|  |  |
| --- | --- |
| **Name** | **Akshat Biniwale** |
| **UID** | **2021300014** |
| **Branch** | **BE COMPS A** |
| **Batch** | **VI** |
| **Subject** | **Cryptography and System Security** |
| **Experiment** | 8 |

**Aim:**

To understand and implement IP tables on a Linux-based system to manage network traffic by configuring different rules for packet filtering and firewall setup.

**Objective:**

* To learn and implement basic to advanced IP table configurations.
* To explore how IP tables can be used to secure a Linux system by allowing or blocking specific network traffic.
* To observe the behaviour of packets when subjected to various rules and policies set in the IP tables.

**Theory:**

IP tables is a command-line firewall utility in Linux that uses policy chains to allow or block traffic. Each chain contains a set of rules that govern the behaviour of packets that pass through it. The main tables in IP tables are:

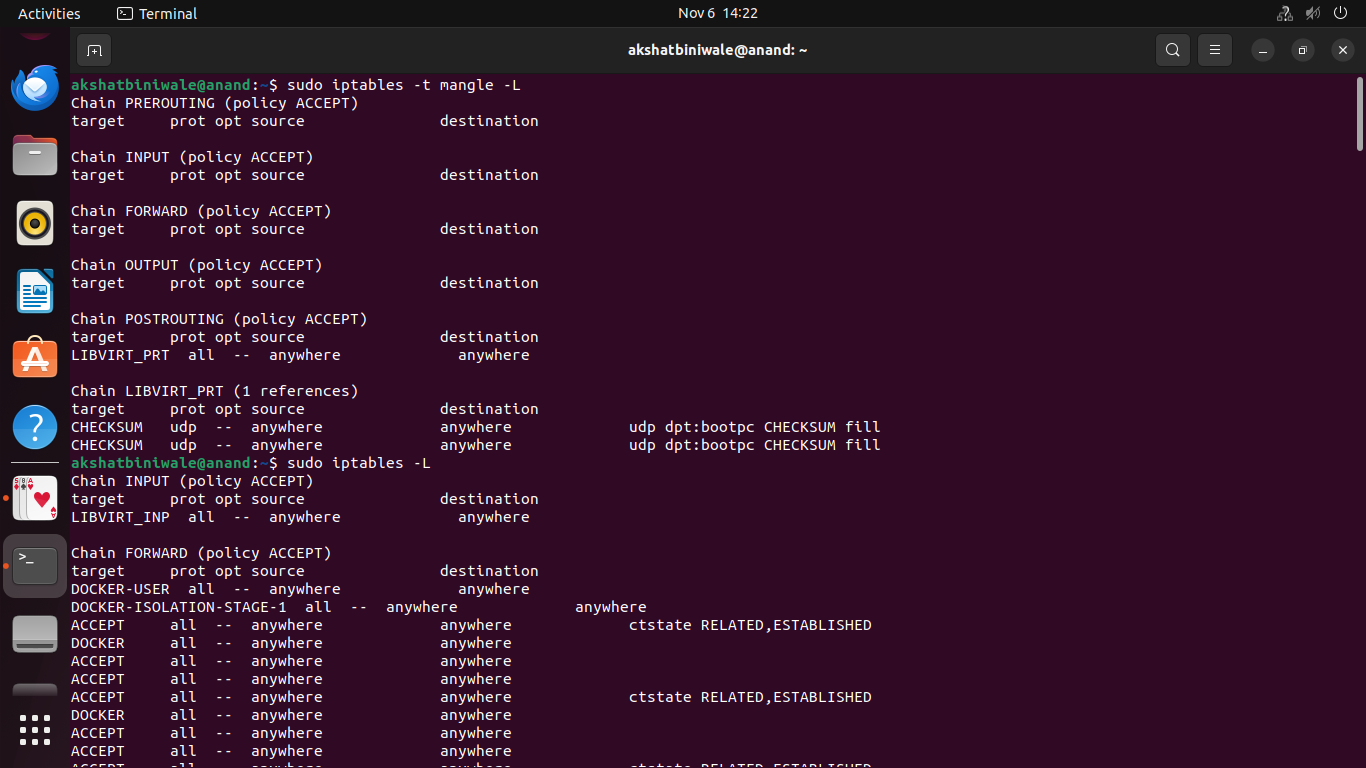
1. **Filter Table** - Default table for packet filtering, with built-in chains like INPUT, OUTPUT, and FORWARD.
2. **NAT Table** - Used for Network Address Translation, with chains like PREROUTING and POSTROUTING.
3. **Mangle Table** - Used for specialized packet alteration.

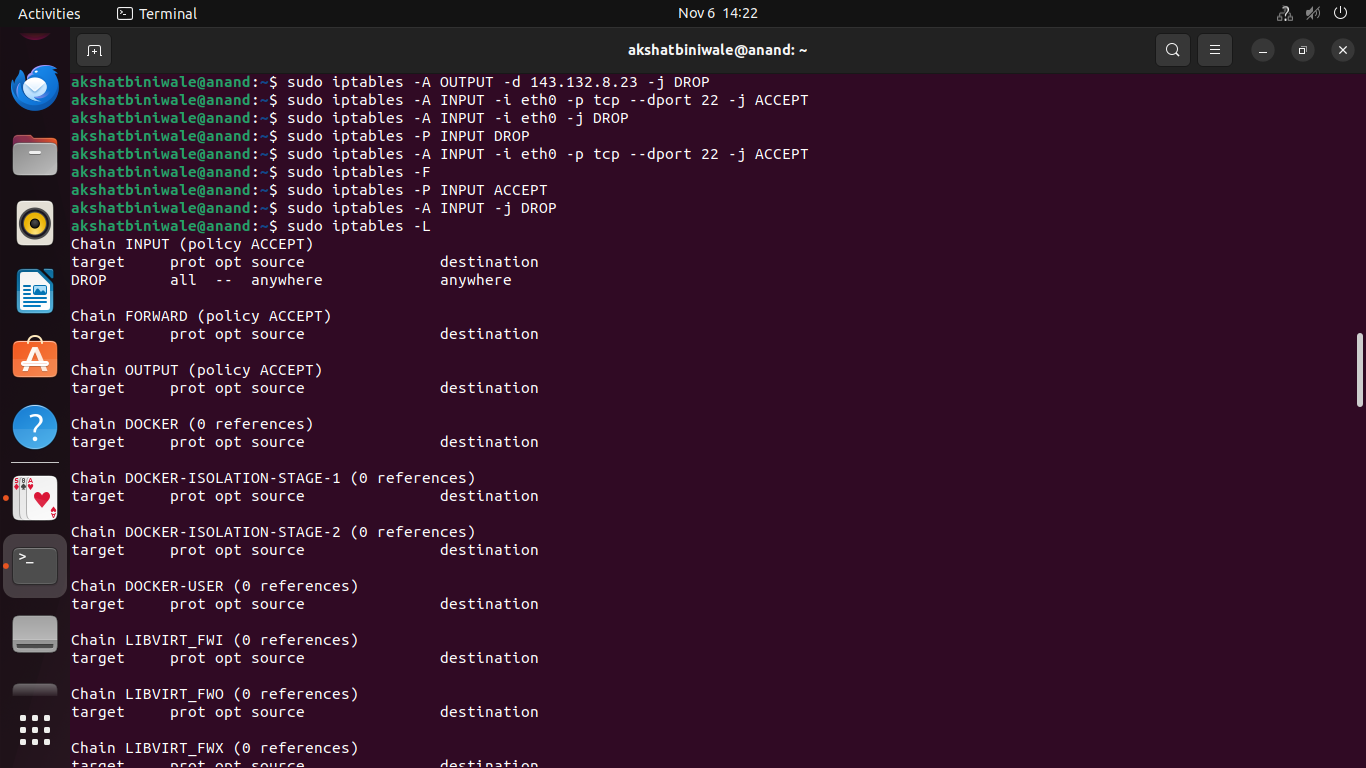
Each table has chains that consist of rules to specify the type of action taken on packets (e.g., ACCEPT, DROP). IP tables are commonly used for network security, preventing unauthorized access, and managing incoming and outgoing traffic.

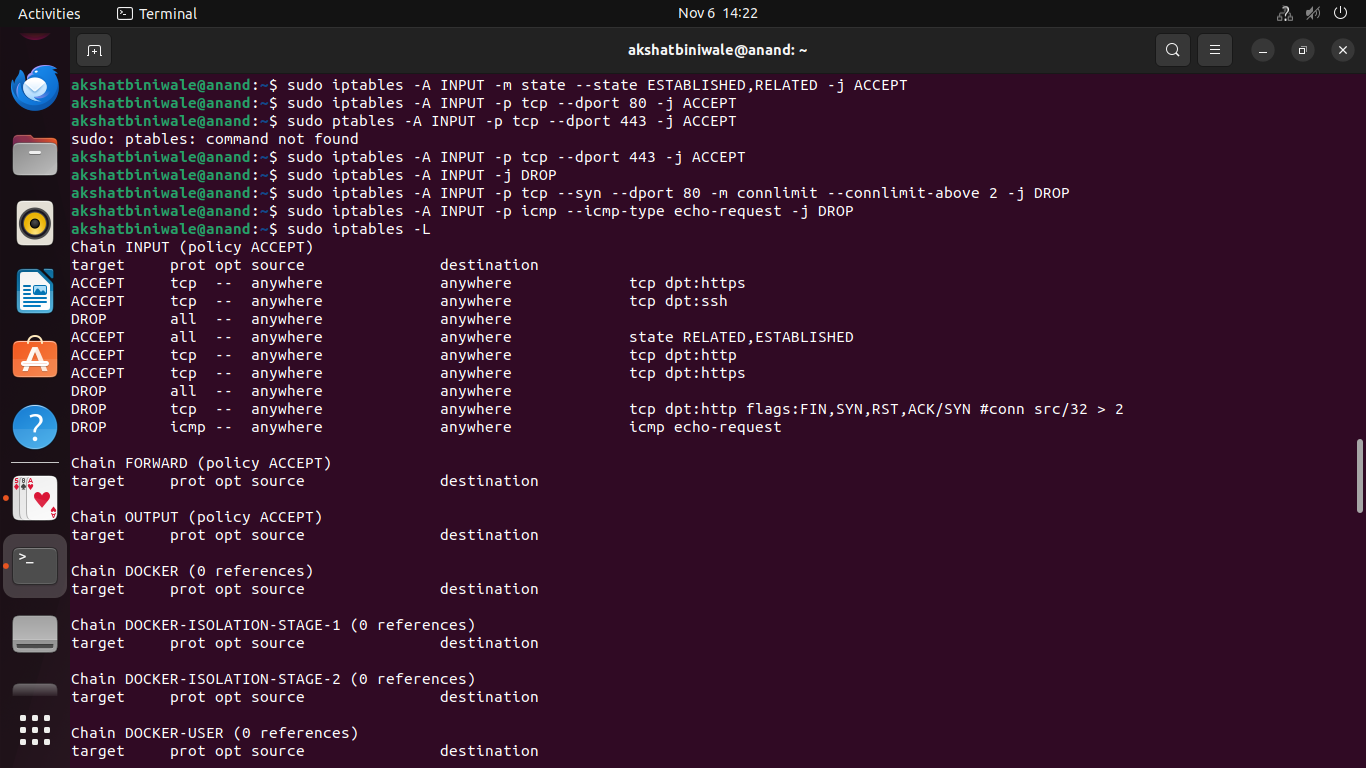
**Procedure and Observations:**

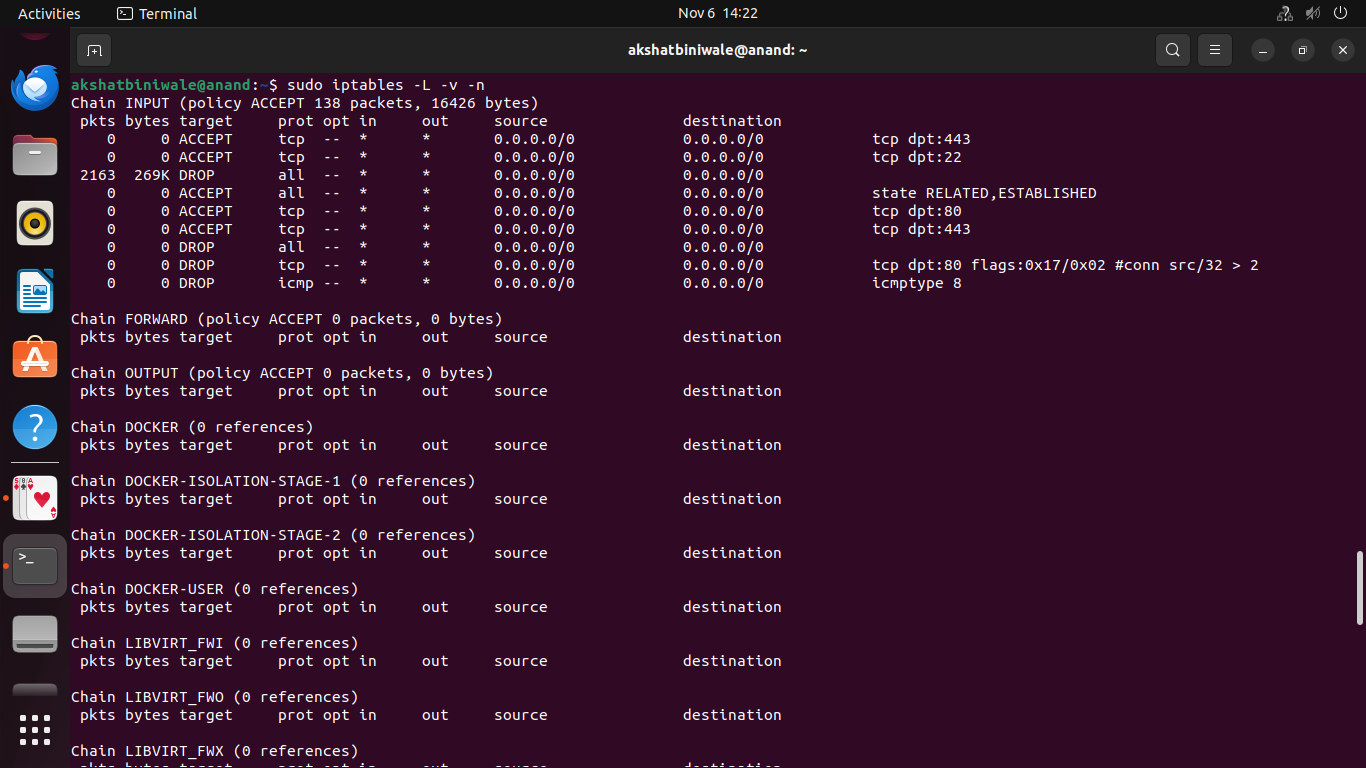
1. **Listing the mangle table:**
   * **Command:** iptables -t mangle -L
   * **Observation:** The mangle table is displayed, showing empty chains with the default policy set to ACCEPT.
2. **Listing the filter table:**
   * **Command:** iptables -L
   * **Observation:** The default filter table shows the chains and their default ACCEPT policies.
3. **Blocking access to a specific IP:**
   * **Command:** iptables -A OUTPUT -d <IP\_Address> -j DROP
   * **Observation:** Traffic to the specified IP was blocked. Testing in a browser confirmed that the IP could not be reached.
4. **Flushing the IP tables:**
   * **Command:** iptables -F
   * **Observation:** All rules in the current table were cleared, but the default policies remained intact.
5. **Allowing only SSH connections:**
   * **Command:**
     + iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
     + iptables -A INPUT -i eth0 -j DROP
   * **Observation:** Only SSH traffic was allowed, and other incoming traffic was blocked.
6. **Configuring access for a specific network range for web traffic:**
   * **Command:** iptables -A INPUT -i eth0 -p tcp --dport 80 -s <Network\_Range> -j ACCEPT
   * **Observation:** Web traffic from the specified network range was allowed, blocking other traffic.
7. **Blocking incoming pings while allowing outgoing pings:**
   * **Command:** iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
   * **Observation:** Remote machines could not ping the local machine, but the local machine could ping remote machines.

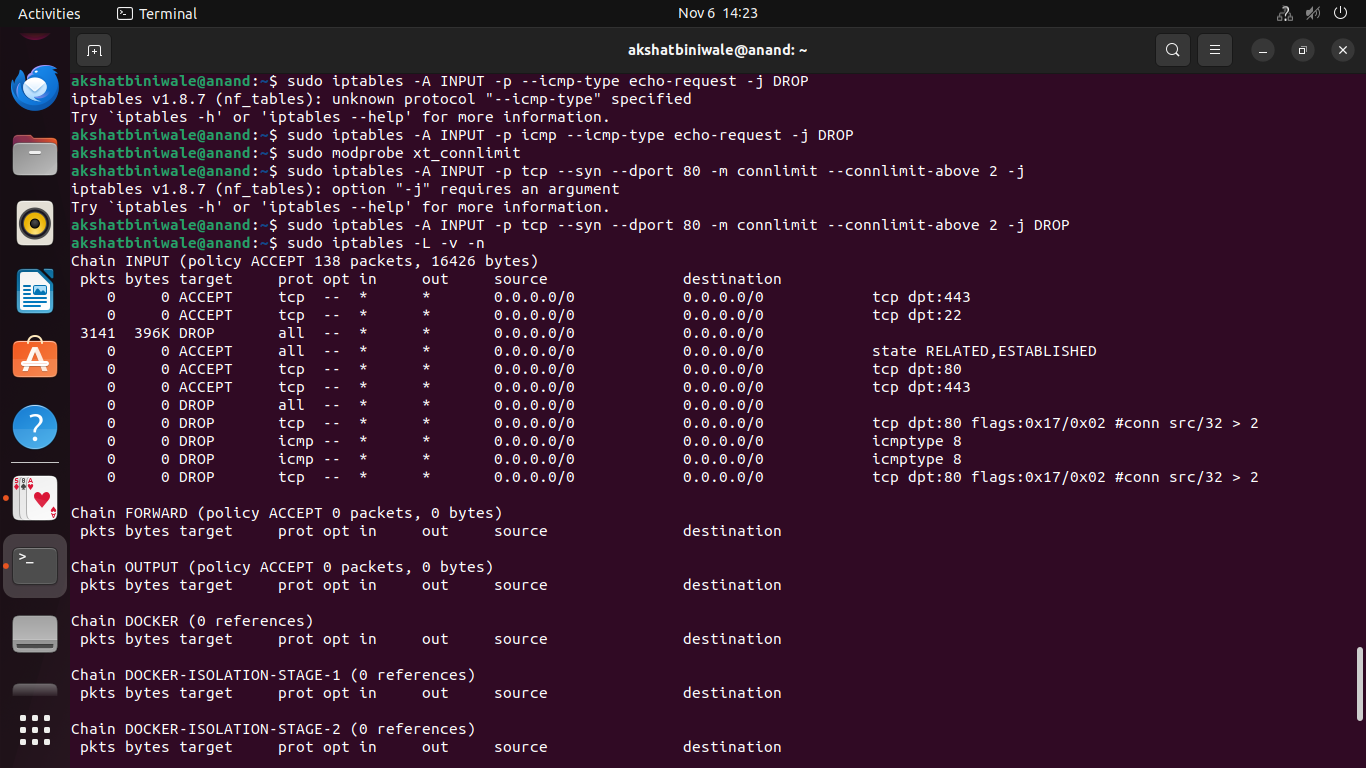
**Screenshots:**

  
Fig 1

  
Fig 2

  
Fig 3

  
Fig 4

  
Fig 5

**Conclusion:**

Through this experiment, we observed how to utilize IP tables to set up a firewall and control network traffic on a Linux machine. By applying various rules and policies, we could selectively permit or deny network access, enhancing system security. This hands-on experiment provided valuable insights into using IP tables for network security, making it a crucial tool for managing network traffic on Linux systems in real-world scenarios.