that are not allowed will be answered with a reject if you use `reject` where with `deny` the packet will simply be dropped.

The one you choose depends on what you want to achieve with the firewall.

Is the interface only internal facing maybe you want to reject so that developers will get a clearer message that the request was rejected and that the interface doesn't just not answer.

However it is a good idea in most cases to just drop the packet to not give the possible attacker any information and not to waste computing cycles creating the reject message.

1.3 Permitting a Service

UFW configuration after completing this sub task

```
Status: active
Logging: on (low)
Default: deny (incoming), deny (outgoing), deny (routed)
New profiles: skip

To Action From
--
Anywhere on eth1 ALLOW IN 10.0.20.0/24
10.0.10.1 22/tcp on eth0 ALLOW IN 10.0.10.0/24

Anywhere ALLOW OUT 10.0.20.0/24 on eth1
```

Command used to add this rule:

```
ufw allow in on eth0 to 10.0.10.1 port 22 from 10.0.10.0/24 proto tcp
```

Q 3

The main advantage is that opening the port allows for remote maintenance outside of the internal network.

The disadvantages are that this opens up an attack vector to the firewall and also if no restriction is set to the user that login via ssh, then they can access the rest of the network freely by having nested ssh sessions.

1.4 Stateful Filtering

UFW configuration after completing this sub task

```
Status: active
Logging: on (low)
Default: deny (incoming), deny (outgoing), deny (routed)
New profiles: skip

To Action From
--
Anywhere on eth1 ALLOW IN 10.0.20.0/24
10.0.10.1 22/tcp on eth0 ALLOW IN 10.0.10.0/24

Anywhere ALLOW OUT 10.0.20.0/24 on eth1

Anywhere on eth0 ALLOW FWD 10.0.20.0/24 on eth1
```

Command used to add this rule:

```
ufw route allow in on eth1 out on eth0 from 10.0.20.0/24 to any
```

Q 4

TCP and UDP can be verified using `nc` servers as you can run that server in either TCP or UDP mode. When I host a `nc` server on the `outside-host` I can successfully connect to it from the `inside-host`, both when the server is in TCP and UDP mode.

This is not the case if I instead host the server on the `inside-host` and try to connect from the `outside-host` .

1.5 Opening Ports

UFW configuration after completing this sub task

```
Status: active
Logging: on (low)
Default: deny (incoming), deny (outgoing), deny (routed)
New profiles: skip

To Action From
--
Anywhere on eth1 ALLOW IN 10.0.20.0/24
10.0.10.1 22/tcp on eth0 ALLOW IN 10.0.10.0/24

Anywhere ALLOW OUT 10.0.20.0/24 on eth1

Anywhere on eth0 ALLOW FWD 10.0.20.0/24 on eth1
10.0.20.2 9000 on eth1 ALLOW FWD 10.0.10.2 on eth0
```

Command used to add this rule:

```
ufw route allow in on eth0 out on eth1 from 10.0.10.2 to 10.0.20.2 port 9000
```

Q 5

The configuration can be verified by hosting a `nc` server on the `inside-host` first on a random port, `1024` for example, and then on `9000` .

The `outside-host` can only connect to the server when it is hosted on port `9000` .

1.6 Blocking Ports

UFW configuration after completing this sub task

```
Status: active
Logging: on (low)
Default: deny (incoming), deny (outgoing), deny (routed)
New profiles: skip

To Action From
--
Anywhere on eth1 ALLOW IN 10.0.20.0/24
10.0.10.1 22/tcp on eth0 ALLOW IN 10.0.10.0/24

Anywhere ALLOW OUT 10.0.20.0/24 on eth1

135 on eth0 DENY FWD 10.0.20.0/24 on eth1
Anywhere on eth0 ALLOW FWD 10.0.20.0/24 on eth1
10.0.20.2 9000 on eth1 ALLOW FWD 10.0.10.2 on eth0
```

Command used to add this rule:

```
ufw route insert 1 deny in on eth1 out on eth0 from 10.0.20.0/24 to any port 135
```

Q 6

The configuration can be verified by hosting a `nc` server on the `outside-host` first on a random port, `1024` for example, and then on `135`.

The `inside-host` can only connect to the server when it is not hosted on port `135`.

Task 2 - Defending against SSG Brute-force Attacks

UFW configuration after completing this task

```
Status: active
Logging: on (low)
Default: deny (incoming), deny (outgoing), deny (routed)
New profiles: skip

To Action From
--
Anywhere on eth1 ALLOW IN 10.0.20.0/24
10.0.10.1 22/tcp on eth0 ALLOW IN 10.0.10.0/24
22 LIMIT IN Anywhere

Anywhere ALLOW OUT 10.0.20.0/24 on eth1

135 on eth0 DENY FWD 10.0.20.0/24 on eth1
Anywhere on eth0 ALLOW FWD 10.0.20.0/24 on eth1
10.0.20.2 9000 on eth1 ALLOW FWD 10.0.10.2 on eth0
```

Q 7

UFW includes a rate limit option that is very simple to set up. To rate limit the ssh port specifically you can run the following command: `ufw limit to any port 22`. This command will limit the rate of messages on port 22 that can come from a certain address every minute.

As I did not specify a network interface this will also apply to any clients on the internal network, which you might want to change if you completely trust the internal network.

Q 8

I used the brute-force command available in netwox to test the configuration. I created a file containing several passwords, in this case I chose 35. Then I ran the following command to spam port 22 on the firewall with TCP packets: `netwox 101 -i 10.0.20.1 -p 22 -w pass.txt -L ubuntu@10.0.20.1 -n 35`

This results in the following showing up in Wireshark:

1024	664	11487735	10.0.20.1	10.0.20.3	TCP	66	22	→	49846	[ACK] Seq=3192399801 Ack=1060901301 Win=65280 Len=0 TSval=1527954435 TSecr=81509468
1026	664	11487735	10.0.20.1	10.0.20.3	TCP	66	22	→	49846	[RST, ACK] Seq=3192399801 Ack=1060901301 Win=0 Len=0 TSval=1527954436 TSecr=81509467
1026	664	119094521	10.0.20.1	10.0.20.3	SSH	197	Server: Protocol (SSH-2.0-OpenSSH.7.6p1 Ubuntu-ubuntu0.3)			
1027	664	119094521	10.0.20.3	10.0.20.1	TCP	66	48986	→	22	[ACK] Seq=905098404 Ack=2893685622 Win=64256 Len=0 TSval=81509472 TSecr=1527954439
1028	664	120216809	10.0.20.1	10.0.20.3	SSH	197	Server: Protocol (SSH-2.0-OpenSSH.7.6p1 Ubuntu-ubuntu0.3)			
1029	664	120227596	10.0.20.3	10.0.20.1	TCP	66	48978	→	22	[ACK] Seq=2820909726 Ack=1138453233 Win=64256 Len=0 TSval=81509473 TSecr=1527954440
1030	664	121185766	10.0.20.1	10.0.20.3	TCP	66	22	→	49810	[RST, ACK] Seq=4147617541 Ack=2783046345 Win=65280 Len=0 TSval=1527954441 TSecr=81509467
1031	664	121226991	10.0.20.1	10.0.20.3	TCP	66	22	→	49844	[RST, ACK] Seq=2284911372 Ack=2296830274 Win=65280 Len=0 TSval=1527954441 TSecr=81509467
1032	664	121905057	10.0.20.1	10.0.20.3	TCP	66	22	→	49820	[RST, ACK] Seq=3005208806 Ack=771458424 Win=65280 Len=0 TSval=1527954442 TSecr=81509467
1033	664	128567088	10.0.20.1	10.0.20.3	TCP	66	22	→	49822	[RST, ACK] Seq=373727513 Ack=3385438988 Win=65280 Len=0 TSval=1527954449 TSecr=81509467
1034	664	128922039	10.0.20.1	10.0.20.3	TCP	66	22	→	49826	[RST, ACK] Seq=3261454656 Ack=3597821575 Win=65280 Len=0 TSval=1527954449 TSecr=81509467
1035	664	133127934	10.0.20.1	10.0.20.3	TCP	66	22	→	49832	[RST, ACK] Seq=3313199830 Ack=1199437027 Win=65280 Len=0 TSval=1527954453 TSecr=81509468
1036	664	133489605	10.0.20.1	10.0.20.3	TCP	66	22	→	49836	[RST, ACK] Seq=4077664952 Ack=1640330296 Win=65280 Len=0 TSval=1527954454 TSecr=81509468
1037	664	134664142	10.0.20.1	10.0.20.3	TCP	66	22	→	49838	[RST, ACK] Seq=4233919346 Ack=2891910648 Win=65280 Len=0 TSval=1527954455 TSecr=81509468
1038	664	135060962	10.0.20.1	10.0.20.3	TCP	66	22	→	49842	[RST, ACK] Seq=890430843 Ack=4264498228 Win=65280 Len=0 TSval=1527954455 TSecr=81509468
1039	664	135590416	10.0.20.1	10.0.20.3	SSH	197	Server: Protocol (SSH-2.0-OpenSSH.7.6p1 Ubuntu-ubuntu0.3)			
1040	664	135609423	10.0.20.3	10.0.20.1	TCP	66	48982	→	22	[ACK] Seq=2956179895 Ack=2578881959 Win=64256 Len=0 TSval=81509489 TSecr=1527954456

In other words, the rate limit makes the firewall respond with TCP messages with the *reset* flag enabled to indicate that the messages are not being accepted.