Session-3

VIM & Networking Basics

Vim (short for "Vi IMproved") is a highly configurable and powerful text editor that is designed for efficient text editing. It is an advanced version of the older Unix text editor called "vi." Vim is known for its modal editing system, which allows users to perform various tasks by switching between different modes.

Important commands for Vim Editor:

- gg: Go to the beginning of the file.
- G: Go to the end of the file.
- 0: Move to the beginning of the current line.
- $: Move to the end of the current line.
- w: Move to the beginning of the next word.
- b: Move back to the beginning of the current word.
- i: Enter Insert Mode before the cursor.
- a: Enter Insert Mode after the cursor.
- I: Enter Insert Mode at the beginning of the line.
- 3G: goto line 3
- /searchpattern: searches the pattern
- :%s/old/new/gc: Replace “old” word with “new” word with confirmation.
- A: Enter Insert Mode at the end of the line.
- o: Open a new line below the current line.
- O: Open a new line above the current line.
- r: Replace the character under the cursor.
- x: Delete the character under the cursor.
- d2: delete 2next lines+currentline
- dd: Delete the current line.
- u: Undo the last action.
- Ctrl + r: Redo the undone action.
- :w: Save the file.
- :q: Quit Vim.
- :wq: Save and quit.
- :q!: Quit without saving changes.

Networking Basics

Imagine your computer is like a house, and you want to communicate with other houses in your neighborhood. To do this, you need a network. In the Linux world, a network is like a bunch of houses (computers) that can talk to each other and share things, like information and files.

Here are a few key things to understand about networking in Linux:

- **IP Address:** Just like every house in your neighborhood has its own address, every computer on a network has a unique IP address. This address helps computers find each other. It's like the GPS coordinates of your computer on the network.

- **Router:** Think of the router as the neighborhood's central hub. It helps all the houses (computers) in the network communicate with each other and with the outside world, like the internet. It's like the mail center that forwards messages between houses.

- **Sending and Receiving Data:** When you want to send a message to another computer, your computer packages the message and puts an address on it (the IP address of the receiving computer). The router helps the message find its way to the right destination. When the message reaches the other computer, it opens the package and reads the message.

4. **Protocols:** Imagine that everyone in your neighborhood speaks a common language to understand each other. Similarly, computers use protocols (set of rules) to communicate. Some protocols help with sending emails, while others help with browsing websites.

5. **Firewall:** Just like you might have a fence around your house for security, Linux has a firewall to protect your computer from unwanted stuff coming in from the network. It decides which messages are allowed to enter your computer and which should be blocked.

6. **Wired and Wireless:** Just as you can connect to the internet using a cable or Wi-Fi at home, your Linux computer can connect using Ethernet cables (wired) or Wi-Fi signals (wireless).

7. **Network Services:** Think of network services as different types of shops in the neighborhood. Some computers might be running a service to share files, while others might offer web pages. These services make the network more useful.

8. **DNS:** When you type a website address in your browser, your computer needs to know the IP address of that website. DNS (Domain Name System) is like a phone book that translates the website address into an IP address so your computer can find it on the network.

So, in a nutshell, networking in Linux is all about computers talking to each other, sharing information, and working together like neighbors in a community. It's what lets your computer connect to the internet, access files on other computers, and do so much more!

LAN

A LAN is a network of interconnected devices within a limited geographic area, such as a home, office building, or campus. LANs are used to facilitate communication and data sharing between devices that are in close proximity to each other.

In a Local Area Network (LAN), a switch is a networking device that plays a crucial role in connecting and managing the communication between devices within the LAN. Switches operate at the data link layer

(Layer 2) of the OSI (Open Systems Interconnection) model and are essential for creating efficient and reliable LAN environments

An IP address (Internet Protocol address) is like a unique home address for devices on a network, allowing them to find and communicate with each other. It's made up of numbers separated by dots, such as 192.168.1.1. Here are examples to explain different aspects:

1. **IPv4 Addressing:**
   - Example: 192.168.0.10
   - Think of it like a street address for your computer on the internet or local network. Each number (0-255) identifies a part of the address hierarchy.
2. **Subnet Mask:**
   - Example: 255.255.255.0
   - It's like a fence that divides your big address space into smaller parts. It helps devices know which addresses are in the same "neighborhood."
3. **Default Gateway:**
   - Example: 192.168.0.1
   - This is like the main entrance to your neighborhood. When your device wants to talk to devices in other neighborhoods (outside your local network), it goes through this gateway.
4. **DNS Address:**
   - Example: 8.8.8.8
   - It's like the phonebook of the internet. Converts domain names (like www.example.com) to IP addresses. DNS helps your device find websites.

Remember, IP addresses help devices communicate by uniquely identifying them, just like addresses do for houses.

An IP address consists of four octets (8-bit groups) represented in binary form. Each octet represents a portion of the IP address, ranging from 0 to 255 in decimal. Understanding IP addresses in binary octets can help illustrate how computers identify and communicate with each other on networks. Let's break down an example IP address into its binary octets:

Example IP Address: 192.168.1.10

1. **Convert to Binary:**
   - 192 in binary: 11000000
   - 168 in binary: 10101000
   - 1 in binary: 00000001
   - 10 in binary: 00001010
2. **Representing in Octets:**

- First Octet (192): 11000000
- Second Octet (168): 10101000
- Third Octet (1): 00000001
- Fourth Octet (10): 00001010

This IP address, 192.168.1.10, can be broken down into its binary octets like this:

```
11000000 . 10101000 . 00000001 . 00001010
```

Each octet represents a portion of the address, and the dots separate the octets. Computers use these binary patterns to route data within networks.

In this example, the binary pattern helps routers and devices know how to direct data packets to the correct destination. It's like the postal code of a house, where each digit specifies a part of the route for the package.

Concepts of a router and a gateway IP address with examples:

Router: A router is a networking device that connects different networks together and directs traffic between them. It's like a traffic cop that decides where data should go within a network and beyond. In a home or office network, the router acts as a central hub that allows devices to communicate with each other and with the internet.

Example: Imagine your home network. Your router has an IP address that serves as its identity within the network. Let's say your router's IP address is 192.168.1.1. It's like the main entrance to your home's network, handling data traffic between devices in your home and the larger internet.

Gateway IP Address: A gateway IP address is the address of a device on a network that serves as an entry and exit point to another network. It's usually associated with the router that connects your local network to the internet or another external network.

Example: Continuing from the previous example, your router's IP address (192.168.1.1) is also the gateway IP address for devices within your home network. When a device wants to access resources outside your home network, like a website on the internet, it sends data through the gateway (your router), which then forwards the data to its intended destination and brings back the response.

In summary, the router is the device that manages traffic within your local network and also serves as the gateway to connect your local network to the larger internet or another external network. The gateway IP address is the IP address of this router, which serves as the entry and exit point for data traveling between your local network and external networks.

SUBNETS:

The subnet concept is a fundamental aspect of networking that involves dividing a larger IP network into smaller, manageable segments. Subnetting is used to efficiently allocate IP addresses, manage network traffic, and improve security. Let's explore this concept in more detail:

1. IP Address Structure: An IP address consists of two parts: the network portion and the host portion. Subnetting focuses on dividing the IP address space into these two segments.

2. Subnet Mask: A subnet mask is a 32-bit value that separates the network portion and the host portion of an IP address. It uses a combination of 1s and 0s to indicate which bits represent the network and host portions. The subnet mask is typically written in decimal-dotted format, like 255.255.255.0.

3. Why Subnet? Subnetting allows you to create smaller networks within a larger network. This has several benefits:

- Efficient IP address allocation: You can assign IP addresses based on specific needs within each subnet.
- Improved network performance: Smaller subnets mean less broadcast traffic and more efficient routing.
- Enhanced security: Subnets can be isolated from each other, providing better control over network access.

4. Subnet Calculation: To create subnets, you borrow bits from the host portion of an IP address to form the subnet portion. The number of borrowed bits determines the number of subnets and hosts per subnet you can have.

5. Subnetting Example: Let's consider the IP address 192.168.1.0 with a subnet mask of 255.255.255.0 (which represents a /24 subnet). By borrowing a few host bits, you can create subnets like this:

- Subnet 1: 192.168.1.0/25 (255.255.255.128)
- Subnet 2: 192.168.1.128/25 (255.255.255.128)

6. CIDR Notation: CIDR (Classless Inter-Domain Routing) notation is used to represent IP addresses and subnet masks more efficiently. It uses a prefix length to indicate the number of bits in the subnet mask. For example, the /24 subnet mask can be represented as 192.168.1.0/24.

7. VLSM: Variable Length Subnet Masking (VLSM) is a technique where different subnets can have different subnet mask lengths. This allows for more efficient utilization of IP address space.

In summary, subnetting is the process of dividing a larger IP network into smaller subnetworks, enabling efficient IP address management, improved network performance, and better security. It involves using subnet masks and CIDR notation to define the boundaries of these subnetworks. Subnetting is a crucial skill for network administrators and engineers to design and maintain efficient and scalable networks.

Let's dive deeper into subnetting, subnet masks, CIDR, and subnetting with detailed examples:

1. Subnetting: Subnetting is the process of breaking a larger IP network into smaller subnetworks or subnets. This helps in efficient address allocation, improved network management, and better control over network traffic.

2. Subnet Mask: A subnet mask is a 32-bit value that separates the network portion and the host portion of an IP address. It uses binary bits to indicate which parts of the IP address belong to the network and which parts belong to individual hosts within that network.

3. CIDR (Classless Inter-Domain Routing): CIDR notation is a compact way to represent IP addresses and subnet masks. It includes the IP address followed by a slash and a number indicating the number of bits in the subnet mask. For example, "192.168.1.0/24" means a network with an IP address of 192.168.1.0 and a subnet mask of 24 bits.

4. Subnetting Example: Let's consider the IP address range "192.168.0.0" with a default subnet mask of "255.255.255.0" (or /24 in CIDR notation). We'll divide this network into smaller subnets.

Step 1: Start with the default subnet mask (24 bits) and identify how many subnets and hosts per subnet are needed. Let's say we need 4 subnets and each subnet should accommodate 30 hosts.

Step 2: Determine the number of bits needed for the subnet portion of the IP address. Calculate this using the formula: 2^n >= Number of Subnets, where n is the number of subnet bits.

In this case, 2^n >= 4. So, n = 2 bits for subnetting.

Step 3: Now, subtract the number of subnet bits from the default 24 bits to find the remaining bits for host addresses: 24 - 2 = 22 bits for host addresses.

Step 4: Calculate the subnet mask for the subnets: It will be the default subnet mask (24 bits) with the additional subnet bits (2 bits) set to 1. This gives us a subnet mask of "255.255.255.192."

Step 5: Divide the original IP address range into 4 subnets:

- Subnet 1: 192.168.0.0/26 (IP Range: 192.168.0.0 - 192.168.0.63)
- Subnet 2: 192.168.0.64/26 (IP Range: 192.168.0.64 - 192.168.0.127)
- Subnet 3: 192.168.0.128/26 (IP Range: 192.168.0.128 - 192.168.0.191)
- Subnet 4: 192.168.0.192/26 (IP Range: 192.168.0.192 - 192.168.0.255)

In each subnet, the first and last addresses are reserved (network address and broadcast address, respectively), leaving 62 usable addresses for hosts in each subnet.

By subnetting, we've efficiently divided the original network into smaller segments, meeting the requirement for 4 subnets, each with 30 hosts. This allows for better address management and traffic control within the network.

EXAMPLE OF CONVERTNG AN IP IN TO 4&7 SUBNETS

To subnet the given IP address "192.168.32.8" with a subnet mask of "255.255.255.0" into 4 and 7 subnets, we need to determine the subnet mask for each case and calculate the subnet ranges.

1. For 4 Subnets:

Step 1: Calculate the number of bits needed to represent 4 subnets: 2^2 = 4, so we need 2 bits for subnetting.

Step 2: Subtract the subnet bits from the total bits in the default subnet mask (24 bits) to get the host bits: 24 - 2 = 22 bits for hosts.

Step 3: Calculate the new subnet mask:

- Network Bits: 24 (unchanged)
- Subnet Bits: 2
- Host Bits: 22
- New Subnet Mask: 255.255.255.192

Step 4: Divide the IP address range into 4 subnets:

- Subnet 1: 192.168.32.0/26 (IP Range: 192.168.32.0 - 192.168.32.63)
- Subnet 2: 192.168.32.64/26 (IP Range: 192.168.32.64 - 192.168.32.127)
- Subnet 3: 192.168.32.128/26 (IP Range: 192.168.32.128 - 192.168.32.191)
- Subnet 4: 192.168.32.192/26 (IP Range: 192.168.32.192 - 192.168.32.255)

2. For 7 Subnets:

Step 1: Calculate the number of bits needed to represent 7 subnets: The nearest power of 2 that is equal to or greater than 7 is 8, which requires 3 bits for subnetting.

Step 2: Subtract the subnet bits from the total bits in the default subnet mask (24 bits) to get the host bits: 24 - 3 = 21 bits for hosts.

Step 3: Calculate the new subnet mask:

- Network Bits: 24 (unchanged)
- Subnet Bits: 3
- Host Bits: 21
- New Subnet Mask: 255.255.255.224

Step 4: Divide the IP address range into 7 subnets:

- Subnet 1: 192.168.32.0/27 (IP Range: 192.168.32.0 - 192.168.32.31)
- Subnet 2: 192.168.32.32/27 (IP Range: 192.168.32.32 - 192.168.32.63)
- Subnet 3: 192.168.32.64/27 (IP Range: 192.168.32.64 - 192.168.32.95)
- Subnet 4: 192.168.32.96/27 (IP Range: 192.168.32.96 - 192.168.32.127)
- Subnet 5: 192.168.32.128/27 (IP Range: 192.168.32.128 - 192.168.32.159)
- Subnet 6: 192.168.32.160/27 (IP Range: 192.168.32.160 - 192.168.32.191)
- Subnet 7: 192.168.32.192/27 (IP Range: 192.168.32.192 - 192.168.32.223)

These calculations result in the subnet ranges for both 4 and 7 subnets based on the given IP address and subnet mask.

Network Address Translation (NAT)

Network Address Translation (NAT) is a technique used in networking to allow multiple devices within a local network to share a single public IP address. NAT helps manage the limited availability of IPv4 addresses and provides security by hiding internal network details from the external world. Let's understand NAT with an example:

Scenario: Home Network with NAT

Imagine you have a home network with multiple devices (computers, smartphones, smart devices) all connected to a router. Your ISP provides you with a single public IP address, but you have several devices that need to access the internet simultaneously.

Without NAT:

- All devices in your home network would need unique public IP addresses.
- This would quickly exhaust the limited pool of available IPv4 addresses.

With NAT:

- Your router acts as a NAT device. It assigns private IP addresses to the devices within your home network.

- Private IP addresses are used within your local network and are not routable on the internet.

Example:

- Your router's public IP address: 203.0.113.5
- Your home network's private IP range: 192.168.1.0/24

1. Your laptop (192.168.1.10) wants to access a website (e.g., www.example.com).
2. Your laptop sends a request to your router, saying, "I want to visit www.example.com."
3. Your router changes the source IP address in the request to its own public IP (203.0.113.5).
4. The request reaches the website, which sends back the response to your router's public IP.
5. Your router receives the response and checks its NAT table to determine which device requested the website.
6. Your router changes the destination IP address in the response to your laptop's private IP (192.168.1.10).
7. The response reaches your laptop, and you see the website.

Benefits of NAT:

- **IP Conservation:** NAT helps reduce the need for multiple public IP addresses, as several devices share a single public IP.
- **Security:** NAT acts as a barrier between the public internet and private devices. External entities don't directly interact with your devices, enhancing security.
- **Simplified Configuration:** Devices in the private network don't require unique public IP addresses, simplifying network management.

In summary, Network Address Translation (NAT) is a technique that enables multiple devices with private IP addresses to share a single public IP address for accessing the internet. NAT enhances security and conserves public IP addresses while allowing internal devices to communicate with the external network.

FIREWALL

A firewall in Linux networking is a software or hardware-based security system that controls incoming and outgoing network traffic based on a set of rules. It acts as a barrier between your computer and the outside world, allowing you to define what traffic is allowed or blocked. Let's delve into this with real examples:

Example 1: Basic Firewall Rule Let's say you want to allow only incoming SSH (Secure Shell) connections to your Linux server:

1. You set up a firewall rule that allows incoming traffic on port 22 (SSH port).
2. Any external device trying to establish an SSH connection to your server will be allowed, while other incoming traffic is blocked.

Example 2: Web Server and HTTP Traffic Imagine you're running a web server and want to permit only HTTP (port 80) and HTTPS (port 443) traffic while blocking other types:

1. You configure your firewall to allow incoming traffic on ports 80 and 443.
2. This allows users to access your website over HTTP and HTTPS, while other ports remain closed for security.

Example 3: Outgoing Traffic Control You might want to restrict outgoing traffic from your network:

1. You set up a firewall rule to block all outgoing traffic on port 25 (SMTP) except from your mail server.
2. This prevents unauthorized devices on your network from sending emails directly, reducing the risk of spam.

Example 4: Stateful Inspection Firewalls can also perform stateful inspection, which tracks the state of active connections:

1. When you initiate an outbound request (e.g., visiting a website), the firewall dynamically opens a port for the response traffic.
2. Once the communication is complete, the firewall automatically closes that port.

Example 5: Application Filtering Modern firewalls offer application filtering, allowing or blocking specific applications:

1. You can configure the firewall to allow certain applications (e.g., web browsers) to access the internet while blocking others (e.g., peer-to-peer file sharing).

Example 6: Logging and Reporting Firewalls can log traffic and events for analysis and troubleshooting:

1. If a certain rule is triggered (e.g., a blocked connection attempt), the firewall logs the event.
2. You can review these logs to understand network activity and identify potential threats.

In Linux, the built-in firewall management tool is often "iptables" or its successor "nftables," while distributions like Ubuntu might use "ufw" (Uncomplicated Firewall) for simplified management. More advanced firewall solutions like "firewalld" provide dynamic configuration and user-friendly interfaces.

Firewalls are crucial for network security, helping to prevent unauthorized access, data breaches, and cyberattacks. By defining rules and policies, you control the flow of traffic in and out of your system, safeguarding your network and resources.

PORTS

In Linux, "ports" refer to communication endpoints that allow different applications and services to exchange data over a network. Ports are identified by numbers and are categorized as either TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) ports. Each protocol has its own set of port numbers, and they help direct data to the appropriate application or service running on a system. Here are some examples to help you understand ports in Linux:

1. Common Port Numbers:

- Port 80: HTTP - Web servers use this port to serve websites.
- Port 443: HTTPS - Secure version of HTTP used for encrypted web communication.

- Port 22: SSH - Secure Shell for remote access and secure file transfers.
- Port 25: SMTP - Simple Mail Transfer Protocol for sending emails.
- Port 53: DNS - Domain Name System for translating domain names to IP addresses.
- Port 3306: MySQL - Database system port for MySQL database connections.
- Port 8080: Alternative HTTP port often used for web proxies or non-standard setups.

2. Port Number Range:

- Well-known Ports (0-1023): Reserved for standard services like HTTP, FTP, SMTP.
- Registered Ports (1024-49151): Used for various applications and services.
- Dynamic/Private Ports (49152-65535): Typically used for temporary connections.

3. Checking Open Ports: In Linux, you can use tools like "netstat" or "ss" to check open ports and established connections:

```
# Using netstat
netstat -tuln

# Using ss (Socket Statistics)

ss -tuln
```

4. Firewall Rules and Ports: Linux firewalls (like iptables, nftables, or firewalld) allow you to define rules to control which ports are open or closed:

```
# Allow incoming traffic on port 80 (HTTP) using iptables
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# Allow incoming traffic on port 443 (HTTPS) using firewalld
sudo firewall-cmd --add-port=443/tcp --permanent

sudo firewall-cmd –reload
```

5. Network Services and Ports: Different services use specific ports to function. For example, a web server uses port 80 for unencrypted HTTP traffic and port 443 for encrypted HTTPS traffic.

6. Binding to Ports: When you run a server application, it binds to a specific port number. For instance, a web server binds to port 80 or 443 to listen for incoming web requests.

Ports are essential for networking, allowing various applications to communicate with each other using standardized communication endpoints. Understanding ports is important for configuring firewalls, troubleshooting network issues, and securing your Linux system.

DNS

DNS (Domain Name System) in Linux, as well as in general networking, is a crucial service that translates human-readable domain names (like www.example.com) into IP addresses (like 192.168.1.1). It acts like a phonebook for the internet, making it easier for us to access websites and services. Here's an explanation with examples:

1. How DNS Works: When you type a website URL into your browser, your computer needs to find the IP address associated with that domain name to connect to the web server. The DNS system handles this translation process.

2. DNS Components:

- **DNS Client:** Your computer acts as a DNS client. When you enter a domain name in your browser, the DNS client queries DNS servers to resolve the IP address.
- **DNS Server:** These servers store information about domain names and their corresponding IP addresses. They provide answers to DNS queries.

3. DNS Resolution Process: Let's say you want to visit "www.example.com."

1. Your computer sends a DNS query to a DNS resolver. This could be a local DNS server provided by your ISP or a public DNS server like Google's (8.8.8.8).
2. The resolver checks its cache. If it already knows the IP address for "www.example.com," it returns the result.
3. If not cached, the resolver contacts a root DNS server to find out which DNS server is authoritative for the ".com" top-level domain.
4. The root DNS server directs the resolver to the ".com" DNS server.
5. The ".com" DNS server guides the resolver to the DNS server responsible for "example.com."

6. The "example.com" DNS server finally provides the IP address for "www.example.com" to the resolver.
7. The resolver sends the IP address to your computer, and your browser can now connect to the web server.

4. DNS Records: DNS servers store various types of records, including:

- **A Record:** Maps a domain name to an IPv4 address.
- **AAAA Record:** Maps a domain name to an IPv6 address.
- **CNAME Record:** Creates an alias for another domain name (e.g., subdomain.example.com could be an alias for www.example.com).

5. DNS Configuration in Linux: In Linux, DNS configuration involves editing the **/etc/resolv.conf** file. This file specifies the DNS servers your system should use for name resolution.

```
nameserver 8.8.8.8

nameserver 8.8.4.4
```

6. Command-Line DNS Tools:

- **nslookup**: Used to query DNS servers for information about domain names and IP addresses.
- **dig** (Domain Information Groper): A powerful tool for querying DNS servers.

```
nslookup example.com

dig www.example.com
```

- DNS is an essential part of the internet, enabling seamless browsing by converting human-friendly domain names into computer-readable IP addresses. It plays a vital role in connecting users to online resources and services.

Here's a list of 50 essential Linux networking commands along with examples:

1. ifconfig

Display or configure network interfaces.

Example: ifconfig eth0

2. ip

Display or configure network interfaces and routing.

Example: ip addr show

3. ping

Send ICMP echo requests to a host for testing reachability.

Example: ping google.com

4. traceroute

Display the route taken by packets to reach a host.

Example: traceroute google.com

5. netstat

Display network statistics and active connections.

Example: netstat -tuln

6. ss

Similar to netstat, displays socket statistics.

Example: ss -tuln

7. route

View or manipulate the kernel routing table.

Example: route -n

8. dig

Query DNS servers for DNS-related information.

Example: dig google.com

9. nslookup

Query DNS servers for DNS-related information (alternative to dig).

Example: nslookup google.com

10. hostname

Display or set the system's hostname.

Example: hostname

11. iwconfig

Configure wireless network interfaces.

Example: iwconfig wlan0

12. iw

A newer tool for wireless configuration.

Example: iw dev wlan0 scan

13. ifup / ifdown

Activate or deactivate network interfaces.

Example: ifup eth0

14. iftop

Monitor network bandwidth usage in real-time.

Example: iftop -n 10

15. tcpdump

Capture and analyze network traffic.

Example: tcpdump -i eth0

16. nmap

Scan for open ports and services on a target host.

Example: nmap -p 80 google.com

17. ssh

Securely access remote systems over SSH.

Example: ssh username@hostname

18. scp

Securely copy files between hosts over SSH.

Example: scp file.txt remoteuser@remotehost:/path

19. wget

Download files from the internet.

Example: wget https://example.com/file.txt

20. curl

A versatile tool to transfer data with URLs.

Example: curl https://example.com

21. nc (netcat)

Networking utility for reading/writing to network connections.

Example: echo "Hello" | nc remotehost 1234

22. telnet

Interact with remote servers using the Telnet protocol.

Example: telnet example.com 80

23. arp

View and manipulate ARP cache entries.

Example: arp -a

24. ethtool

Display or modify Ethernet device settings.

Example: ethtool eth0

25. mtr

Combines ping and traceroute functionality.

Example: mtr google.com

26. route

View or manipulate routing tables.

Example: route -n

27. iptables

Configure and manage firewall rules.

Example: iptables -L

28. iperf

Measure network performance by sending/receiving data.

Example: Server: iperf -s, Client: iperf -c server_ip

29. dig

Query DNS servers for DNS information.

Example: dig google.com

30. ntpdate

Synchronize system time with NTP servers.

Example: ntpdate pool.ntp.org

31. netcat

Networking utility for sending/receiving data over TCP/UDP.

Example: nc -l 1234 (Listen), nc remotehost 1234 (Connect)

32. ifstat

Display bandwidth usage for network interfaces.

Example: ifstat -i eth0

33. ss

Display socket statistics.

Example: ss -tuln

34. host

Perform DNS lookups.

Example: host google.com

35. dhclient

Request IP address from a DHCP server.

Example: dhclient eth0

36. hostnamectl

View or change hostname-related settings.

Example: hostnamectl status

37. nmcli

Control NetworkManager from the command line.

Example: nmcli connection show

38. smbclient

Interact with Windows or Samba shares.

Example: smbclient //server/share -U user

39. ipcalc

Calculate IP subnet information.

Example: ipcalc 192.168.1.0/24

40. route

Manage IP routing tables.

Example: route add default gw gateway_ip

41. ssh-keygen

Generate SSH key pairs.

Example: ssh-keygen -t rsa

42. iptables-save / iptables-restore

Save and restore iptables rules.

Example: iptables-save > rules.txt, iptables-restore < rules.txt

43. iptraf

Interactive console-based network monitoring tool.

Example: iptraf

44. tcpdump

Capture and analyze network traffic.

Example: tcpdump -i eth0

45. arp-scan

Scan for active hosts on a network.

Example: arp-scan 192.168.1.0/24

46. hostname

Display or set the system's hostname.

Example: hostname

47. ss

Display socket statistics.

Example: ss -tuln

48. dig

Query DNS servers for DNS information.

Example: dig google.com

49. iperf3

Measure network performance using iperf tool.

Example: Server: iperf3 -s, Client: iperf3 -c server_ip

50. nc (netcat)

Networking utility for reading/writing to network connections.

Example: echo "Hello" | nc remotehost 1234