LAB :: iptables/ip6tables

- In this LAB we will see how to configure firewall for ipv4/ipv6 traffic using `iptables` and `ip6tables` command.
- OS Ubuntu 14.04

Login to your server

- Windows: use puTTY
- Mac and Linux: use your terminal
- Username `apnic` and password `training`
- Login to your server using the above username and password.

Status of the firewall

In a newly installed Ubuntu server the firewall chains are empty by default. To see the chains type the below command:

```
sudo iptables -L

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
```

```
sudo ip6tables -L

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
```

From the above output we can see that all the `chains` **(INPUT, FORWARD, OUTPUT)** are empty and the default `policy` for the `chains` are set to **ACCEPT**.

Chains

With –S (that's a capital S) flag all chains and their default policy can be seen. Type the below command to see the existing chain (at this point only default) and their default policy set for ipv4 and ipv6 traffic:

```
sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

```
sudo ip6tables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

As you can see the default policy is set to `ACCEPT` for all the chains.

First Rule

Let's make our first rule. Type the below command to add the below rule to our `INPUT` chain.

```
sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

```
sudo ip6tables -A INPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

For this rule, we are not going to disconnect from the server.

Allow Rules for SSH

Let's add some rules in our firewall. We are connected with our server through `SSH`. Let's allow `SSH` traffic. Type the below command to allow `SSH` traffic to our server.

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

```
sudo ip6tables -A INPUT -p tcp --dport 22 -j ACCEPT
```

The above two commands will allow `SSH` traffic to our server for ipv4 and ipv6 hosts. Type the below command to check the status of the firewall.

```
sudo iptables -L
```

```
sudo ip6tables -L
```

To check only the `INPUT` chain type the below command.

```
sudo iptables -L INPUT --line-numbers -v
```

```
sudo ip6tables -L INPUT --line-numbers -v
```

`--line-numbers` will show the `INPUT` rules with line number and `-v` will show the number of packets and the aggregate size of the packet sent to the server.

Allow Rules for SSH from specific host

It's a good practice to allow remote login from specific hosts or from specific network. First clear the rules which we enter in the `INPUT` chain. (For the simplicity of the LAB we flush the chain, but don't flush chains until unless you really need it in real life.)

```
sudo iptables -F INPUT
```

```
sudo ip6tables -F INPUT
```

The chain will be empty now. Type the below command to allow `SSH` connection only from your LAPTOP. Before adding this rule again add the **`First rule`**.

```
sudo iptables -A INPUT -p tcp --dport ssh -s LAPTOP_IPV4_ADDRESS
-j ACCEPT
```

```
sudo ip6tables -A INPUT -p tcp --dport ssh -s
LAPTOP_IPV6_ADDRESS -j ACCEPT
```

Test your work

Things we did:

1. Allow `SSH` traffic only from our host address.

To test your rule invite other groups to login to your server. If they can login or get the `login as:` prompt then your rule is not working. **Do you know why?**

Allow http and other traffic

To do this exercise we need to install the `apache2` package. Type the below command to install `apache2` on your server.

```
sudo apt-get install apache2 -y
```

Open a web browser and type your group server address in the browser address field. If you solved the question in the `SSH` section then you should not see any page, if not then you should see the apache2 Ubuntu default page in your browser.

Type the below command to allow http traffic.

```
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

```
sudo ip6tables -A INPUT -p tcp --dport 80 -j ACCEPT
```

You should able to see the `apache2` Ubuntu default page in your web browser. Using the above `http` example, you can enable others required port like `ftp (21), smtp (25), https (443)` etc in your server.

Change policy

If you need to change the default policy, then type the below command. Let's change our `INPUT` policy from `ACCEPT` to `DROP`.

```
sudo iptables -P INPUT DROP
```

```
sudo ip6tables -P INPUT DROP
```

Delete iptables rules

In some case you need to delete one or more than one rule from your `iptables` chains. There are two ways you can delete the rules from the chain. By rule specification and by rule number.

Type the below command to delete rules by rule specification. Let's delete the `ftp (21)` rule.

```
sudo iptables -D INPUT -p tcp --dport 21 -j ACCEPT
```

```
sudo ip6tables -D INPUT -p tcp --dport 21 -j ACCEPT
```

Just like append (`A`) command, replace the **A** with **D** for rule deletion.

Type the below command to delete rules by rule number. Let's delete the `smtp (25)` rules.

To see the rules line number, type the below command

```
sudo iptables -L --line-numbers
```

```
sudo ip6tables -L --line-numbers
```

You will see the rules with line numbers. To delete the rules from a chain type the below command.

```
sudo iptables -D INPUT RULES_LINE_NUMBER
```

```
sudo ip6tables -D INPUT RULES_LINE_NUMBER
```

Be careful while deleting rules by rule number. Cause After deleting one rule the order of the chain number changed.

Insert Rule

`iptables` rules are working sequentially, and if a match found then the rest of the rules will be skipped. If you want to rearrange your rules or want to add a new rule in a specific position, then first list the rules with `--line-numbers` option then type the below command.

```
sudo iptables -I INPUT 2 -p tcp --dport 21 -j ACCEPT
```

```
sudo ip6tables -I INPUT 2 -p tcp --dport 21 -j ACCEPT
```

This will place the rule in 2nd position of the `INPUT` chain.

New chain

You can create or your own chain in `iptables`. Type the below command to create a new chain name `LOG_TAB` or any other name you like just don't use space.

```
sudo iptables -N LOG_TAB
```

```
sudo ip6tables -N LOG_TAB
```

To delete the chain simply type the below command.

```
sudo iptables -X LOG_TAB
```

```
sudo ip6tables -X LOG_TAB
```

iptables LOG

One of the most interesting feature of `iptables` is logging facility. Perform the activity below to enable `logging` for `INPUT` chain drop packets.

Let's Create a new chain first for `logging` name `LOG_TAB` or any other name you like, just don't use any space in the name.

```
sudo iptables -N LOG_TAB
```

```
sudo ip6tables -N LOG_TAB
```

Next, make sure all the remaining incoming connections jump to the `LOGGING` chain as shown below.

```
sudo iptables -A INPUT -j LOG_TAB
```

```
sudo ip6tables -A INPUT -j LOG_TAB
```

Next, log these packets by specifying a custom "`log-prefix`".

```
sudo iptables -A LOG_TAB -j LOG --log-prefix "IPTables Packet
Dropped: " --log-level 7
```

```
sudo ip6tables -A LOG_TAB -j LOG --log-prefix "IPTables Packet
Dropped: " --log-level 7
```

Finally, drop these packets.

```
sudo iptables -A LOG_TAB -j DROP
```

```
sudo ip6tables -A LOG_TAB -j DROP
```

Let's see the chains now by typing the below command

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

```
sudo ip6tables -L -n
```

Now let's test our work. By default `iptables` log messages in `/var/log/kern.log` file. We haven't made any rules for `ICMP` traffic and our `INPUT` chain default policy is set to `DROP`, so any other traffic, which is not allowed will be dropped by `iptables` and that message can be seen in `kern.log` file. Type the below command to see the log file.

```
sudo tail -f /var/log/kern.log | grep ICMP
```

Now initiate a `ping` command from your laptop to your server IP and check the log file.

Allow ICMP traffic

For this lab we have already added a drop rule for logging, so we need to `INSERT` the rule before that drop rule. Type the below command for allowing `ICMP` traffic.

```
sudo iptables -I INPUT 7 -p icmp -j ACCEPT
```

```
sudo ip6tables -I INPUT 7 -p icmp -j ACCEPT
```

You can `ping` your server from your host.

iptables Save

`iptables` rules are working instantly, but when you restart your server then all rules will be gone. That's why you need to save the rules so that they become active after a reboot. There are several ways to do this, but the easiest way is to use the `iptables-persistent` package. Type the below command to install the `iptables-persistent` package.

```
sudo apt-get install iptables-persistent
```

Press "`yes`" for both ipv4 and ipv6 rules when prompted. After installation, you will find two files in `/etc/iptables` location name `ipv4` and `ipv6`. You can open the file and you can make your change here. You can do `start|restart|reload|force-reload|save|flush` from here. For example, if you want to save the current loaded `iptables` rules type the below command.

```
sudo /etc/init.d/iptables-persistent save
```

This will save both ipv4 and ipv6 rules.

```
******END of LAB******
```