**Project: COFIGURE A FIREWALL IN LINUX**

**Introduction:**

This project involves configuring a firewall on a Linux system using two popular tools: **iptables** and **UFW** (Uncomplicated Firewall). The objective is to enhance the security of the system by setting up rules to control incoming and outgoing network traffic. By doing so, you'll gain practical experience with network security measures and learn how to tailor these settings to specific environments. This project is ideal for those looking to improve their system administration and cybersecurity skills, providing both theoretical knowledge and hands-on practice in securing Linux systems.

**Motivation:**

Choosing to configure a firewall in Linux is a highly beneficial project for several reasons. Firstly, it enhances your system's security by allowing you to control incoming and outgoing network traffic. This project offers hands-on experience with Linux tools such as iptables and UFW, which are essential for any aspiring system administrator or cybersecurity professional. It helps to develop practical skills needed to secure and manage Linux systems, fostering a deeper understanding of network security principles. Additionally, it provides the opportunity to customize security settings to fit specific network environments, ensuring compliance with various security standards and regulations. By undertaking this project, you gain both theoretical knowledge and practical expertise, making you better prepared for advanced security challenges in real-world scenarios.

**Software Requirements:**

** Operating System**: A Linux distribution (e.g., Ubuntu, CentOS, Debian).

 **Permissions**: Root or sudo access to execute administrative commands.

 **Network Access**: Ability to configure and manage network interfaces.

 **Software**: **iptables** and **UFW** packages installed.

 **Basic Knowledge**: Understanding of TCP/IP networking, ports, and services.

 **Memory and Storage**: Sufficient RAM and disk space to handle firewall configurations and logging.

**Configuring a Firewall in Linux:**

**What is a Firewall:** A firewall is a security system that monitors and controls incoming and outgoing network traffic based on predetermined rules. It serves as a barrier between a trusted internal network and untrusted external networks, such as the internet. By filtering traffic, it helps protect networks from unauthorized access and cyber threats.

**Iptables:** **iptables** is a command-line utility for configuring the built-in Linux [kernel](https://phoenixnap.com/glossary/what-is-a-kernel) firewall. It enables administrators to define chained rules that control incoming and outgoing network traffic.

**How Does iptables Work?**

iptables uses rules to determine what to do with a network packet. The utility consists of the following components:

* **Tables**: Tables are files that group similar rules. A table consists of several rule **chains**.
* **Chains**: A chain is a string of **rules**. When a packet is received, iptables finds the appropriate table and filters it through the rule chain until it finds a match.
* **Rules:** A rule is a statement that defines the conditions for matching a packet, which is then sent to a **target**.
* **Targets:** A target is a decision of what to do with a packet. The packet is either accepted, dropped, or rejected.

### **Tables:** Linux firewall iptables have four default tables that manage different rule chains:

* **Filter**: The default packet filtering table. It acts as a gatekeeper that decides which packets enter and leave a network.
* [**Network Address Translation (NAT)**](https://phoenixnap.com/glossary/nat-network-address-translation): Contains NAT rules for routing packets to remote networks. It is used for packets that require alterations.
* **Mangle**: Adjusts the IP header properties of packets.
* **Raw**: Exempts packets from connection tracking.

### **Chains:** Chains are rule lists within tables. The lists control how to handle packets at different processing stages. There are different chains, each with a specific purpose:

* **INPUT**: Handles incoming packets whose destination is a local application or service. The chain is in the filter and mangle tables.
* **OUTPUT**: Manages outgoing packets generated on a local [application](https://phoenixnap.com/glossary/what-is-an-application) or service. All tables contain this chain.
* **FORWARD**: Works with packets that pass through the system from one network interface to another. The chain is in the filter, mangle, and security tables.
* **PREROUTING**: Alters packets before they are routed. The alteration happens before a routing decision. The NAT, mangle, and raw tables contain this chain.
* **POSTROUTING**: Alters packets after they are routed. The alteration happens after a routing decision. The NAT and mangle tables contain this chain.

### **Rules:** Rules are statements that define conditions for matching packets. Every rule is part of a chain and contains specific criteria, such as source or destination [IP addresses](https://phoenixnap.com/glossary/what-is-an-ip-address), [port numbers](https://phoenixnap.com/glossary/port-number), or protocols. Any packet matching a rule's conditions is forwarded to a target that determines what happens to the packet.

### **Targets:** A target is what happens after a packet matches a rule criteria. Common targets include:

* **ACCEPT**: Allows the packet to pass through the firewall.
* **DROP**: Discards the packet without informing the sender.
* **REJECT**: Discards the packet and returns an error response to the sender.
* **LOG**: Records packet information into a log file.
* **SNAT**: Stands for Source Network Address Translation. Alters the packet's source address.
* **DNAT**: Stands for Destination Network Address Translation. Changes the packet's destination address.
* **MASQUERADE**: Alters a packet's source address for dynamically assigned IPs.

## **INSTALLING IPTABLES ON LINUX**

For Debian-based distributions (such as Ubuntu), do the following:

Install iptables using the APT package manager:

**sudo apt install iptables**

To confirm that iptables is installed, run:

**iptables –version**



  To keep iptables firewall rules after reboot, install the persistent package:

## sudo apt install iptables-persistent

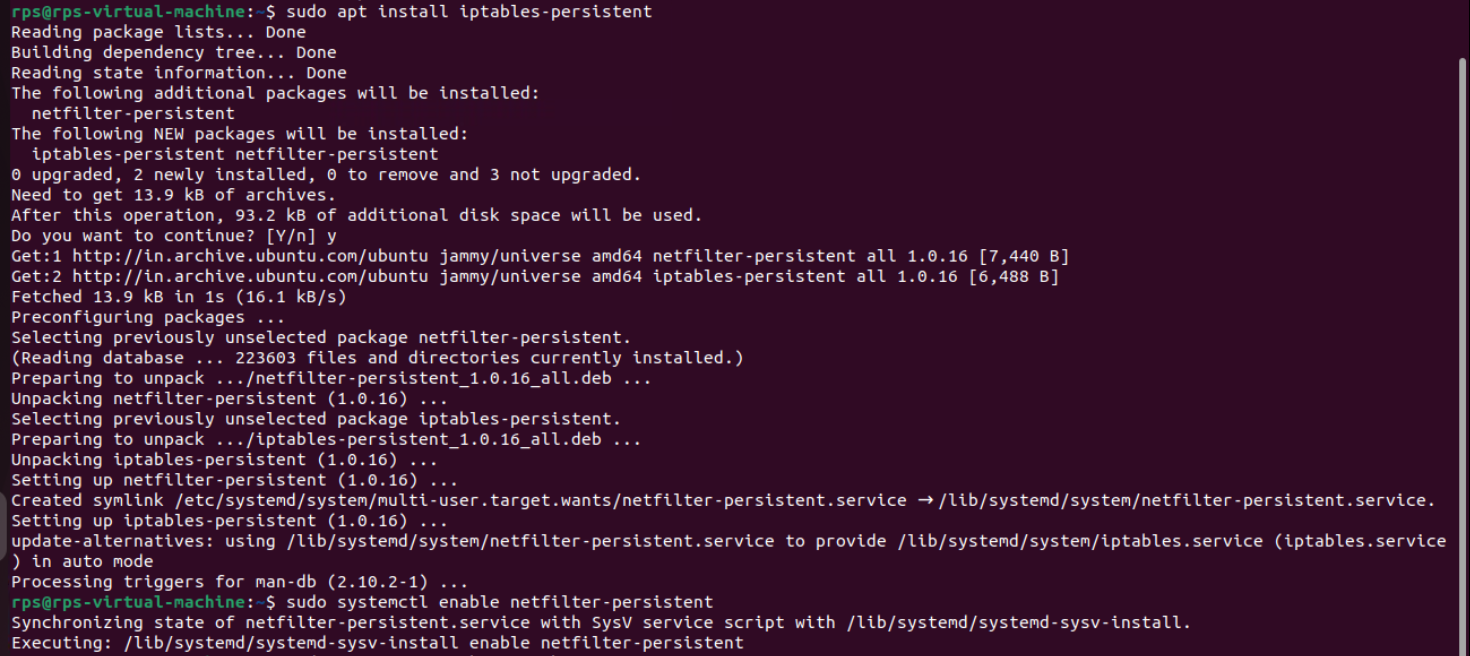
## 

## **Enable the netfilter-persistent service on restart:**

## sudo systemctl enable netfilter-persistent

## 

## **The command enables the service to start on reboot automatically.**

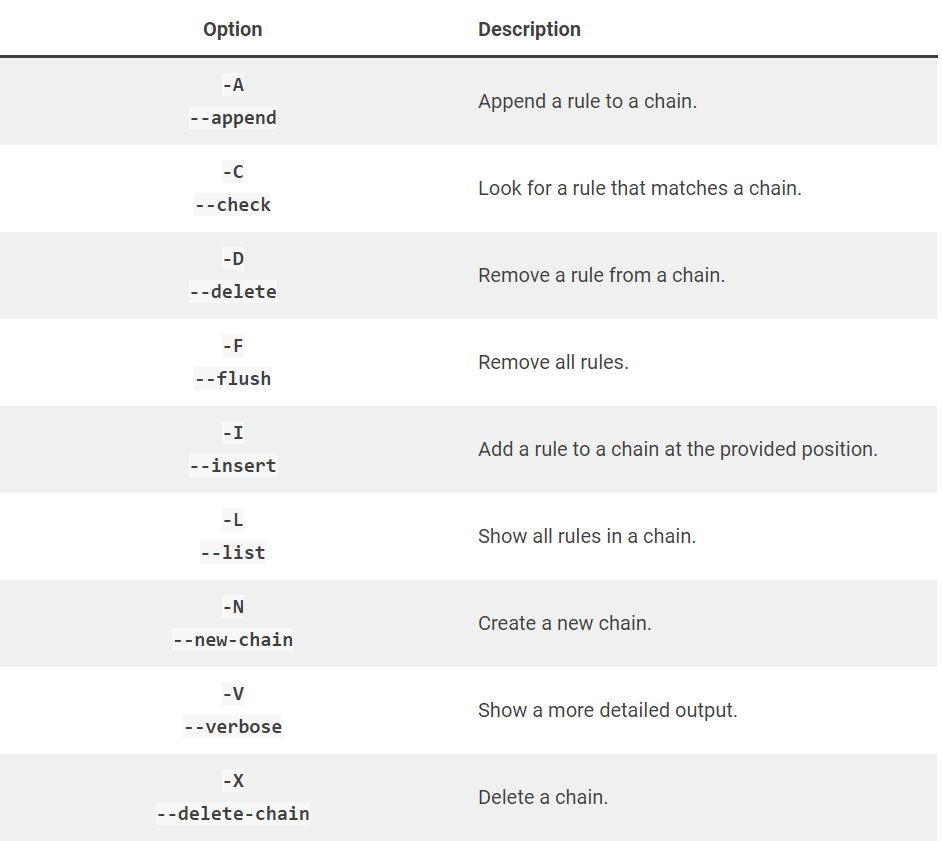


## IPTABLES SYNTAX AND OPTIONS

An iptables command looks as follows:

**iptables [options] [chain] [criteria] -j [target]**

The table below contains common iptables options:



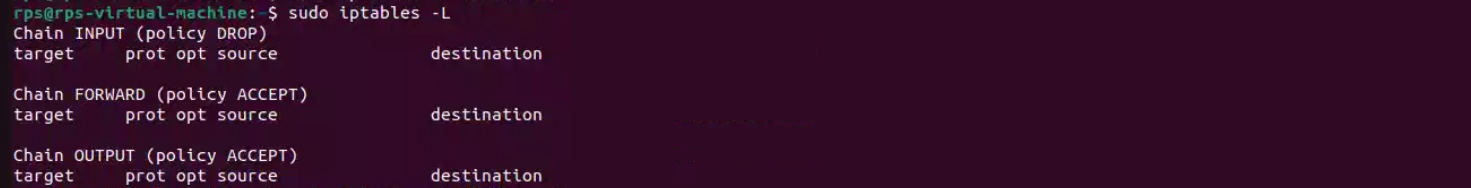
## HOW TO CONFIGURE IPTABLES ON LINUX

The **iptables** command applies actions to the filters table by default. To use a different table, add the **-t** option followed by the table name (for example, use **-t nat** for the NAT table).

**View Current Rules:**

To view the current rules, use the command with the **-L** option:

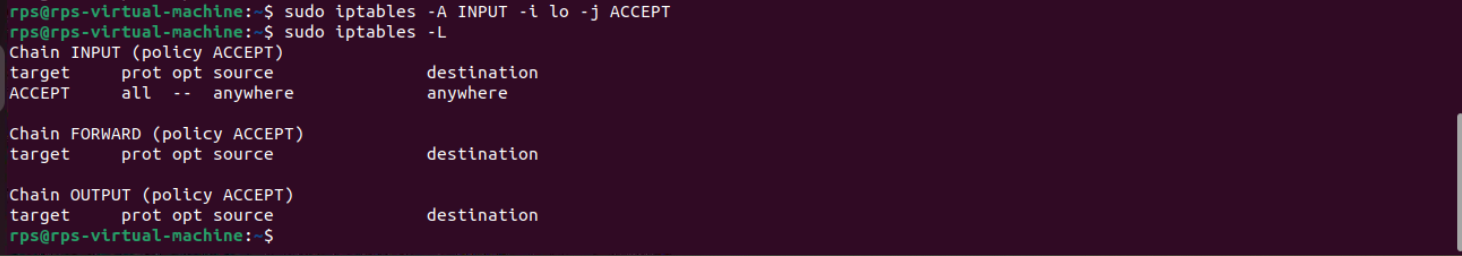
**sudo iptables -L**



The system displays the status of your chains. The output lists three chains: INPUT, FORWARD, and OUTPUT.

**Enable Loopback Traffic:** Allowing traffic from your system (localhost) is secure and allows applications to communicate with the localhost interface. Enter the following to append the INPUT chain:

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

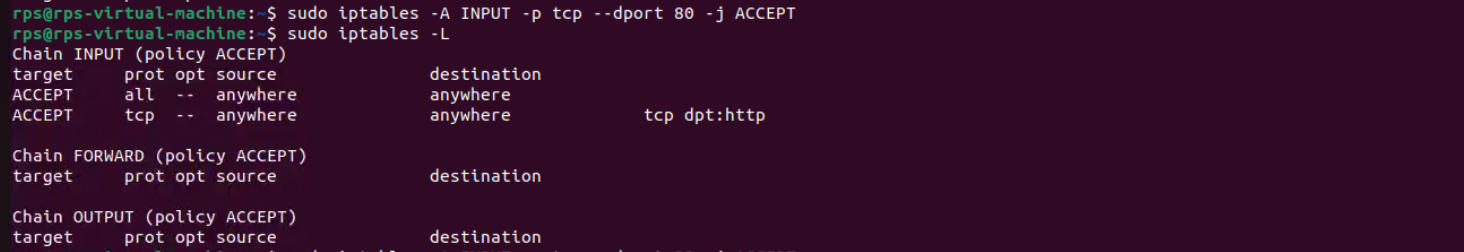


**Allow Traffic for Specific Services:**

Allow traffic on different **ports** to enable various services. See the examples below:

* Allow [HTTP](https://phoenixnap.com/glossary/http-definition) web traffic:

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



* Allow only incoming [SSH (Secure Shell)](https://phoenixnap.com/kb/what-is-ssh) traffic:

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



* Allow HTTPS traffic:

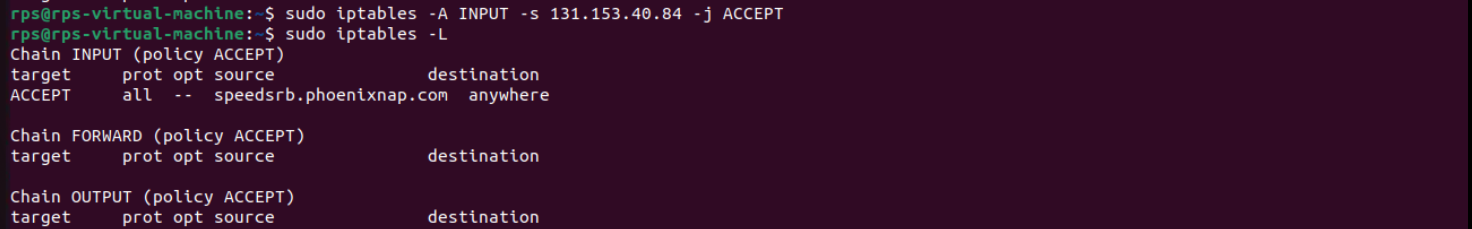
 **sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT**

### **Control Traffic by IP Address**

Use the following commands to control traffic based on an [IP address](https://phoenixnap.com/glossary/what-is-an-ip-address):

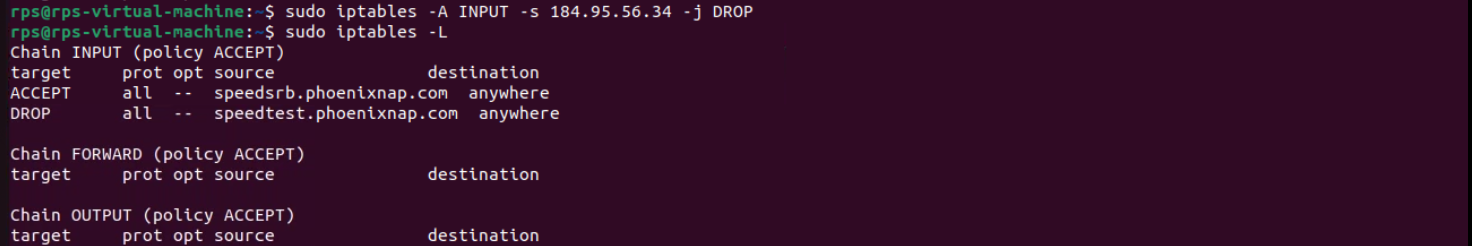
* Accept all traffic from an IP address:

**sudo iptables -A INPUT -s [IP-address] -j ACCEPT**

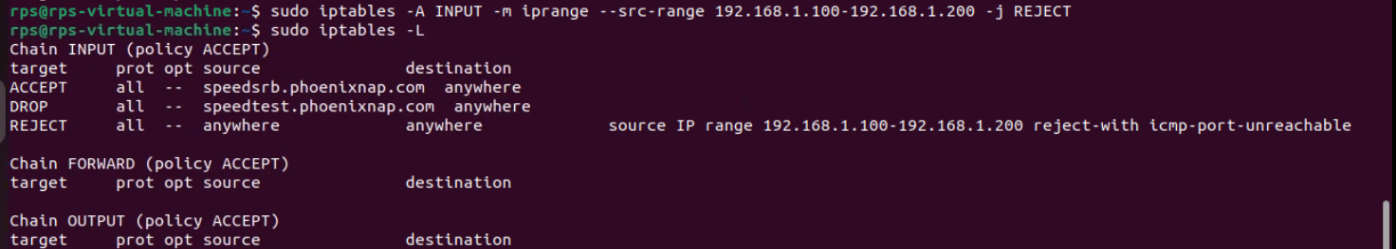


* Drop traffic from an IP address:

**sudo iptables -A INPUT -s [IP-address] -j DROP**



* Reject traffic from an IP address range:

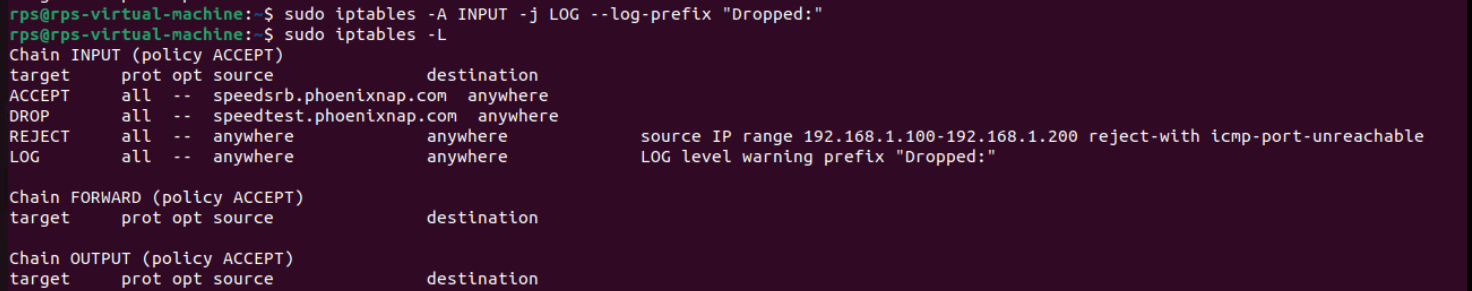
**sudo iptables -A INPUT -m iprange --src-range [IP-address-range] -j REJECT**

### **Log Dropped Packets**

To log packets, do the following:

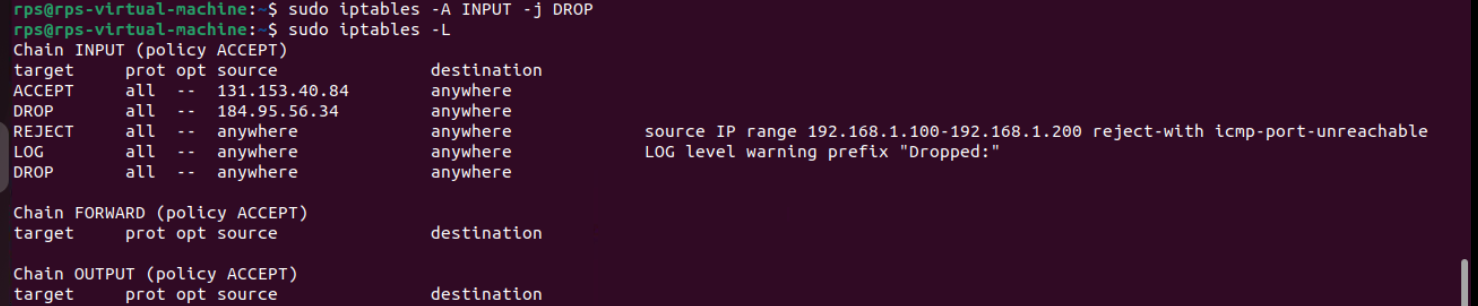
1. Use the LOG target and add a message prefix:

**sudo iptables -A INPUT -j LOG --log-prefix "Dropped: "**



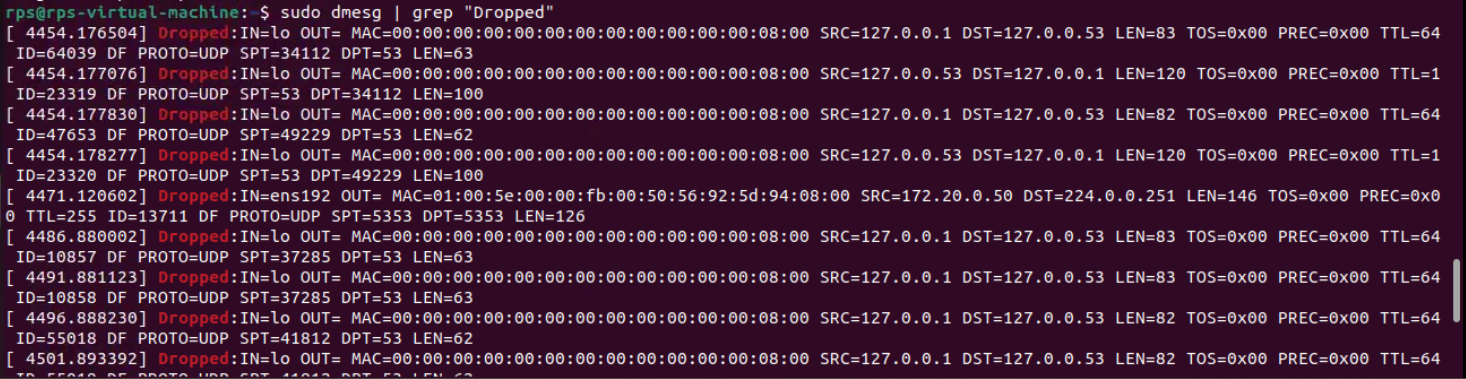
2. Add a rule to drop packets after logging:

**sudo iptables -A INPUT -j DROP**



3. To check logs, use the [dmesg command](https://phoenixnap.com/kb/dmesg-linux) to view system logs and [grep](https://phoenixnap.com/kb/grep-command-linux-unix-examples) to filter the output:

**sudo dmesg | grep "Dropped"**



Alternatively, access the syslog file using the [tail command](https://phoenixnap.com/kb/linux-tail):

**sudo tail -f /var/log/syslog | grep "Dropped packet"**

Adjust the path if messages are logged to a different location.

### **Delete a Rule**

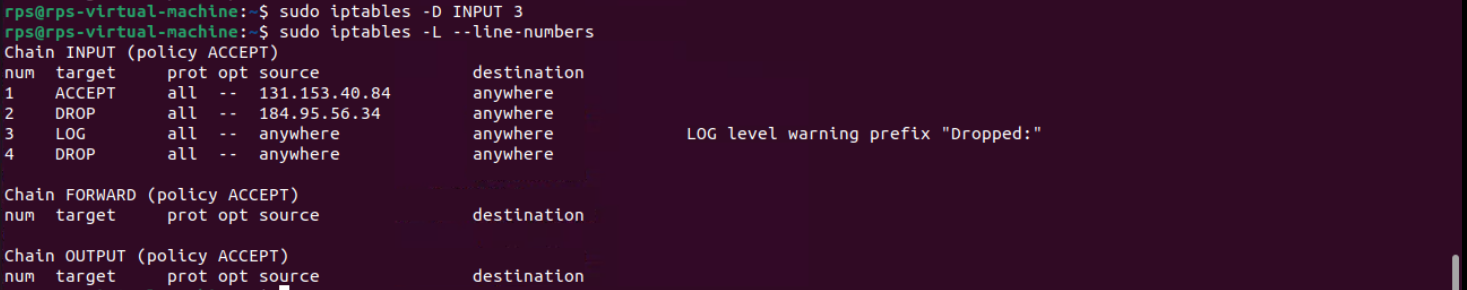
Use the **-F** option to clear all iptables firewall rules. To delete a specific rule, list all rules:

**sudo iptables -L --line-numbers**



Locate the line number of the firewall rule you want to delete and run:

**sudo iptables -D INPUT [number]**



Replace **[number]** with the rule line number you want to remove.

### **Block All Incoming Traffic Except SSH:**

To block all incoming traffic, except for SSH connections, do the following:

1. Set the default policy for the **INPUT** chain to **DROP**:

**sudo iptables -P INPUT DROP**

The **INPUT** table policy changes to **DROP**.

2. Allow SSH connections:

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

If the system uses a different [port for SSH](https://phoenixnap.com/kb/change-ssh-port) connections, change the port number in the command.

3. Allow related and established connections:

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

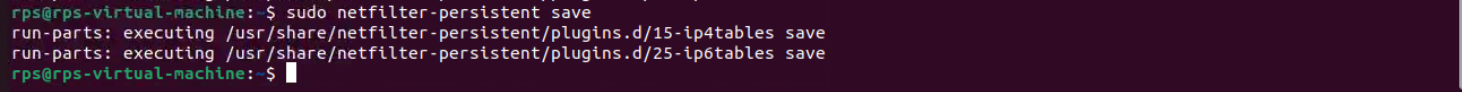
The command allows packets from established connections (such as responses to outgoing requests) and related connections, which is crucial for SSH session data.



**Save Your Changes:**

iptables does not persist rules when the system reboots. All the changes apply only until the first restart. To save the rules, see the commands below:

**sudo netfilter-persistent save**



On the next restart, iptables will automatically reload the firewall rules.

**Configure a Firewall with UFW:**

**What is UFW?**

UFW (Uncomplicated Firewall) is a simple tool for managing a firewall on Linux systems. It provides an easy-to-use interface to set up rules that allow or block network traffic. Designed to make firewall configuration straightforward, UFW is ideal for users who find **iptables** complex. Commonly used on Ubuntu, it helps improve system security with simple commands like enabling the firewall or allowing specific ports.

**Why UFW?**

ufw is preferred over iptables for its simplicity and user-friendly interface on Linux systems. It simplifies firewall configuration with commands like **ufw allow <port>** and **ufw deny <port>,** making it accessible even for beginners. By default, ufw denies all incoming connections and allows outgoing ones, which can be easily adjusted. It supports application profiles for common services and offers straightforward status checks (ufw status). While iptables provides more advanced control, ufw strikes a balance between ease of use and effectiveness, making it a practical choice for most users managing firewall settings.

## INSTALLING UFW ON LINUX

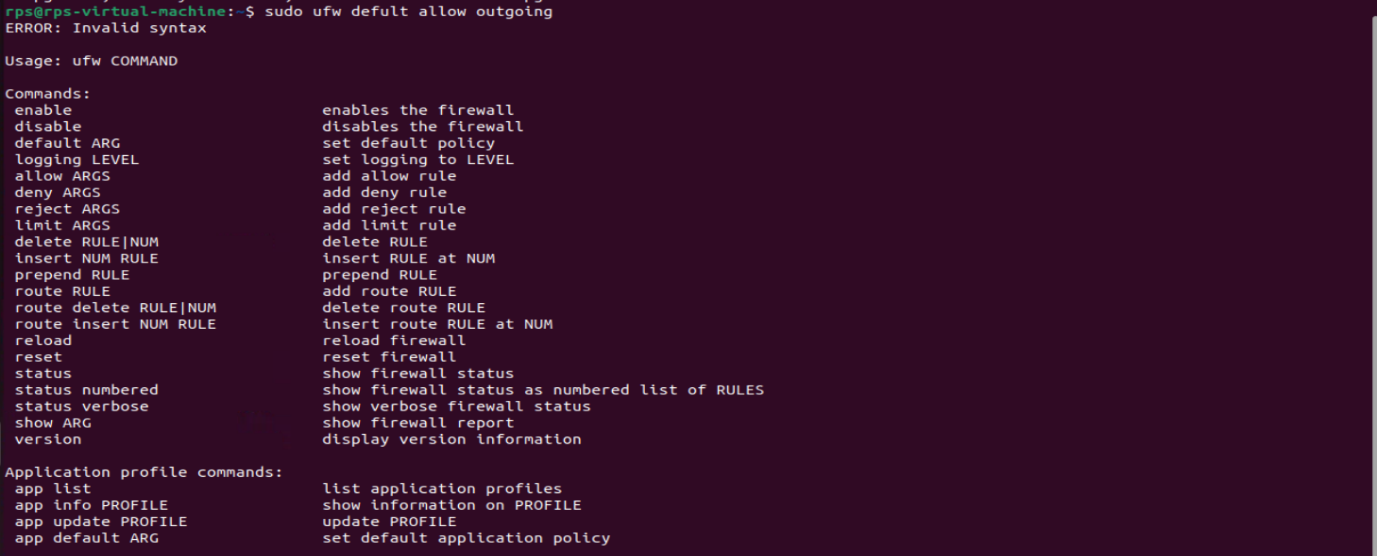
## sudo apt-get install ufw

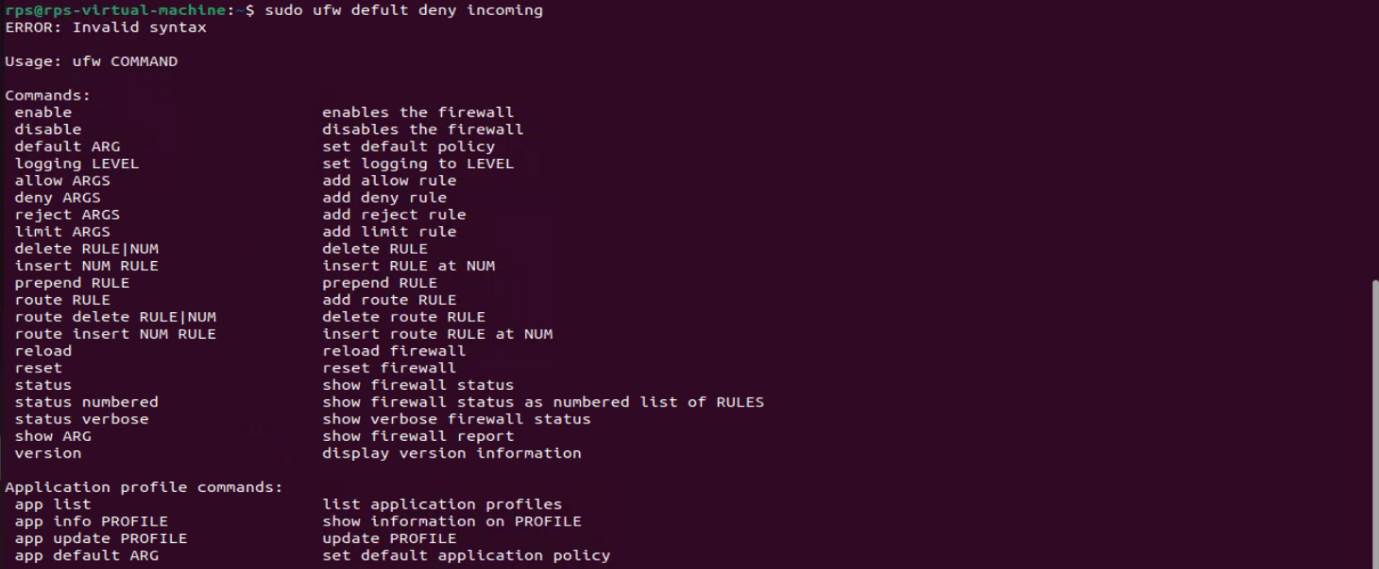
## Use UFW to Manage Firewall Rules

### **Set Default Rules:** To start with an easy basis of rules, the **ufw default**

### command can be used to set the default response to incoming and outgoing connections. To deny all incoming and allow all outgoing connections, run:

**sudo ufw default allow outgoing**

**sudo ufw default deny incoming**



The **ufw default** command also allows for the use of the **reject** parameter.

### **Add Rules**

Rules can be added in two ways: By denoting the **port number** or by using the **service name**.

For example, to allow both incoming and outgoing connections on port 22 for SSH, you can run:

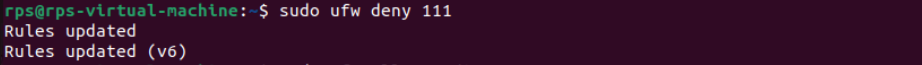
**sudo ufw allow ssh**

You can also run:

**sudo ufw allow 22**

Similarly, to **deny** traffic on a certain port (in this example, 111) you would only have to run:

**sudo ufw deny 111**



To further fine-tune your rules, you can also allow packets based on TCP or UDP. The following allows TCP packets on port 80:

**sudo ufw allow 80/tcp**

**sudo ufw allow http/tcp**



Whereas this will allow UDP packets on 1725:

**sudo ufw allow 1725/udp**



### **Advanced Rules**

Along with allowing or denying based solely on port, UFW also allows you to allow/block by IP addresses, subnets, and a IP address/subnet/port combinations.

To allow connections from an IP address:

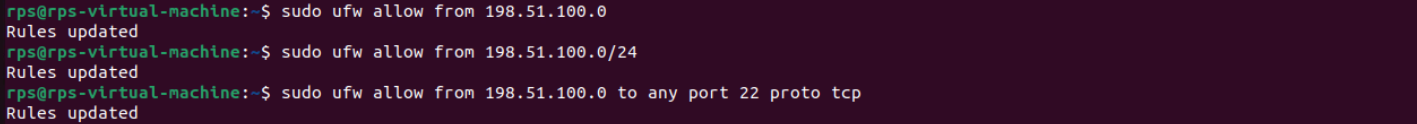
**sudo ufw allow from 198.51.100.0**

To allow connections from a specific subnet:

**sudo ufw allow from 198.51.100.0/24**

To allow a specific IP address/port combination:

**sudo ufw allow from 198.51.100.0 to any port 22 proto tcp**



**proto tcp** can be removed or switched to **proto udp** depending upon your needs, and all instances of **allow** can be changed to **deny** as needed.

**Remove Rules:** To remove a rule, add **delete** before the rule implementation. If you no longer wished to allow HTTP traffic, you could run:

**sudo ufw delete allow 80**



Deleting also allows the use of service names.

## Edit UFW’s Configuration Files

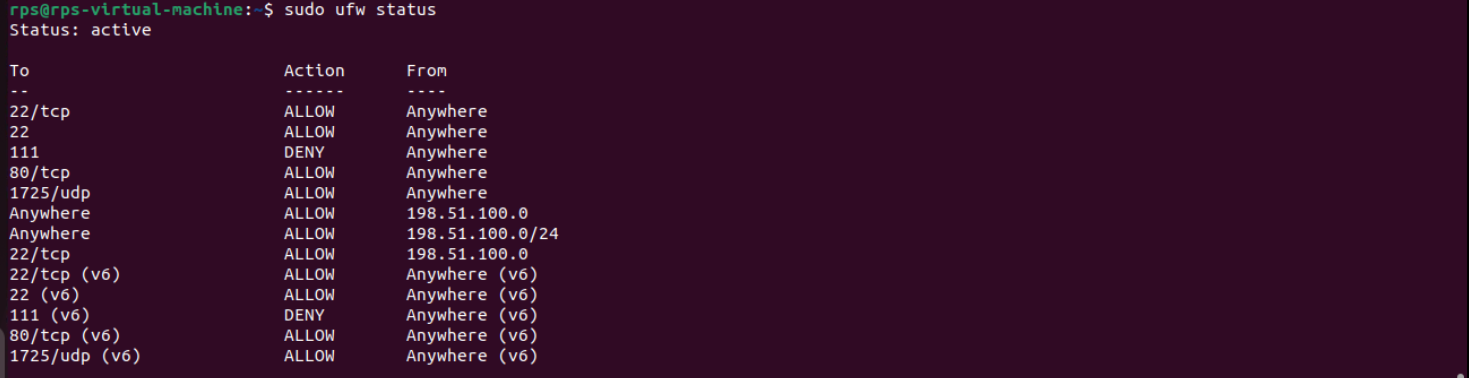
Although simple rules can be added through the command line, there may be a time when more advanced or specific rules need to be added or removed. Prior to running the rules input through the terminal, UFW will run a file, **before.rules**, that allows loopback, ping, and DHCP. To add to alter these rules edit the **/etc/ufw/before.rules** file. A **before6.rules** file is also located in the same directory for IPv6.

An **after.rule** and an **after6.rule** file also exists to add any rules that would need to be added after UFW runs your command-line-added rules.

An additional configuration file is located at **/etc/default/ufw**. From here IPv6 can be disabled or enabled, default rules can be set, and UFW can be set to manage built-in firewall chains.

## UFW Status

You can check the status of UFW at any time with the command: **sudo ufw** **status**. This will show a list of all rules, and whether or not UFW is active:



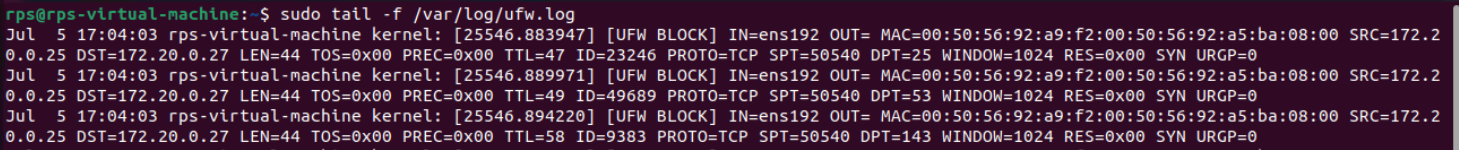
## Logging

You can enable logging with the command:

**sudo ufw logging on**

Log levels can be set by running **sudo ufw logging low|medium|high**, selecting either **low**, **medium**, or **high** from the list. The default setting is **low.**

A normal log entry will resemble the following, and will be located at **/var/log/ufw.log**:



The initial values list the date, time, and hostname of your Linode. Additional important values include:

* **[UFW BLOCK]:** This location is where the description of the logged event will be located. In this instance, it blocked a connection.
* **IN:** If this contains a value, then the event was incoming
* **OUT:** If this contain a value, then the event was outgoing
* **MAC:** A combination of the destination and source MAC addresses
* **SRC:** The IP of the packet source
* **DST:** The IP of the packet destination
* **LEN:** Packet length
* **TTL:** The packet TTL, or time to live. How long it will bounce between routers until it expires, if no destination is found.
* **PROTO:** The packet’s protocol
* **SPT:** The source port of the package
* **DPT:** The destination port of the package
* **WINDOW:** The size of the packet the sender can receive
* **SYN URGP:** Indicated if a three-way handshake is required. 0 means it is not.

### **Enable the Firewall**

With your chosen rules in place, your initial run of **ufw status** will probably output **Status: inactive**. To enable UFW and enforce your firewall rules:

**sudo ufw enable**

Similarly, to disable UFW’s rules:

**sudo ufw disable**

**CONCLUSION:**

Configuring a firewall in Linux using iptables and UFW is a crucial step in securing your system. This project not only enhances your system's security but also provides valuable hands-on experience with essential Linux tools and network security practices. By understanding and applying these configurations, you develop skills that are fundamental to any cybersecurity or system administration role. Additionally, it helps you gain confidence in managing and protecting network environments effectively.