ARYAMAN MISHRA

19BCE1027

FIREWALL CONFIGURATIONS

UFW (**u**ncomplicated **f**ire**w**all) is a firewall configuration tool that runs on top of iptables, included by default within Ubuntu distributions. It provides a streamlined interface for configuring common firewall use cases via the command line.

Execute following commands:

apt-get update

```
┌──(root💀 kali)-[~]
└─# apt-get update
Get:1 https://mirrors.ocf.berkeley.edu/kali kali-rolling InRelease [30.6 kB]
Get:2 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 Packages [17.9 MB]
Get:3 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 Contents (deb) [40.1 MB]
Get:4 https://mirrors.ocf.berkeley.edu/kali kali-rolling/contrib amd64 Packages [111 kB]
Get:5 https://mirrors.ocf.berkeley.edu/kali kali-rolling/contrib amd64 Contents (deb) [148 kB]
Get:6 https://mirrors.ocf.berkeley.edu/kali kali-rolling/non-free amd64 Packages [210 kB]
Fetched 58.5 MB in 31s (1,902 kB/s)
Reading package lists ... Done
```

apt-install ufw

```
┌──(root💀 kali)-[~]
└─# apt install ufw
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
ufw is already the newest version (0.36.1-3).
0 upgraded, 0 newly installed, 0 to remove and 896 not upgraded.
```

```
┌──(root💀 kali)-[~]
└─# ufw --help

Usage: ufw COMMAND

Commands:
 enable                          enables the firewall
 disable                         disables the firewall
 default ARG                     set default policy
 logging LEVEL                   set logging to LEVEL
 allow ARGS                      add allow rule
 deny ARGS                       add deny rule
 reject ARGS                     add reject rule
 limit ARGS                      add limit rule
 delete RULE|NUM                 delete RULE
 insert NUM RULE                 insert RULE at NUM
 prepend RULE                    prepend RULE
 route RULE                      add route RULE
 route delete RULE|NUM           delete route RULE
 route insert NUM RULE           insert route RULE at NUM
 reload                          reload firewall
 reset                           reset firewall
 status                          show firewall status
 status numbered                 show firewall status as numbered list of RULES
 status verbose                  show verbose firewall status
 show ARG                        show firewall report
 version                         display version information

Application profile commands:
 app list                        list application profiles
 app info PROFILE                show information on PROFILE
 app update PROFILE              update PROFILE
 app default ARG                 set default application policy
```

Delete UFW Rule

To delete a rule that you previously set up within UFW, use ufw delete followed by the rule (allow or deny) and the target specification. The following example would delete a rule previously set to allow all connections from an IP address of 203.0.113.101:

```
sudo ufw delete allow from 203.0.113.101
```

```
Rule deleted
```

Another way to specify which rule you want to delete is by providing the rule ID. This information can be obtained with the following command:

```
sudo ufw status numbered
```

```
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] Anywhere                   DENY IN     203.0.113.100
[ 2] Anywhere on eth0           ALLOW IN    203.0.113.102
```

From the , you can see that there are two active rules. The first rule, with highlighted values, denies all connections coming from the IP address 203.0.113.100. The second rule allows connections on the eth0 interface coming in from the IP address 203.0.113.102.

Because by default UFW already blocks all external access unless explicitly allowed, the first rule is redundant, so you can remove it. To delete a rule by its ID, run:

```
sudo ufw delete 1
```

You will be prompted to confirm the operation and to make sure the ID you're providing refers to the correct rule you want to delete.

```
Deleting:
 deny from 203.0.113.100
Proceed with operation (y|n)? y
```

Rule deleted
If you list your rules again with sudo ufw status, you'll see that the rule was removed.

```
┌──(root💀 kali)-[~]
└─# ufw app list
Available applications:
  AIM
  Bonjour
  CIFS
  DNS
  Deluge
  IMAP
  IMAPS
  IPP
  KTorrent
  Kerberos Admin
  Kerberos Full
  Kerberos KDC
  Kerberos Password
  LDAP
  LDAPS
  LPD
  MSN
  MSN SSL
  Mail submission
  NFS
  Nginx Full
  Nginx HTTP
  Nginx HTTPS
  OpenSSH
  POP3
  POP3S
  PeopleNearby
  SMTP
  SSH
  Samba
  Socks
  Telnet
  Transmission
  Transparent Proxy
  VNC
  WWW
  WWW Cache
  WWW Full
  WWW Secure
  XMPP
  Yahoo
  qBittorrent
  svnserve
```

sudo ufw status

Verify UFW Status

To check if ufw is enabled, run:

- sudo ufw status
-

Status: inactive

The will indicate if your firewall is active or not.

```
┌──(root💀 kali)-[~]
└─# sudo ufw status
Status: active
```

ufw status verbose

```
┌──(root💀 kali)-[~]
└─# ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)
```

sudo ufw status numbered

```
┌──(root💀 kali)-[~]
└─# sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 80/tcp                     ALLOW IN    Anywhere
[ 3] 443                        ALLOW IN    Anywhere
[ 4] 80,443/tcp                 ALLOW IN    Anywhere
[ 5] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 6] 80/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 7] 443 (v6)                   ALLOW IN    Anywhere (v6)
[ 8] 80,443/tcp (v6)            ALLOW IN    Anywhere (v6)
```

ufw enable

Enable UFW

If you got a `Status: inactive` message when running `ufw status`, it means the firewall is not yet enabled on the system. You'll need to run a command to enable it.

By default, when enabled UFW will block external access to all ports on a server. In practice, that means if you are connected to a server via SSH and enable ufw before allowing access via the SSH port, you'll be disconnected. Make sure you follow the section on how to enable SSH access of this guide before enabling the firewall if that's your case.

To enable UFW on your system, run:

```
sudo ufw enable
```

You'll see like this:

Firewall is active and enabled on system startup
To see what is currently blocked or allowed, you may use the verbose parameter when running ufw status, as follows:

- sudo ufw status
-

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

```
┌──(root💀 kali)-[~]
└─# ufw enable
Firewall is active and enabled on system startup
```

ufw reset

```
┌──(root💀 kali)-[~]
└─# ufw reset
Resetting all rules to installed defaults. Proceed with operation (y|n)? y
Backing up 'user.rules' to '/etc/ufw/user.rules.20211124_110756'
Backing up 'before.rules' to '/etc/ufw/before.rules.20211124_110756'
Backing up 'after.rules' to '/etc/ufw/after.rules.20211124_110756'
Backing up 'user6.rules' to '/etc/ufw/user6.rules.20211124_110756'
Backing up 'before6.rules' to '/etc/ufw/before6.rules.20211124_110756'
Backing up 'after6.rules' to '/etc/ufw/after6.rules.20211124_110756'
```

Disable UFW

If for some reason you need to disable UFW, you can do so with the following command:

sudo ufw disable

Be aware that this command will fully disable the firewall service on your system.

Block an IP Address

To block all network connections that originate from a specific IP address, run the following command, replacing the highlighted IP address with the IP address that you want to block:

```
sudo ufw deny from 203.0.113.100
```

```
Rule added
```

In this example, from 203.0.113.100 specifies a **source** IP address of “203.0.113.100”.

If you run sudo ufw status now, you’ll see the specified IP address listed as denied:

```
Status: active

To                         Action      From
--                         ------      ----
Anywhere                   DENY        203.0.113.100
```

All connections, coming in or going out, are blocked for the specified IP address.

Block a Subnet

If you need to block a full subnet, you may use the subnet address as from parameter on the ufw deny command. This would block all IP addresses in the example subnet 203.0.113.0/24:

```
sudo ufw deny from 203.0.113.0/24
```

```
Rule added
```

Block Incoming Connections to a Network Interface

To block incoming connections from a specific IP address to a specific network interface, run the following command, replacing the highlighted IP address with the IP address you want to block:

```
sudo ufw deny in on eth0 from 203.0.113.100
```

```
Rule added
```

The in parameter tells ufw to apply the rule only for **incoming** connections, and the on eth0 parameter specifies that the rule applies only for the eth0 interface. This might be useful if you have a system with several network interfaces (including virtual ones) and you need to block external access to some of these interfaces, but not all.

Allow an IP Address

To allow all network connections that originate from a specific IP address, run the following command, replacing the highlighted IP address with the IP address that you want to allow access:

```
sudo ufw allow from 203.0.113.101
```

Rule added
If you run sudo ufw status now, you'll see similar to this, showing the word ALLOW next to the IP address you just added.

```
Status: active

To                         Action      From
--                         ------      ----
...
Anywhere                   ALLOW       203.0.113.101
```

You can also allow connections from a whole subnet by providing the corresponding subnet mask for a host, such as 203.0.113.0/24.

sudo ufw default deny incoming

```
┌──(root💀 kali)-[~]
└─# sudo ufw default allow incoming
Default incoming policy changed to 'allow'
(be sure to update your rules accordingly)
```

sudo ufw default allow outgoing

```
┌──(root💀 kali)-[~]
└─# sudo ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
```

sudo ufw allow ssh

Allow SSH

When working with remote servers, you'll want to make sure that the SSH port is open to connections so that you are able to log in to your server remotely.

The following command will enable the OpenSSH UFW application profile and allow all connections to the default SSH port on the server:

- sudo ufw allow OpenSSH
-

Rule added
Rule added (v6)
Although less user-friendly, an alternative syntax is to specify the exact port number of the SSH service, which is typically set to 22 by default:

- sudo ufw allow 22
-

Rule added
Rule added (v6)

Allow Incoming SSH from Specific IP Address or Subnet

To allow incoming connections from a specific IP address or subnet, you'll include a from directive to define the source of the connection. This will require that you also specify the destination address with a to parameter. To lock this rule to SSH only, you'll limit the proto (protocol) to tcp and then use the port parameter and set it to 22, SSH's default port.

The following command will allow only SSH connections coming from the IP address 203.0.113.103:

```
sudo ufw allow from 203.0.113.103 proto tcp to any port 22
```

Rule added
You can also use a subnet address as from parameter to allow incoming SSH connections from an entire network:

```
sudo ufw allow from 203.0.113.0/24 proto tcp to any port 22
```

Rule added

Allow Incoming Rsync from Specific IP Address or Subnet

The Rsync program, which runs on port 873, can be used to transfer files from one computer to another.

To allow incoming rsync connections from a specific IP address or subnet, use the from parameter to specify the source IP address and the port parameter to set the destination port 873.
The following command will allow only Rsync connections coming from the IP address 203.0.113.103:

```
sudo ufw allow from 203.0.113.103 to any port 873
```

Rule added
To allow the entire 203.0.113.0/24 subnet to be able to rsync to your server, run:

```
sudo ufw allow from 203.0.113.0/24 to any port 873
```

Rule added

Allow Nginx HTTP / HTTPS

Upon installation, the Nginx web server sets up a few different UFW profiles within the server. Once you have Nginx installed and enabled as a service, run the following command to identify which profiles are available:

```
sudo ufw app list | grep Nginx
```

Nginx Full
Nginx HTTP
Nginx HTTPS

To enable both HTTP and HTTPS traffic, choose Nginx Full. Otherwise, choose either Nginx HTTP to allow only HTTP or Nginx HTTPS to allow only HTTPS.

The following command will allow both HTTP and HTTPS traffic on the server (ports 80 and 443):

```
sudo ufw allow "Nginx Full"
```

```
Rule added
Rule added (v6)
```

Allow Apache HTTP / HTTPS

Upon installation, the Apache web server sets up a few different UFW profiles within the server. Once you have Apache installed and enabled as a service, run the following command to identify which profiles are available:

```
sudo ufw app list | grep Apache
```

```
  Apache
  Apache Full
  Apache Secure
```

To enable both HTTP and HTTPS traffic, choose Apache Full. Otherwise, choose either Apache for HTTP or Apache Secure for HTTPS.

The following command will allow both HTTP and HTTPS traffic on the server (ports 80 and 443):

```
sudo ufw allow "Nginx Full"
```

```
Rule added
Rule added (v6)
```

Allow All Incoming HTTP (port 80)

Web servers, such as Apache and Nginx, typically listen for HTTP requests on port 80. If your default policy for incoming traffic is set to drop or deny, you'll need to create a UFW rule to allow external access on port 80. You can use either the port number or the service name (http) as a parameter to this command.

To allow all incoming HTTP (port 80) connections, run:

```
sudo ufw allow http
```

Rule added
Rule added (v6)
An alternative syntax is to specify the port number of the HTTP service:

```
sudo ufw allow 80
```

Rule added
Rule added (v6)

Allow All Incoming HTTPS (port 443)

HTTPS typically runs on port 443. If your default policy for incoming traffic is set to drop or deny, you'll need to create a UFW rule to allow external access on port 443. You can use either the port number or the service name (https) as a parameter to this command.

To allow all incoming HTTPS (port 443) connections, run:

```
sudo ufw allow https
```

Rule added
Rule added (v6)
An alternative syntax is to specify the port number of the HTTPS service:

```
sudo ufw allow 443
```

Rule added
Rule added (v6)

Allow All Incoming HTTP and HTTPS

If you want to allow both HTTP and HTTPS traffic, you can create a single rule that allows both ports. This usage requires that you also define the protocol with the proto parameter, which in this case should be set to tcp.

To allow all incoming HTTP and HTTPS (ports 80 and 443) connections, run:

```
sudo ufw allow proto tcp from any to any port 80,443
```

Rule added
Rule added (v6)

Allow MySQL Connection from Specific IP Address or Subnet

MySQL listens for client connections on port 3306. If your MySQL database server is being used by a client on a remote server, you'll need to create a UFW rule to allow that access.

To allow incoming MySQL connections from a specific IP address or subnet, use the from parameter to specify the source IP address and the port parameter to set the destination port 3306.

The following command will allow the IP address 203.0.113.103 to connect to the server's MySQL port:

```
sudo ufw allow from 203.0.113.103 to any port 3306
```

Rule added
To allow the entire 203.0.113.0/24 subnet to be able to connect to your MySQL server, run:

```
sudo ufw allow from 203.0.113.0/24 to any port 3306
```

Rule added

Allow PostgreSQL Connection from Specific IP Address or Subnet

PostgreSQL listens for client connections on port 5432. If your PostgreSQL database server is being used by a client on a remote server, you need to be sure to allow that traffic.

To allow incoming PostgreSQL connections from a specific IP address or subnet, specify the source with the from parameter, and set the port to 5432:

```
sudo ufw allow from 203.0.113.103 to any port 5432
```

Rule added
To allow the entire 203.0.113.0/24 subnet to be able to connect to your PostgreSQL server, run:

```
sudo ufw allow from 203.0.113.0/24 to any port 5432
```

Rule added

Block Outgoing SMTP Mail

Mail servers, such as Sendmail and Postfix, typically use port 25 for SMTP traffic. If your server shouldn't be sending outgoing mail, you may want to block that kind of traffic. To block outgoing SMTP connections, run:

```
sudo ufw deny out 25
```

Rule added
Rule added (v6)
This configures your firewall to **drop** all outgoing traffic on port 25. If you need to reject outgoing connections on a different port number, you can repeat this command and replace 25 with the port number you want to block.

```
┌──(root💀 kali)-[~]
└─# sudo ufw allow ssh
Rules updated
Rules updated (v6)
```

sudo ufw allow http

```
┌──(root💀 kali)-[~]
└─# sudo ufw allow http
Rules updated
Rules updated (v6)
```

sudo ufw allow https

```
┌──(root💀 kali)-[~]
└─# sudo ufw allow https
Rules updated
Rules updated (v6)
```

sudo ufw allow 443

```
┌──(root💀 kali)-[~]
└─# sudo ufw allow 443
Skipping adding existing rule
Skipping adding existing rule (v6)
```

ufw status verbose

```
┌──(root💀 kali)-[~]
└─# ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
24/udp                     DENY IN     Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)
24/udp (v6)                DENY IN     Anywhere (v6)
```

systemctl start ufw

```
┌──(root💀 kali)-[~]
└─# systemctl status firewalld                                                  252 ✗
● firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/lib/systemd/system/firewalld.service; disabled; vendor preset: disabled)
     Active: inactive (dead)
       Docs: man:firewalld(1)

┌──(root💀 kali)-[~]
└─# systemctl unmask --now firewalld                                              3 ✗

┌──(root💀 kali)-[~]
└─# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/lib/systemd/system/firewalld.service; disabled; vendor preset: disabled)
     Active: inactive (dead)
       Docs: man:firewalld(1)

┌──(root💀 kali)-[~]
└─# systemctl enable firewalld                                                    3 ✗
Created symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service → /lib/systemd/system/firewalld.service
.
Created symlink /etc/systemd/system/multi-user.target.wants/firewalld.service → /lib/systemd/system/firewalld.service
.

┌──(root💀 kali)-[~]
└─# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/lib/systemd/system/firewalld.service; enabled; vendor preset: disabled)
     Active: inactive (dead)
       Docs: man:firewalld(1)

┌──(root💀 kali)-[~]
└─# systemctl start firewalld                                                     3 ✗

┌──(root💀 kali)-[~]
└─# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/lib/systemd/system/firewalld.service; enabled; vendor preset: disabled)
     Active: active (running) since Wed 2021-11-24 11:14:38 EST; 1s ago
       Docs: man:firewalld(1)
   Main PID: 3347 (firewalld)
      Tasks: 2 (limit: 4633)
     Memory: 23.3M
        CPU: 634ms
     CGroup: /system.slice/firewalld.service
             └─3347 /usr/bin/python3 /usr/sbin/firewalld --nofork --nopid

Nov 24 11:14:38 kali systemd[1]: Starting firewalld - dynamic firewall daemon...
Nov 24 11:14:38 kali systemd[1]: Started firewalld - dynamic firewall daemon.
```

```
┌──(root💀 kali)-[~]
└─# systemctl start ufw
```

sudo ufw allow proto tcp from any to any port 80,443

```
┌──(root💀 kali)-[~]
└─# sudo ufw allow proto tcp from any to any port 80,443
Rule added
Rule added (v6)
```

```
┌──(root💀 kali)-[~]
└─# ufw allow 22/tcp
Rule added
Rule added (v6)
```

```
┌──(root💀 kali)-[~]
└─# ufw deny 24/udp
Rule added
Rule added (v6)
```

grep VERSION /etc/os-release

```
┌──(root💀 kali)-[~]
└─# grep VERSION /etc/os-release
VERSION="2021.3"
VERSION_ID="2021.3"
VERSION_CODENAME="kali-rolling"
```

firewall-cmd --get-services

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --get-services
RH-Satellite-6 RH-Satellite-6-capsule amanda-client amanda-k5-client amqp amqps apcupsd audit bacula bacula-client bb
 bgp bitcoin bitcoin-rpc bitcoin-testnet bitcoin-testnet-rpc bittorrent-lsd ceph ceph-mon cfengine cockpit collectd c
ondor-collector ctdb dhcp dhcpv6 dhcpv6-client distcc dns dns-over-tls docker-registry docker-swarm dropbox-lansync e
lasticsearch etcd-client etcd-server finger foreman foreman-proxy freeipa-4 freeipa-ldap freeipa-ldaps freeipa-replic
ation freeipa-trust ftp galera ganglia-client ganglia-master git grafana gre high-availability http https imap imaps
ipp ipp-client ipsec irc ircs iscsi-target isns jenkins kadmin kdeconnect kerberos kibana klogin kpasswd kprop kshell
 kube-api kube-apiserver kube-control-plane kube-controller-manager kube-scheduler kubelet-worker ldap ldaps libvirt
libvirt-tls lightning-network llmnr managesieve matrix mdns memcache minidlna mongodb mosh mountd mqtt mqtt-tls ms-wb
t mssql murmur mysql nbd netbios-ns nfs nfs3 nmea-0183 nrpe ntp nut openvpn ovirt-imageio ovirt-storageconsole ovirt-
vmconsole plex pmcd pmproxy pmwebapi pmwebapis pop3 pop3s postgresql privoxy prometheus proxy-dhcp ptp pulseaudio pup
petmaster quassel radius rdp redis redis-sentinel rpc-bind rquotad rsh rsyncd rtsp salt-master samba samba-client sam
ba-dc sane sip sips slp smtp smtp-submission smtps snmp snmptrap spideroak-lansync spotify-sync squid ssdp ssh steam-
streaming svdrp svn syncthing syncthing-gui synergy syslog syslog-tls telnet tentacle tftp tile38 tinc tor-socks tran
smission-client upnp-client vdsm vnc-server wbem-http wbem-https wireguard wsman wsmans xdmcp xmpp-bosh xmpp-client x
mpp-local xmpp-server zabbix-agent zabbix-server
```

firewall-cmd --zone=external --list-all

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --get-active-zones
public
  interfaces: eth0
```

firewall-cmd --set-default=external

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --set-default=external
success
```

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --list-all
Command 'firewall-cmd' not found, but can be installed with:
apt install firewalld
Do you want to install it? (N/y)y
apt install firewalld
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ipset libipset13 libnftables1 nftables python3-cap-ng python3-firewall python3-nftables
The following NEW packages will be installed:
  firewalld ipset libipset13 python3-cap-ng python3-firewall python3-nftables
The following packages will be upgraded:
  libnftables1 nftables
2 upgraded, 6 newly installed, 0 to remove and 894 not upgraded.
Need to get 994 kB of archives.
After this operation, 4,337 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 nftables amd64 1.0.0-1 [70.0 kB]
Get:2 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 libnftables1 amd64 1.0.0-1 [277 kB]
Get:3 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 python3-nftables amd64 1.0.0-1 [17.0 kB]
Get:4 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 python3-firewall all 1.0.2-1 [132 kB]
Get:5 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 firewalld all 1.0.2-1 [360 kB]
Get:6 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 libipset13 amd64 7.10-1 [68.8 kB]
Get:7 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 ipset amd64 7.10-1 [47.8 kB]
Get:8 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 python3-cap-ng amd64 0.7.9-2.2+b1 [20.6 kB]
Fetched 994 kB in 4s (221 kB/s)
(Reading database ... 267973 files and directories currently installed.)
Preparing to unpack .../0-nftables_1.0.0-1_amd64.deb ...
Unpacking nftables (1.0.0-1) over (0.9.8-3.1) ...
Preparing to unpack .../1-libnftables1_1.0.0-1_amd64.deb ...
Unpacking libnftables1:amd64 (1.0.0-1) over (0.9.8-3.1) ...
Selecting previously unselected package python3-nftables.
Preparing to unpack .../2-python3-nftables_1.0.0-1_amd64.deb ...
Unpacking python3-nftables (1.0.0-1) ...
Selecting previously unselected package python3-firewall.
Preparing to unpack .../3-python3-firewall_1.0.2-1_all.deb ...
Unpacking python3-firewall (1.0.2-1) ...
Selecting previously unselected package firewalld.
Preparing to unpack .../4-firewalld_1.0.2-1_all.deb ...
Unpacking firewalld (1.0.2-1) ...
Selecting previously unselected package libipset13:amd64.
Preparing to unpack .../5-libipset13_7.10-1_amd64.deb ...
Unpacking libipset13:amd64 (7.10-1) ...
Selecting previously unselected package ipset.
Preparing to unpack .../6-ipset_7.10-1_amd64.deb ...
Unpacking ipset (7.10-1) ...
Selecting previously unselected package python3-cap-ng.
Preparing to unpack .../7-python3-cap-ng_0.7.9-2.2+b1_amd64.deb ...
Unpacking python3-cap-ng (0.7.9-2.2+b1) ...
Setting up libnftables1:amd64 (1.0.0-1) ...
```

```
┌──(root💀 kali)-[~]
└─# firewalld -h                                                                    2 ×
usage: firewalld [-h] [--debug [level]] [--debug-gc] [--nofork] [--nopid] [--system-config path]
                 [--default-config path] [--log-file path]

optional arguments:
  -h, --help            show this help message and exit
  --debug [level]       Enable logging of debug messages. Additional argument in range 1..10 can be used to specify
                        log level.
  --debug-gc            Turn on garbage collector leak information. The collector runs every 10 seconds and if
                        there are leaks, it prints information about the leaks.
  --nofork              Turn off daemon forking, run as a foreground process.
  --nopid               Disable writing pid file and don't check for existing server process.
  --system-config path  Path to firewalld system configuration
  --default-config path
                        Path to firewalld default configuration
  --log-file path       Path to firewalld log file
```

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --zone=external --list-all
external
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh
  ports:
  protocols:
  forward: yes
  masquerade: yes
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

IP TABLES OPERATIONS

```
┌──(root💀 kali)-[~]
└─# iptables -h
iptables v1.8.7

Usage: iptables -[ACD] chain rule-specification [options]
       iptables -I chain [rulenum] rule-specification [options]
       iptables -R chain rulenum rule-specification [options]
       iptables -D chain rulenum [options]
       iptables -[LS] [chain [rulenum]] [options]
       iptables -[FZ] [chain] [options]
       iptables -[NX] chain
       iptables -E old-chain-name new-chain-name
       iptables -P chain target [options]
       iptables -h (print this help information)

Commands:
Either long or short options are allowed.
  --append  -A chain            Append to chain
  --check   -C chain            Check for the existence of a rule
  --delete  -D chain            Delete matching rule from chain
  --delete  -D chain rulenum
                                Delete rule rulenum (1 = first) from chain
  --insert  -I chain [rulenum]
                                Insert in chain as rulenum (default 1=first)
  --replace -R chain rulenum
                                Replace rule rulenum (1 = first) in chain
  --list    -L [chain [rulenum]]
                                List the rules in a chain or all chains
  --list-rules -S [chain [rulenum]]
                                Print the rules in a chain or all chains
  --flush   -F [chain]          Delete all rules in  chain or all chains
  --zero    -Z [chain [rulenum]]
                                Zero counters in chain or all chains
  --new     -N chain            Create a new user-defined chain
  --delete-chain
            -X [chain]          Delete a user-defined chain
  --policy  -P chain target
                                Change policy on chain to target
  --rename-chain
            -E old-chain new-chain
                                Change chain name, (moving any references)
Options:
    --ipv4      -4              Nothing (line is ignored by ip6tables-restore)
    --ipv6      -6              Error (line is ignored by iptables-restore)
[!] --proto     -p proto        protocol: by number or name, eg. `tcp'
[!] --source    -s address[/mask][...]
                                source specification
[!] --destination -d address[/mask][...]
                                destination specification
[!] --in-interface -i input name[+]
                                network interface name ([+] for wildcard)
 --jump -j target
```

```
  --delete  -D chain rulenum
                                Delete rule rulenum (1 = first) from chain
  --insert  -I chain [rulenum]
                                Insert in chain as rulenum (default 1=first)
  --replace -R chain rulenum
                                Replace rule rulenum (1 = first) in chain
  --list    -L [chain [rulenum]]
                                List the rules in a chain or all chains
  --list-rules -S [chain [rulenum]]
                                Print the rules in a chain or all chains
  --flush   -F [chain]          Delete all rules in  chain or all chains
  --zero    -Z [chain [rulenum]]
                                Zero counters in chain or all chains
  --new     -N chain            Create a new user-defined chain
  --delete-chain
            -X [chain]          Delete a user-defined chain
  --policy  -P chain target
                                Change policy on chain to target
  --rename-chain
            -E old-chain new-chain
                                Change chain name, (moving any references)
Options:
    --ipv4      -4              Nothing (line is ignored by ip6tables-restore)
    --ipv6      -6              Error (line is ignored by iptables-restore)
[!] --proto     -p proto        protocol: by number or name, eg. `tcp'
[!] --source    -s address[/mask][...]
                                source specification
[!] --destination -d address[/mask][...]
                                destination specification
[!] --in-interface -i input name[+]
                                network interface name ([+] for wildcard)
 --jump -j target
                                target for rule (may load target extension)
  --goto      -g chain
                               jump to chain with no return
  --match       -m match
                                extended match (may load extension)
  --numeric     -n              numeric output of addresses and ports
[!] --out-interface -o output name[+]
                                network interface name ([+] for wildcard)
  --table       -t table        table to manipulate (default: `filter')
  --verbose     -v              verbose mode
  --wait        -w [seconds]    maximum wait to acquire xtables lock before give up
  --wait-interval -W [usecs]    wait time to try to acquire xtables lock
                                default is 1 second
  --line-numbers                print line numbers when listing
  --exact       -x              expand numbers (display exact values)
[!] --fragment  -f              match second or further fragments only
  --modprobe=<command>          try to insert modules using this command
  --set-counters PKTS BYTES     set the counter during insert/append
[!] --version   -V              print package version.
```

```
┌──(root💀 kali)-[~]
└─# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

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

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --list-all-zones
block
  target: %%REJECT%%
  icmp-block-inversion: no
  interfaces:
  sources:
  services:
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

dmz
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

drop
  target: DROP
  icmp-block-inversion: no
  interfaces:
  sources:
  services:
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
        rule priority="32767" reject

public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

trusted
  target: ACCEPT
  icmp-block-inversion: no
  interfaces:
  sources:
  services:
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

work
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

```
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: ssh
  ports:
  protocols:
  forward: yes
  masquerade: yes
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

home
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client mdns samba-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

internal
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client mdns samba-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

nm-shared
  target: ACCEPT
  icmp-block-inversion: no
  interfaces:
  sources:
```

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: dhcpv6-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --list-all
FirewallD is not running

┌──(root💀 kali)-[~]
└─# firewall-cmd --get-active-zones
FirewallD is not running
```

```
┌──(root💀 kali)-[~]
└─# sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 80/tcp                     ALLOW IN    Anywhere
[ 3] 443                        ALLOW IN    Anywhere
[ 4] 80,443/tcp                 ALLOW IN    Anywhere
[ 5] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 6] 80/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 7] 443 (v6)                   ALLOW IN    Anywhere (v6)
[ 8] 80,443/tcp (v6)            ALLOW IN    Anywhere (v6)
```

```
┌──(root💀 kali)-[~]
└─# sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 80/tcp                     ALLOW IN    Anywhere
[ 3] 443                        ALLOW IN    Anywhere
[ 4] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 5] 80/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 6] 443 (v6)                   ALLOW IN    Anywhere (v6)
```

```
┌──(root💀 kali)-[~]
└─# ufw enable
Firewall is active and enabled on system startup
```

```
┌──(root💀 kali)-[~]
└─# sudo ufw status numbered
Status: inactive
```

```
┌──(root💀 kali)-[~]
└─# ufw status verbose
Status: inactive
```

```
┌──(root💀 kali)-[~]
└─# sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
```

```
┌──(root💀 kali)-[~]
└─# ufw enable
Firewall is active and enabled on system startup
```

```
┌──(root💀 kali)-[~]
└─# apt-get install firewall-applet
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  firewall-config gir1.2-notify-0.7 python3-dbus.mainloop.pyqt5
The following NEW packages will be installed:
  firewall-applet firewall-config gir1.2-notify-0.7 python3-dbus.mainloop.pyqt5
0 upgraded, 4 newly installed, 0 to remove and 894 not upgraded.
Need to get 258 kB of archives.
After this operation, 1,645 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 firewall-config all 1.0.2-1 [82.0 kB]
Get:2 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 gir1.2-notify-0.7 amd64 0.7.9-3 [10.7 kB]
Get:3 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 python3-dbus.mainloop.pyqt5 amd64 5.15.6+dfsg-1+b
1 [111 kB]
Get:4 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 firewall-applet all 1.0.2-1 [54.2 kB]
Fetched 258 kB in 5s (57.2 kB/s)
Selecting previously unselected package firewall-config.
(Reading database ... 268418 files and directories currently installed.)
Preparing to unpack .../firewall-config_1.0.2-1_all.deb ...
Unpacking firewall-config (1.0.2-1) ...
Selecting previously unselected package gir1.2-notify-0.7:amd64.
Preparing to unpack .../gir1.2-notify-0.7_0.7.9-3_amd64.deb ...
Unpacking gir1.2-notify-0.7:amd64 (0.7.9-3) ...
Selecting previously unselected package python3-dbus.mainloop.pyqt5.
Preparing to unpack .../python3-dbus.mainloop.pyqt5_5.15.6+dfsg-1+b1_amd64.deb ...
Unpacking python3-dbus.mainloop.pyqt5 (5.15.6+dfsg-1+b1) ...
Selecting previously unselected package firewall-applet.
Preparing to unpack .../firewall-applet_1.0.2-1_all.deb ...
Unpacking firewall-applet (1.0.2-1) ...
Setting up python3-dbus.mainloop.pyqt5 (5.15.6+dfsg-1+b1) ...
Setting up firewall-config (1.0.2-1) ...
Setting up gir1.2-notify-0.7:amd64 (0.7.9-3) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for mailcap (3.70) ...
Processing triggers for kali-menu (2021.3.3) ...
Processing triggers for desktop-file-utils (0.26-1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for libglib2.0-0:amd64 (2.68.4-1) ...
Setting up firewall-applet (1.0.2-1) ...
```

```
┌──(root💀 kali)-[~]
└─# apt-get install firewalld
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
firewalld is already the newest version (1.0.2-1).
0 upgraded, 0 newly installed, 0 to remove and 894 not upgraded.
```

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --permanent --new-zone=liconfig
success

┌──(root💀 kali)-[~]
└─# firewall-cmd --zone=public --list-interfaces


┌──(root💀 kali)-[~]
└─# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

┌──(root💀 kali)-[~]
└─# iptables -L -n -v
Chain INPUT (policy ACCEPT 220 packets, 13536 bytes)
 pkts bytes target     prot opt in     out     source               destination

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

Chain OUTPUT (policy ACCEPT 4 packets, 228 bytes)
 pkts bytes target     prot opt in     out     source               destination

┌──(root💀 kali)-[~]
└─# iptables -A input -j DROP
iptables: No chain/target/match by that name.
```

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --get-active-zones
FirewallD is not running
```

```
2 upgraded, 6 newly installed, 0 to remove and 894 not upgraded.
Need to get 994 kB of archives.
After this operation, 4,337 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 nftables amd64 1.0.0-1 [70.0 kB]
Get:2 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 libnftables1 amd64 1.0.0-1 [277 kB]
Get:3 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 python3-nftables amd64 1.0.0-1 [17.0 kB]
Get:4 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 python3-firewall all 1.0.2-1 [132 kB]
Get:5 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 firewalld all 1.0.2-1 [360 kB]
Get:6 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 libipset13 amd64 7.10-1 [68.8 kB]
Get:7 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 ipset amd64 7.10-1 [47.8 kB]
Get:8 https://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 python3-cap-ng amd64 0.7.9-2.2+b1 [20.6 kB]
Fetched 994 kB in 4s (221 kB/s)
(Reading database ... 267973 files and directories currently installed.)
Preparing to unpack .../0-nftables_1.0.0-1_amd64.deb ...
Unpacking nftables (1.0.0-1) over (0.9.8-3.1) ...
Preparing to unpack .../1-libnftables1_1.0.0-1_amd64.deb ...
Unpacking libnftables1:amd64 (1.0.0-1) over (0.9.8-3.1) ...
Selecting previously unselected package python3-nftables.
Preparing to unpack .../2-python3-nftables_1.0.0-1_amd64.deb ...
Unpacking python3-nftables (1.0.0-1) ...
Selecting previously unselected package python3-firewall.
Preparing to unpack .../3-python3-firewall_1.0.2-1_all.deb ...
Unpacking python3-firewall (1.0.2-1) ...
Selecting previously unselected package firewalld.
Preparing to unpack .../4-firewalld_1.0.2-1_all.deb ...
Unpacking firewalld (1.0.2-1) ...
Selecting previously unselected package libipset13:amd64.
Preparing to unpack .../5-libipset13_7.10-1_amd64.deb ...
Unpacking libipset13:amd64 (7.10-1) ...
Selecting previously unselected package ipset.
Preparing to unpack .../6-ipset_7.10-1_amd64.deb ...
Unpacking ipset (7.10-1) ...
Selecting previously unselected package python3-cap-ng.
Preparing to unpack .../7-python3-cap-ng_0.7.9-2.2+b1_amd64.deb ...
Unpacking python3-cap-ng (0.7.9-2.2+b1) ...
Setting up libnftables1:amd64 (1.0.0-1) ...
Setting up nftables (1.0.0-1) ...
Setting up python3-cap-ng (0.7.9-2.2+b1) ...
Setting up python3-firewall (1.0.2-1) ...
Setting up libipset13:amd64 (7.10-1) ...
Setting up ipset (7.10-1) ...
Setting up python3-nftables (1.0.0-1) ...
Setting up firewalld (1.0.2-1) ...
update-alternatives: using /usr/share/polkit-1/actions/org.fedoraproject.FirewallD1.server.policy.choice to provide /
usr/share/polkit-1/actions/org.fedoraproject.FirewallD1.policy (org.fedoraproject.FirewallD1.policy) in auto mode
firewalld.service is a disabled or a static unit not running, not starting it.
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for dbus (1.12.20-2) ...
Processing triggers for kali-menu (2021.3.3) ...
Processing triggers for libc-bin (2.31-13) ...
```

```
┌──(root💀 kali)-[~]
└─# firewall-cmd --permanent --new-zone=liconfig
success

┌──(root💀 kali)-[~]
└─# firewall-cmd --permanent --new-zone-from-file=file --name=linconfig
Failed to load zone file 'file': INVALID_NAME: 'file' is missing .xml suffix

┌──(root💀 kali)-[~]
└─# firewall-cmd --permanent --new-zone-from-public=file --name=linconfig
usage: see firewall-cmd man page
firewall-cmd: error: unrecognized arguments: --new-zone-from-public=file

┌──(root💀 kali)-[~]
└─# firewall-cmd --permanent --new-zone-from-public=public --name=linconfig
usage: see firewall-cmd man page
firewall-cmd: error: unrecognized arguments: --new-zone-from-public=public

┌──(root💀 kali)-[~]
└─# firewall-cmd --zone=public  --list-interfaces
eth0

┌──(root💀 kali)-[~]
└─# iptables -p forward drop
iptables v1.8.7 (nf_tables): unknown protocol "forward" specified
Try `iptables -h' or 'iptables --help' for more information.

┌──(root💀 kali)-[~]
└─# iptables -P forward drop
iptables: Bad policy name. Run `dmesg' for more information.

┌──(root💀 kali)-[~]
└─# iptables -P forward DROP
iptables: Bad built-in chain name.
```

```
┌──(root💀 kali)-[~]
└─# iptables -A INPUT -j ACCEPT

┌──(root💀 kali)-[~]
└─# iptables -L -n -v
Chain INPUT (policy DROP 1331 packets, 210K bytes)
 pkts bytes target     prot opt in     out     source               destination
    1   328 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0

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

Chain OUTPUT (policy ACCEPT 1281 packets, 114K bytes)
 pkts bytes target     prot opt in     out     source               destination

┌──(root💀 kali)-[~]
└─# iptables -F

┌──(root💀 kali)-[~]
└─# iptables -L -n -v
Chain INPUT (policy DROP 1525 packets, 226K bytes)
 pkts bytes target     prot opt in     out     source               destination

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

Chain OUTPUT (policy ACCEPT 1624 packets, 143K bytes)
 pkts bytes target     prot opt in     out     source               destination

┌──(root💀 kali)-[~]
└─# iptables -A INPUT -s 192.168.0.23 -j DROP

┌──(root💀 kali)-[~]
└─# iptables -L -n -v
Chain INPUT (policy DROP 2269 packets, 291K bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      *       192.168.0.23         0.0.0.0/0

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

```
┌──(root💀 kali)-[~]
└─# iptables -A INPUT -s 192.168.0.0/24 -P TCP --dport 25 -j DROP
iptables v1.8.7 (nf_tables): Cannot use -P with -A

Try `iptables -h' or 'iptables --help' for more information.

┌──(root💀 kali)-[~]
└─# iptables -A INPUT -S  192.168.0.0/24 -P TCP --dport 25 -j DROP
iptables v1.8.7 (nf_tables): Cannot use -S with -A

Try `iptables -h' or 'iptables --help' for more information.

┌──(root💀 kali)-[~]
└─# iptables -A INPUT  192.168.0.0/24 -P TCP --dport 25 -j DROP
Bad argument `192.168.0.0/24'
Try `iptables -h' or 'iptables --help' for more information.

┌──(root💀 kali)-[~]
└─# iptables -A INPUT -s  192.168.0.0/24  --dport 25 -j DROP
iptables v1.8.7 (nf_tables): unknown option "--dport"
Try `iptables -h' or 'iptables --help' for more information.

┌──(root💀 kali)-[~]
└─# iptables -A INPUT -s  192.168.0.0/24  -j DROP

┌──(root💀 kali)-[~]
└─# iptables -L -n -v
Chain INPUT (policy DROP 2950 packets, 349K bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      *       192.168.0.23         0.0.0.0/0
    0     0 DROP       all  --  *      *       192.168.0.0/24       0.0.0.0/0

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

Chain OUTPUT (policy ACCEPT 3048 packets, 263K bytes)
 pkts bytes target     prot opt in     out     source               destination

┌──(root💀 kali)-[~]
└─# iptables -A INPUT -P tcp --dport 80  -j DROP
iptables v1.8.7 (nf_tables): Cannot use -P with -A

Try `iptables -h' or 'iptables --help' for more information.
```

```
┌──(root💀 kali)-[~]
└─# iptables -L -n -v
Chain INPUT (policy DROP 3214 packets, 372K bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      *       192.168.0.23         0.0.0.0/0
    0     0 DROP       all  --  *      *       192.168.0.0/24       0.0.0.0/0
    0     0 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:80
```

Remember that you can check your current UFW ruleset with sudo ufw status or sudo ufw status verbose.

Allow Incoming Connections to a Network Interface

To allow incoming connections from a specific IP address to a specific network interface, run the following command, replacing the highlighted IP address with the IP address you want to allow:

```
sudo ufw allow in on eth0 from 203.0.113.102
```

Rule added
The in parameter tells ufw to apply the rule only for **incoming** connections, and the on eth0 parameter specifies that the rule applies only for the eth0 interface.

If you run sudo ufw status now, you'll see similar to this:

```
Status: active

To                         Action      From
--                         ------      ----
...
Anywhere on eth0           ALLOW       203.0.113.102
```

List Available Application Profiles

Upon installation, applications that rely on network communications will typically set up a UFW profile that you can use to allow connection from external addresses. This is often the same as running ufw allow from, with the advantage of providing a shortcut that abstracts the specific port numbers a service uses and provides a user-friendly nomenclature to referenced services.

To list which profiles are currently available, run the following:

```
sudo ufw app list
```

If you installed a service such as a web server or other network-dependent software and a profile was not made available within UFW, first make sure the service is enabled. For remote servers, you'll typically have OpenSSH readily available:

```
Available applications:
  OpenSSH
```

Enable Application Profile

To enable a UFW application profile, run `ufw allow` followed by the name of the application profile you want to enable, which you can obtain with a `sudo ufw app list` command. In the following example, we're enabling the OpenSSH profile, which will allow all incoming SSH connections on the default SSH port.

```
sudo ufw allow "OpenSSH"
```

```
Rule added
Rule added (v6)
```

Remember to quote profile names that consist of multiple words, such as `Nginx HTTPS`.

Disable Application Profile

To disable an application profile that you had previously set up within UFW, you'll need to remove its corresponding rule. For example, consider the following from `sudo ufw status`:

```
sudo ufw status
```

```
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx Full                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

Nginx Full (v6) ALLOW Anywhere (v6)
This indicates that the Nginx Full application profile is currently enabled, allowing any and all connections to the web server both via HTTP as well as via HTTPS. If you'd want to only allow HTTPS requests from and to your web server, you'd have to first enable the most restrictive rule, which in this case would be Nginx HTTPS, and then disable the currently active Nginx Full rule:

```
sudo ufw allow "Nginx HTTPS"
sudo ufw delete allow "Nginx Full"
```

Remember you can list all available application profiles with sudo ufw app list.

Conclusion

UFW is a powerful tool that can greatly improve the security of your servers when properly configured. This reference guide covers some common UFW rules that are often used to configure a firewall on Ubuntu.

Most of the commands in this guide can be adapted to fit different use cases and scenarios, by changing parameters such as the source IP address and/or destination port. For more detailed information about each command parameter and available modifiers, you can use the man utility to check UFW's manual:

iptables is a command line interface used to set up and maintain tables for the Netfilter firewall for IPv4, included in the Linux kernel. The firewall matches packets with rules defined in these tables and then takes the specified action on a possible match.

- *Tables* is the name for a set of chains.
- *Chain* is a collection of rules.
- *Rule* is condition used to match packet.
- *Target* is action taken when a possible rule matches. Examples of the target are ACCEPT, DROP, QUEUE.
- *Policy* is the default action taken in case of no match with the inbuilt chains and can be ACCEPT or DROP.

Syntax:
iptables --table *TABLE* -A/-C/-D... *CHAIN rule* --jump *Target*

TABLE

There are five possible tables:

- **filter:** Default used table for packet filtering. It includes chains like INPUT, OUTPUT and FORWARD.
- **nat :** Related to Network Address Translation. It includes PREROUTING and POSTROUTING chains.

- **mangle :** For specialised packet alteration. Inbuilt chains include PREROUTING and OUTPUT.
- **raw :** Configures exemptions from connection tracking. Built-in chains are PREROUTING and OUTPUT.
- **security :** Used for Mandatory Access Control

CHAINS

There are few built-in chains that are included in tables. They are:

- **INPUT :**set of rules for packets destined to localhost sockets.
- **FORWARD :**for packets routed through the device.
- **OUTPUT :**for locally generated packets, meant to be transmitted outside.
- **PREROUTING :**for modifying packets as they arrive.
- **POSTROUTING :**for modifying packets as they are leaving.

Note: User-defined chains can also be created.

OPTIONS

1. **-A, –append :** Append to the chain provided in parameters.
   Syntax:
   iptables [-t table] --append [chain] [parameters]

   Example: This command drops all the traffic coming on any port.
   iptables -t filter --append INPUT -j DROP

Syntax:
iptables [-t table] --delete [chain] [rule_number]

Example: This command deletes the rule 2 from INPUT chain.
iptables -t filter --delete INPUT 2

1. **-C, –check :**Check if a rule is present in the chain or not. It returns 0 if the rule exists and returns 1 if it does not.
   Syntax:
   iptables [-t table] --check [chain] [parameters]

   Example: This command checks whether the specified rule is present in the INPUT chain.

   iptables -t filter --check INPUT -s 192.168.1.123 -j DROP

   Output:

PARAMETERS

The parameters provided with the *iptables* command is used to match the packet and perform the specified action. The common parameters are:

1. **-p, –proto :** is the protocol that the packet follows. Possible values maybe: tcp, udp, icmp, ssh etc.
   Syntax:
   iptables [-t table] -A [chain] -p {protocol_name} [target]

   Example: This command appends a rule in the INPUT chain to drop all udp packets.
   iptables -t filter -A INPUT -p udp -j DROP

2. **-s, –source:** is used to match with the source address of the packet.
   Syntax:
   iptables [-t table] -A [chain] -s {source_address} [target]

   Example: This command appends a rule in the INPUT chain to accept all packets originating from 192.168.1.230.
   iptables -t filter -A INPUT -s 192.168.1.230 -j ACCEPT

   Output:

3. **-d, –destination :** is used to match with the destination address of the packet.
   Syntax:
   iptables [-t table] -A [chain] -d {destination_address} [target]

   Example: This command appends a rule in the OUTPUT chain to drop all packets destined for 192.168.1.123.
   iptables -t filter -A OUTPUT -d 192.168.1.123 -j DROP

4. **-i, –in-interface :** matches packets with the specified in-interface and takes the action.
   Syntax:
   iptables [-t table] -A [chain] -i {interface} [target]

   Example: This command appends a rule in the INPUT chain to drop all packets destined for wireless interface.
   iptables -t filter -A INPUT -i wlan0 -j DROP

   Output:

5. **-o, –out-interface :** matches packets with the specified out-interface.
6. **-j, –jump :** this parameter specifies the action to be taken on a match.
   Syntax:
   iptables [-t table] -A [chain] [parameters] -j {target}

   Example: This command adds a rule in the FORWARD chain to drop all packets.
   iptables -t filter -A FORWARD -j DROP

Note:

- While trying out the commands, you can remove all filtering rules and user created chains.
- `sudo iptables --flush`
- To save the iptables configuration use:
  `sudo iptables-save`
- Restoring iptables config can be done with:
  `sudo iptables-restore`
- There are other interfaces such ip6tables which are used to manage filtering tables for IPv6.

Firewalls are a vital part of network security, so it's important for a sysadmin to be familiar with how they work. If you understand firewalls, you can keep your network secure by making intelligent choices about the traffic you allow in and out.

Because "firewall" is such an exciting name, people often imagine an intricate Tron-style neon battle happening on the outskirts of a network, with packets of rogue data being set alight to protect your users' the techno fortress. In reality, a firewall is just a piece of software controlling incoming and outgoing network traffic.

Ports

A firewall is able to manage this traffic by monitoring network ports. In the world of firewalls, the term *port* doesn't refer to a physical connection like a USB, VGA, or HDMI port. For the purpose of firewalls, a *port* is an artificial construct created by the operating system to represent a pathway for a specific type of data. This system could have been called anything, like "contacts," "connections," or even "penguins," but it the creators used "ports," and that's the name that we still use today. The point is, there's nothing special about any port; they are just a way to designate an address where data transference happens.

There are a number of ports that are well-known, but even these are only conventions. For instance, you may know that HTTP traffic occurs on port 80, HTTPS traffic uses port 443, FTP uses port 21, and SSH uses port 22. When your computer transmits data to another computer, it adds a prefix to the data to indicate which port it wants to access. If the port on the receiving end is accepting data of the same protocol as the data you are sending, then the data is successfully exchanged.

You can see this process in action by going to any website. Open a web browser and navigate to example.com:80, which causes your computer to send an HTTP request to port 80 of the computer serving the example.com website. You receive a webpage in return. Web browsers don't require you to enter the port you want to access every time you navigate to a URL, however, because it's assumed that HTTP traffic accesses port 80 or 443.

You can test this process using a terminal-based web browser:

```
$ curl --connect-timeout 3 "http://example.com:80" | head -n4
<!doctype html>
<html>
<head>
  <title>Example Domain</title>
```

Using the same notation, you can force rejection by navigating to a website using a nonstandard port. Navigate to an arbitrary port, example.com:79 for instance. Your request for a webpage is declined:

```
$ curl --connect-timeout 3 "http://example.com:79"
curl: (7) Failed to connect: Network is unreachable
```

The correlation between ports and protocols are merely conventions mutually agreed upon by a standards group, and a user base. These settings can be changed on individual computers. In fact, back in the pioneer days of computing, many people felt that just changing the port number of popular services would allay an attack. Today, attacks are a lot more sophisticated. There's little value in surprising an automated port scanner by changing which port a service listens on.

Instead, a firewall governs what activity is permitted on any given port.

The firewall-cmd interface

Your infrastructure may have a server in a rack with the sole purpose of running a firewall, or you may have a firewall embedded in the router—or modem—acting as your primary gateway to the internet. You probably also have a firewall running on your personal workstation or laptop. All of these firewalls have their own configuration interface. This article covers the firewall-cmd terminal command found on most Linux distributions.

Firewall-cmd is a front-end tool for managing the firewalld daemon, which interfaces with the Linux kernel's netfilter framework. This stack probably isn't present on the embedded modems common in small- to medium-sized businesses, but it's on or available for any Linux distribution that uses systemd.

Without an active firewall, firewall-cmd has nothing to control, so the first step is to ensure that firewalld is running:

```
$ sudo systemctl enable --now firewalld
```

This command starts the firewall daemon and sets it to auto-load upon reboot.

Block (almost) everything

Common advice when configuring a firewall is to first block everything, and then open the ports you know you actually need. That means you have to know what you need, though, and sometimes figuring that out is an afternoon's job all its own.

If your organization runs its own DNS or DNS caching service, for instance, then you must remember to unblock the port (usually 53) handling DNS communication. If you

rely on SSH to configure your servers remotely, then you must not block that port. You must account for every service running on your infrastructure, and you must understand whether that service is internal-only or whether it needs to interact with the outside world.

In the case of proprietary software, there may be calls made to the outside world that you're not even aware of. If some applications react poorly to a strict firewall recently put in place, you may have to reverse engineer (or talk to the application's support line) to discover what kind of traffic it's trying to create, and why. In the open source world, this issue is less common, but it's not outside the realm of possibility, especially in the case of complex software stacks (for example, today even media players make calls out to the internet, if only to fetch album art or a track listing).

Firewall-cmd uses *zones* as presets, giving you sane defaults to choose from. Doing this saves you from having to build a firewall from scratch. Zones apply to a network interface, so on a server with two ethernet interfaces, you may have one zone governing one ethernet interface, and a different zone governing the other.

It's worth taking time to get familiar with the zones provided on your system. To see all available zones, use:

```
$ sudo firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

To see what's unblocked in a specific zone:

```
$ sudo firewall-cmd --zone work --list-all
work
  target: default
  icmp-block-inversion: no
  interfaces: ens3
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Use one of the existing zones as a starting point for your own firewall rules, or just create your own.

Create a zone

To create a new zone, use the --new-zone option.

All firewall-cmd actions persist only until the firewall or the computer running it restarts. Anything you want to be permanent must be accompanied by the --permanent flag.

For an example, create a new permanent zone called corp, and then reload the firewall rules so that your new zone activates:

```
$ sudo firewall-cmd --new-zone corp --permanent
success
$ sudo firewall-cmd --reload
```

Before assigning any network interface to this new zone, add the ssh service so you can access it remotely. Use the --permanent option to make this addition persist across reboots:

```
$ sudo firewall-cmd --zone corp --add-service ssh --permanent
```

Your new zone, called corp, is now active, rejects all but SSH traffic, and is assigned to no specific network interface. To make corp the active and default zone for the network interface you want to protect (ens3 in this example), use the --change-interface option:

```
$ firewall-cmd --change-interface ens3 \
  --zone corp --permanent
The interface is under control of NetworkManager, setting zone to 'corp'.
success
```

By making corp the default zone, all future commands are applied to corp unless the --zone option specifies a different zone. Whether you want to set corp as the default depends on whether you plan to this zone as your new primary zone. If so, the following does the job:

```
$ sudo firewall-cmd --set-default corp
```

To view the zones currently assigned to each interface, use the --get-active-zones option:

```
$ sudo firewall-cmd --get-active-zones``
corp
  interfaces: ens3
work
  interfaces: ens4
```

Add and remove services

Now that you've blocked everything but SSH, you can open the ports your network relies upon. The quick and easy way to permit traffic through your firewall is to add a predefined service.

The list of available predefined services is extensive. To view it, use:

```
$ sudo firewall-cmd --get-services
RH-Satellite-6 amanda-client amqp
amqps apcupsd audit bacula bacula-client
bgp bitcoin bitcoin-rpc bitcoin-testnet ceph
```

```
cockpit dhcp dhcpv6 dhcpv6-client distcc dns
```

[...]

Assume you need to run a webserver. First, you would install the webserver you want to use (the `httpd` package on RHEL or Fedora, `apache2` on Ubuntu and Debian). For this example, we'll use `httpd`:

```
$ sudo dnf install httpd
$ sudo systemctl --enable --now httpd
```

Then, test your webserver locally:

```
$ curl --silent localhost:80 | grep title
<title>Test Page for the Apache HTTP Server on Red Hat Enterprise Linux</title>
```

Next, attempt to connect to your webserver from an external browser. The connection fails, demonstrating that the firewall is effective:

```
$ curl --connect-timeout 3 192.168.122.206
curl: (28) Connection timed out after 3001 milliseconds
```

Unblock a service

To permit HTTP traffic through your firewall, add the `http` service:

```
$ sudo firewall-cmd --add-service http --permanent
$ sudo firewall-cmd --reload
```

Then, test from an outside source:

```
$ curl --silent 192.168.122.206 | grep title
<title>Test Page for the Apache HTTP Server on Red Hat Enterprise Linux</title>
```

Now that you know how to add a service, removing one is fairly intuitive:

```
$ sudo firewall-cmd --remove-service http --permanent
$ sudo firewall-cmd --reload
```

Add and remove ports

Sometimes, a predefined service doesn't exist, or it assumes defaults that don't match your network. Instead of adding a service, you can add a port number and protocol type directly with `--add-port`.

For instance, if you need to add the non-standard port 1622 for SSH to your custom zone (if your custom zone isn't the default zone for your commands, add the `--zone` option):

```
$ sudo firewall-cmd --add-port 1622/tcp --permanent
success
```

```
$ sudo firewall-cmd --reload
```

To remove that port, use --remove-port:

```
$ sudo firewall-cmd --remove-port 1622/tcp --permanent
success
$ sudo firewall-cmd --reload
```