Network Security

Tutorials

Firewalls (Linux IPTables)

Agenda

- Introduction to Linux Firewall
- IP Table Basic Commands
- Writing IP Table Rules
- IPTables Best Practice

Linux Kernel Firewall

- The Linux kernel provides **Xtables** as a collection of **Netfilter** modules.
- **What is Netfilter?**
  a. Is a **network traffic framework** provided by the Linux OS that allows several network traffic operations.
  b. It is **in-kernel packet classification engine** written in C
  c. The most stable release is **6.13** on **20 January 2025; 11 days ago**
- The most common **network traffic operations** supported by Netfilter are:
  a. packet filtering
  b. network address translation
  c. port address translation

Linux Kernel Firewall

- Linux Kernel Firewall uses **Netfilter hooks** to interact with incoming and outgoing traffics.
- Netfilter provide **5 hooks** that any program can register with. As packets progress through the stack, they will trigger the kernel modules that have registered with these hooks.
- Based on the **traffic direction**, and **status** a specific hook will be triggered and the program listening to that hook will be notified

Netfilter Hooks

The Five Netfilter Hooks are:

1. NF_IP_PRE_ROUTING: This hook will be triggered by any incoming traffic very soon after entering the network stack. This hook is processed before any routing decisions have been made regarding where to send the packet.
2. NF_IP_LOCAL_IN: This hook is triggered after an incoming packet has been routed if the packet is destined for the local system.
3. NF_IP_FORWARD: This hook is triggered after an incoming packet has been routed if the packet is to be forwarded to another host.

Netfilter Hooks

The Five Netfilter Hooks are:

4. NF_IP_LOCAL_OUT: This hook is triggered by any locally created outbound traffic as soon it hits the network stack.
5. NF_IP_POST_ROUTING: This hook is triggered by any outgoing or forwarded traffic after routing has taken place and just before being put out on the wire.

IPTables in Linux

- IPTables is a **user-space utility program** for Linux administrators to configure the Linux kernel firewall Xtables.
- The Linux Kernel firewall provide both basic packet filtering and stateful inspection firewall functions.
- IPTables enable the system admin to organize and configure firewall rules by using the concept of **tables**. The tables contain a **set of chains** and the chains contain a **set of rules**.
- There are four types of tables: **Filter** table, **NAT** table, **Mangle** table, **Raw** table, **Security** table

Firewall Tables and Chains: Filter Table

- The most widely used tables in iptables
- The filter table is used to make decisions about whether to let a packet continue to its intended destination or to deny its request
- The **Filter Table** contains the following chains
  - **Input Chain:** an incoming packet to your host or your local network
  - **Output Chain:** an outgoing packet from your host or your local network
  - **Forward Chain:** If the packets, neither the source nor the destination belongs to your network.

Firewall Tables and Chains: NAT Table

- The network address translation table
- The NAT tables contains the following chains:
  - **PREROUTING chain :** This chain is mainly for **DNAT** (Destination NAT). This will edit/change the **destination IP** of the packet.
  - **POSTROUTING chain:** This chain is mainly for **SNAT** (Source NAT). This will edit/change the **source IP** of the packet.
  - **OUTPUT chain:** If the packets get delivered locally, this chain is used.

Firewall Tables and Chains: Mangle Table

- This table allows us to **edit the IP headers of the packet**. For example, we could set the **TTL** value using this table.
- The mangle table could be used to **mark the packet** for further processing by other tables.
- The mangle table contains the following **chains**:
  - **PREROUTING**
  - **OUTPUT**
  - **FORWARD**
  - **INPUT**
  - **POSTROUTING**

Firewall Tables and Chains: RAW & Security Tables

- **Raw Table:**
  - The raw table this mainly used for **marking the packets** to enable the **stateful packet inspection**. This mark does not touch the actual packet but adds the mark to the kernel's representation of the packet.
- **Security Table:**
  - The security table is used to set internal **SELinux security** context marks on packets. This only for SELinux or other systems that can interpret SELinux security contexts.

IPTables: Chain Traversal Order

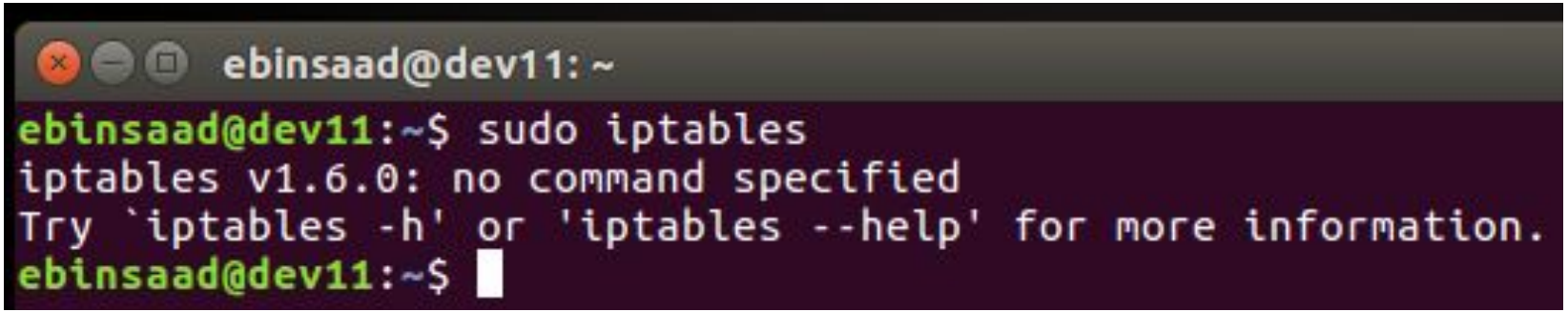
In general all the IP packets will go into one of the following paths, depend on the direction of the packet.

- **Case 1:** PREROUTING -> INPUT
  - An incoming packet from external network to your local network
- **Case 2:** PREROUTING -> FORWARD -> POSTROUTING
  - An incoming packet from external network to another external network or host
- **Case 3:** OUTPUT -> POSTROUTING
  - An outgoing packet from your internal network to an external network or host.

Working with IPTables

Almost all Linux distro comes with IPtables installed on it. To check if IPTables is installed on your Linux box use the following command:

```
sudo iptables
```



Installing IPTables

If IPTables is not installed on your Linux machine you need to check your linux distro manual for IPtables installation instructions. Here are some examples:

For **Debian and Ubuntu** Linux

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

For **Redhat/CentOS 7**

```
yum install iptables-services -y
systemctl enable iptables
systemctl start iptables
systemctl status iptables
```

IPTables: list the chains status

To list the IPtables Filter chains use the following command

```
sudo iptables -L
```

```
ebinsaad@dev11: ~
ebinsaad@dev11:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 71 packets, 11593 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 71 packets, 8148 bytes)
 pkts bytes target     prot opt in     out     source               destination
ebinsaad@dev11:~$
```

IPTables: list the chains for other tables

To list the **IPtables NAT** table chains use the following command

sudo iptables -t <table name> -L -v

sudo iptables -t nat -L -v

```
ebinsaad@dev11:~$ sudo iptables -t nat -L -v
Chain PREROUTING (policy ACCEPT 3 packets, 196 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 3 packets, 196 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 68 packets, 4185 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 68 packets, 4185 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

IPTables Exercise: List Chains & Tables

Each table (filter, mangle, raw and security) uses specific chain. Using IPTables command list the chains used by each table.

IPTables Rules General Notes

1. The rules are placed within a specific chain of a specific table.
2. When chain is called or triggered by packet passing through the table, the packet in question will be checked against each rule within the chain in order.
3. We can create our own chain (user-defined chain) and linked to an existing chain.
4. Each rule has a matching component and an action component.

IPTables: Working with Rules

5. The **matching portion** of a rule specifies the criteria that a packet must meet to fire the rule.
6. The action of a given rule could be a **terminating action** or a **non terminating action**
7. Note each chain must eventually **pass back a final terminating action**. In other words ends with a rule , where the rule action is a terminating action.

IPTables: Working with Rules

Firewall Rule Actions

- ALLOW (aka ACCEPT)
  - Permit a packet to traverse the firewall. This would be the behaviour if the firewall was not present.
- REJECT
  - Prohibit a packet from passing. Send an ICMP destination-unreachable back to the source host [unless the icmp would not normally be permitted, eg. if it is to/from the broadcast address].
- DROP (aka DENY, BLACKHOLE)
  - Prohibit a packet from passing. Send no response.

Creating an IPTables rule

- Creating a rule means appending a new rule to the end of a given chain.
- Any table (e.g. filter, NAT, etc) that uses this chain will be affected by the rules in this chain.
- AN example for a general syntax of an IPtables rule is:

sudo iptables -A -i <interface> -p <(tcp/udp)> -s <source> --dport <port no> -j <target>

-A: for append
-i : to set the network interface
-p: to set the protocol
-s: set the source IP
-dport: set the destination port number
-j: to specify the action when the rule match the packet

IPTables Rules: Examples

Let us say we have an application that connect to a database on the same host (local host) or a web application that connect to a proxy server on the same host. We can simply define a rule that enable all incoming traffic on local host.

```
sudo iptables -A INPUT -i lo -j ACCEPT
```

```
ebinsaad@dev11:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 6 packets, 761 bytes)
 pkts bytes target     prot opt in     out     source               destination
   28  4390 ACCEPT     all  --  lo     any     anywhere             anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 32 packets, 4678 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

IPTables Rules: Examples

OK let us write a rule to accept all incoming traffic to **HTTP** and **HTTPs**

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

sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

IPTables Rules: Examples

```
ebinsaad@dev11:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
ebinsaad@dev11:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
ebinsaad@dev11:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 1 packets, 71 bytes)
 pkts bytes target     prot opt in     out     source               destination
  796  120K ACCEPT     all  --  lo     any     anywhere             anywhere
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:http
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:https

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 5 packets, 566 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

IPTables Rules: What the does following rules mean?

```
ebinsaad@dev11:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 2 packets, 104 bytes)
 pkts bytes target     prot opt in     out     source               destination
   38  5159 ACCEPT     all  --  lo     any     anywhere             anywhere
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:http
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:https
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ftp
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 10 packets, 1086 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

- **sudo iptables -A INPUT -i lo -j ACCEPT**
- **sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT**
- **sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT**
- **sudo iptables -A INPUT -p tcp --dport 21 -j ACCEPT**
- **sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT**

IPTables Rules: Accept/Reject Traffic based on source IP

We can filter incoming traffic based on the source IP address as following

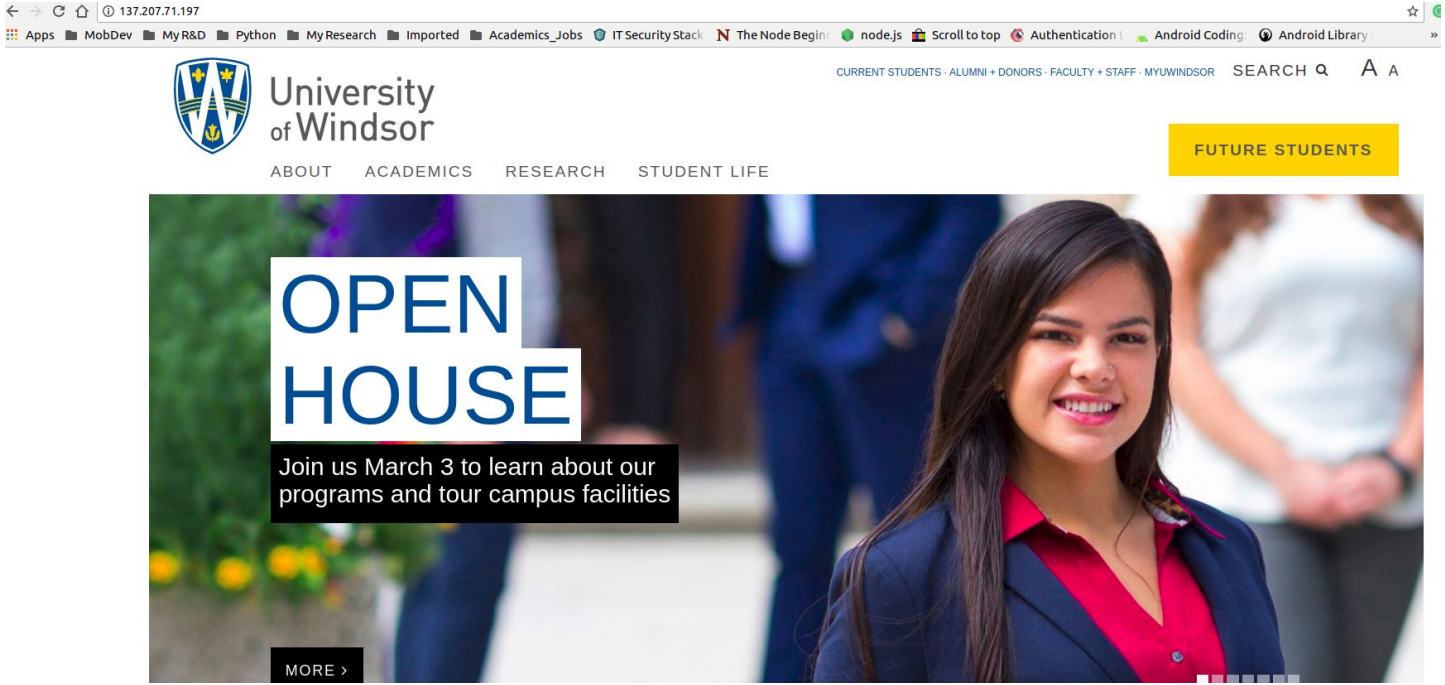
```
sudo iptables -A INPUT -p tcp  -s 142.172.8.4 -j ACCEPT

sudo iptables -A INPUT -p udp -s 142.172.8.4 -j ACCEPT

sudo iptables -A INPUT -p udp -s 137.207.71.197 -j DROP

sudo iptables -A INPUT -p tcp -s 137.207.71.197 -j DROP
```

IPTables Rules: Accept/Reject Traffic based on source IP

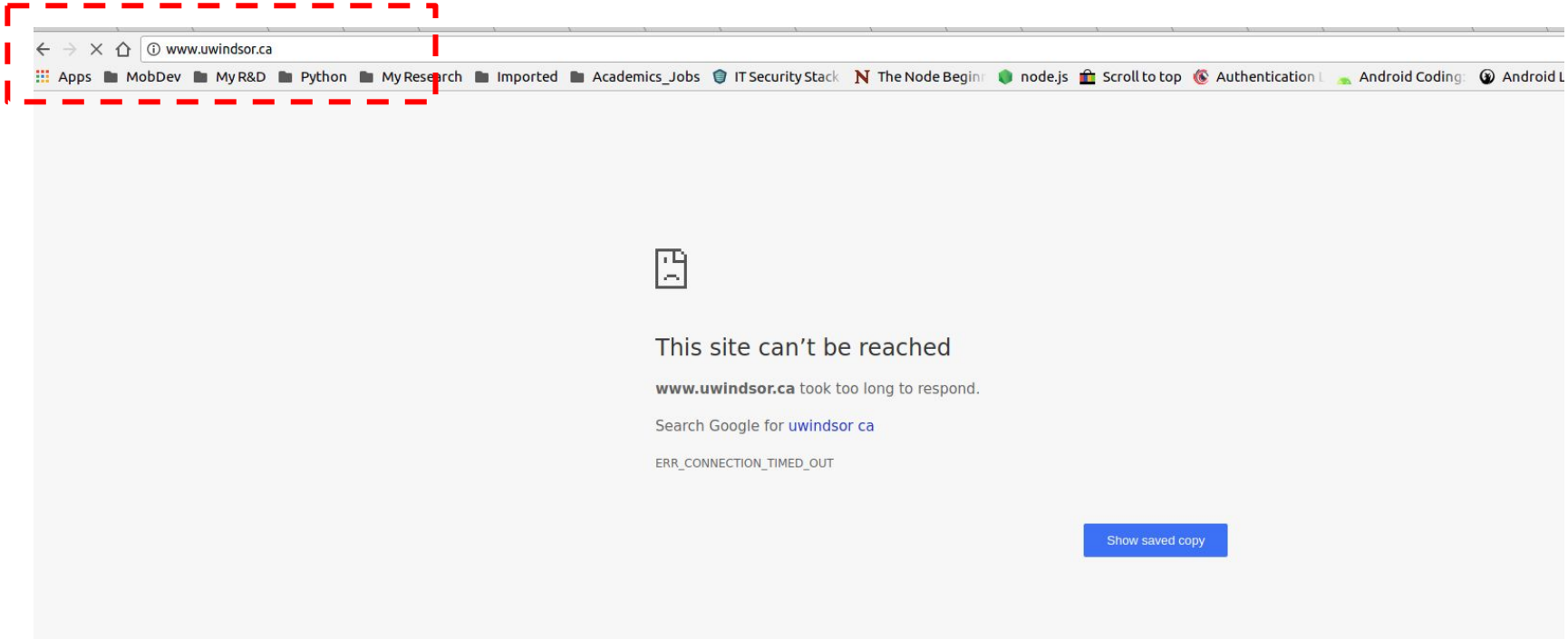


137.207.71.197 = www.uwindsor.ca

IPTables Rules: Accept/Reject Traffic based on source IP

```
ebinsaad@dev11:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 248 packets, 45241 bytes)
 pkts bytes target     prot opt in     out     source               destination
 1964  303K ACCEPT     all  --  lo     any     anywhere             anywhere
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:http
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:https
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ftp
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ssh
    0     0 ACCEPT     tcp  --  any    any     142.172.8.4          anywhere
    0     0 ACCEPT     udp  --  any    any     142.172.8.4          anywhere
    0     0 DROP       udp  --  any    any     www.uwindsor.ca      anywhere
   42  2520 DROP       tcp  --  any    any     www.uwindsor.ca      anywhere
```

IPTables Rules: Accept/Reject Traffic based on source IP



IPTable Rules: Deleting a Rule

Now let us assume we added a rule by mistake or we want to delete a rule because the security policy changed. To delete a specific rule:

First we need to **find the rule number/ID** in the chain

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

Then we use the delete flag with the rule line number

```
sudo iptables -D INPUT 8
```

IPTable Rules: Deleting a Rule

```
ebinsaad@dev11:~$ sudo iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
2    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http
3    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https
4    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ftp
5    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
6    ACCEPT     tcp  --  142.172.8.4          anywhere
7    ACCEPT     udp  --  142.172.8.4          anywhere
8    DROP       udp  --  www.uwindsor.ca      anywhere
9    DROP       tcp  --  www.uwindsor.ca      anywhere
```

When deleting more than one rule, start deleting form the end of the chain (rule with largest id first)

```
sudo iptables -D INPUT 9
sudo iptables -D INPUT 8
```

Saving/Delete IPTables Rules

Iptables rules we have created are saved in memory. That means we have to redefine them on reboot. To make these changes persistent after reboot use the following command

sudo iptables-save

To delete IPtables rules (all the rules) use the flush flag -F as below:

sudo iptabes -F

sudo iptables-save

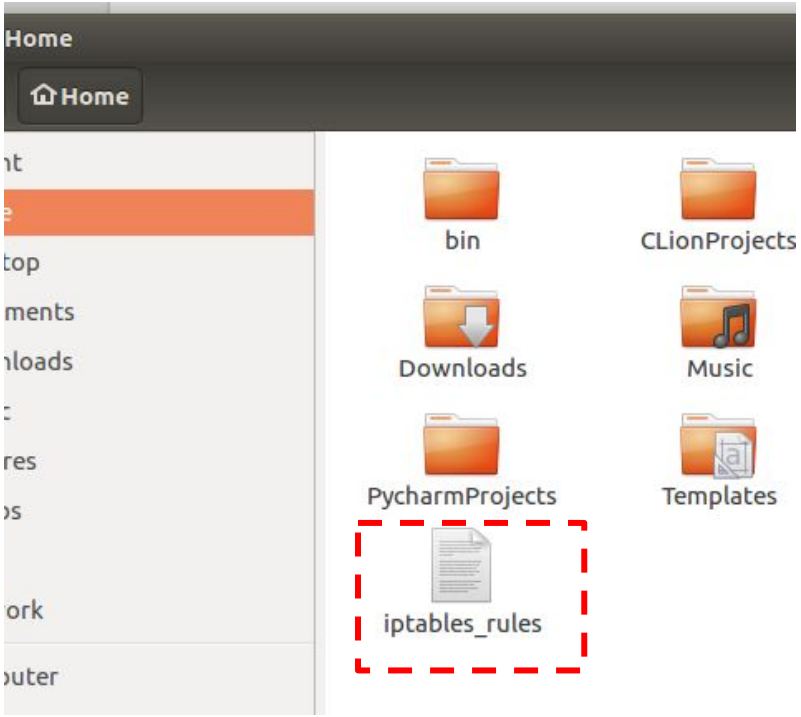
Export/Import IPTables Rules

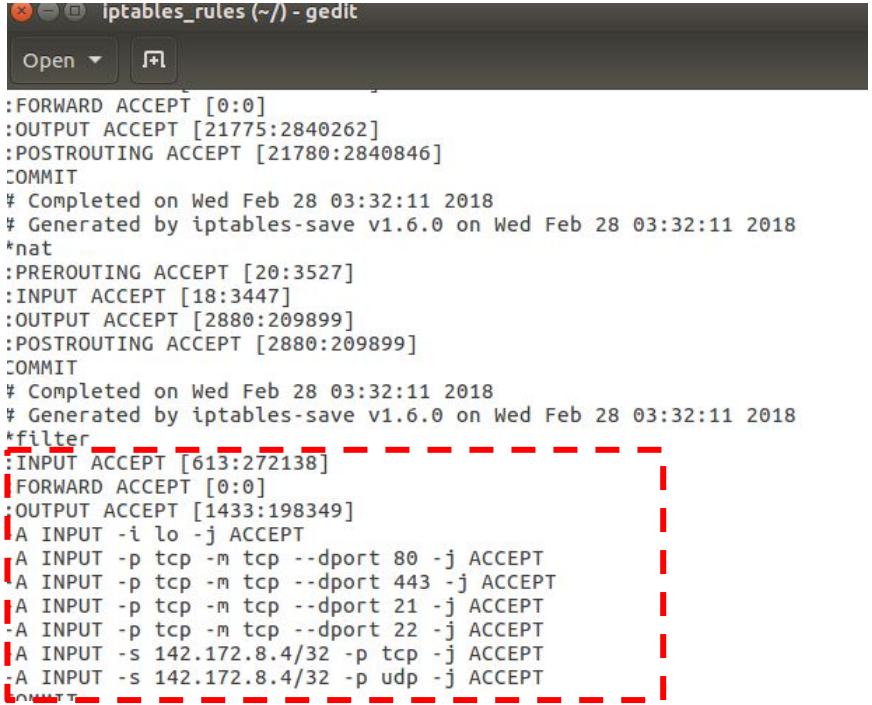
To export IPTables rules for example to migrate the rules to a new firewall server or host. Use the following command

```
sudo iptables-save > iptables_rules
```

This will save the current IPtables rules in a text file named iptables_rules on your home directory.

Export/Import IPTables Rules





Export/Import IPTables Rules

To import an iptables rules from a file use the following command

sudo iptables-restore < {full path to your iptables file }

sudo iptables-restore < /home/ebinsaad/iptables_rules

IPTables Rules: Outgoing Traffic

Let us allow outgoing SSH traffic from our local host

```
sudo iptables -A OUTPUT -o eth0 -p tcp --dport 22 -m state
--state NEW, ESTABLISHED -j ACCEPT
```

```
sudo iptables -A INPUT -i eth0 -p tcp --sport 22 -m state
--state ESTABLISHED -j ACCEPT
```

IPTables Rules: Outgoing Traffic

Let us allow **outgoing SSH** traffic from our local host **to specific target network**

```
sudo iptables -A OUTPUT -o eth0 -p tcp -d 192.168.4.0/24
--dport 22 -m state --state NEW, ESTABLISHED -j ACCEPT
```

```
sudo iptables -A INPUT -i eth0 -p tcp --sport 22 -m state
--state ESTABLISHED -j ACCEPT
```

IPTables Rules: Outgoing Traffic

How would we allow outgoing HTTPs traffic?

```
sudo iptables -A OUTPUT -o eth0 -p tcp --dport 443 -m state
--state NEW,ESTABLISHED -j ACCEPT

sudo iptables -A INPUT -i eth0 -p tcp --sport 443 -m state
--state ESTABLISHED -j ACCEPT
```

IPTables Rules: Allowing Ping Traffic

Allowing Ping from **Outside** **to Inside**

sudo iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

sudo iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT

Allowing Ping from **Inside to Outside**

sudo iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT

sudo iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

IPTables Exercise: Enabling MySQL Access

Let us assume we have a MYSQL server running on IP 192.168.4.7 and port 3306 and we want to allow only host running on IP 192.168.1.102 to connect to this MYSQL server. Write the IPTables rules required to enforce this policy.

IPTables Prevent DoS Attack

Flooding DOS Attacks could be **mitigated** using IPTables:

iptables -A INPUT -p tcp --dport 80 -m limit --limit 30/minute --limit-burst 250 -j ACCEPT

-m limit: This uses the limit iptables extension
–limit 30/minute: This limits only maximum of 30 connections per minute.
–limit-burst 250: This value indicates that the limit/minute will be enforced only after the total number of connection have reached the limit-burst level.

IPTables: Log Action

We can log IPtables matching events using the log option. Note that the log action is not a terminal action For example let us assume we want to log IP spoofed packets before we drop them.

```
iptables -A INPUT -i eth0 -s 10.0.0.0/8 -j LOG --log-prefix "IP_SPOOF A: "

iptables -A INPUT -i eth0 -s 10.0.0.0/8 -j DROP
```

IPTable Best Practice

1. Understand your **network security policy** and build your rules to fit this policy.
2. Decide what your filtering method is. Is it **positive** mode filter or **negative** filter. **For example**, if you use positive filtering then make sure the last rule in your chain is to drop all traffic.
3. Design and document your rules first before adding them to the firewall.
4. Consider the **scope and coverage** of the rules when you add the rules to the chain with respect to the **security policy**

IPTable Best Practice

5. Remember the rules are always evaluated from top to bottom
6. **Split complicated rule groups into separate chains**. Iptables allow you to create additional chains and hook them to another chain
7. **Test your rules** before deploying them in production environment
8. **Log** the traffic that matches each rule
9. Use **REJECT** until you know your rules are working properly. This will help you in troubleshooting.

IPTable Best Practice

10. **Use comments** to explain complex rules, this can be done using -m comment --comment option

    iptables -A INPUT -j DROP -p tcp --dport 22 -m comment --comment "limit ssh access"

11. Always **save and backup** your firewall rules

IPTable: Create User Chain

Let us create a new chain to log and drop packets that does not match our positive filtering rules in the Input Chain

1. Create a new user chain with the name LOGGING

   iptables -N LOGGING

2. Append this chain to the end of the INPUT chain

   iptables -A INPUT -j LOGGING

3. Add a log action to the LOGGING chain

   iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix "IPTables Packet Dropped: " --log-level 7

4. Finally drop the packets that reach the LOGGING chain

   iptables -A LOGGING -j DROP

What is wrong in the following rules?

```
ebinsaad@dev11:~$ sudo iptables  -L -v --line-number
Chain INPUT (policy ACCEPT 32 packets, 6298 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1      642 88668 ACCEPT     all  --  lo     any     anywhere             anywhere
2        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:http
3        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:https
4        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ftp
5        0     0 DROP       tcp  --  any    any     anywhere             192.168.0.12         tcp dpt:ssh
6        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ssh
7        0     0 ACCEPT     tcp  --  any    any     142.172.8.4          anywhere
8        0     0 ACCEPT     udp  --  any    any     142.172.8.4          anywhere
9        0     0 ACCEPT     tcp  --  any    any     mtprnf0117w-142-167-20-1.dhcp-dynamic.fibreop.nl.bellaliant.net  192.168.0.12
   tcp dpt:ssh
```

What is wrong in the following rules?

```
ebinsaad@dev11:~$ sudo iptables  -L -v --line-number
Chain INPUT (policy ACCEPT 32 packets, 6298 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1      642 88668 ACCEPT     all  --  lo     any     anywhere             anywhere
2        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:http
3        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:https
4        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ftp
5        0     0 DROP       tcp  --  any    any     anywhere             192.168.0.12         tcp dpt:ssh
6        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ssh
7        0     0 ACCEPT     tcp  --  any    any     142.172.8.4          anywhere
8        0     0 ACCEPT     udp  --  any    any     142.172.8.4          anywhere
9        0     0 ACCEPT     tcp  --  any    any     mtprnf0117w-142-167-20-1.dhcp-dynamic.fibreop.nl.bellaliant.net  192.168.0.12
   tcp dpt:ssh
```

IPTable insert rule into a chain

An IPtables rule cannot be just "moved up" in the list, but it can be recreated with a fixed position. Using the **Insert flag -I**

sudo iptables -I INPUT 5 -p tcp -s 142.167.20.1 -d 192.168.0.12 --dport 22 -j ACCEPT

IPTable insert rule into a chain

```
ebinsaad@dev11:~$ sudo iptables  -L -v --line-number
Chain INPUT (policy ACCEPT 22 packets, 3579 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1     1070  150K ACCEPT     all  --  lo     any     anywhere             anywhere
2        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:http
3        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:https
4        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ftp
5        0     0 ACCEPT     tcp  --  any    any     mtprnf0117w-142-167-20-1.dhcp-dynamic.fibreop.nl.bellaliant.net  192.168.0.12
   tcp dpt:ssh
6        0     0 DROP       tcp  --  any    any     anywhere             192.168.0.12         tcp dpt:ssh
7        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ssh
8        0     0 ACCEPT     tcp  --  any    any     142.172.8.4          anywhere
9        0     0 ACCEPT     udp  --  any    any     142.172.8.4          anywhere
10       0     0 ACCEPT     tcp  --  any    any     mtprnf0117w-142-167-20-1.dhcp-dynamic.fibreop.nl.bellaliant.net  192.168.0.12
   tcp dpt:ssh
```

Questions