HO#1.4: Inter-Networking Concepts with Linux

TCP/IP Stack

The Internet is a worldwide network that links millions of private, public, academic, business, and government networks using a common set of protocols. It enables devices to connect, share information, and communicate across different regions and countries. The Transmission Control Protocol/Internet Protocol (TCP/IP): is a set of communication protocols essential for interconnecting network devices on the internet. Besides its pivotal role on the internet, TCP/IP is also utilized as a communications protocol within private computer networks, such as intranets and extranets. Its significance can be summarized as follows:

- **Standardization and Interoperability:** TCP/IP provides a standardized set of rules and protocols that ensure devices from different manufacturers and operating systems can communicate with each other effectively. This standardization has enabled the global interconnectivity of networks and devices.

- **Reliable Data Transmission:** TCP (Transmission Control Protocol): TCP is responsible for ensuring reliable, ordered, and error-checked delivery of data between applications over a network. It manages data packet sequencing, retransmission of lost packets, and error correction, making it essential for applications where data integrity is crucial, such as web browsing and email.

- **Addressing and Routing:** IP (Internet Protocol) provides a unique addressing scheme that identifies devices on a network, allowing data to be routed to its correct destination. This addressing system (both IPv4 and IPv6) is fundamental for the Internet's scalability and functionality, facilitating global communication between devices.

- **Scalability:** TCP/IP supports the growth of the Internet by allowing the connection of a vast number of devices and networks. Its hierarchical addressing and routing capabilities enable efficient handling of large-scale networks and the continuous expansion of the Internet.

- **Flexibility and Adaptability:** TCP/IP is not a single protocol but a suite of protocols, including TCP, IP, UDP (User Datagram Protocol), and others. This modular approach allows for flexibility and adaptation, enabling various types of communication, such as reliable data transfer (TCP) and real-time applications (UDP).

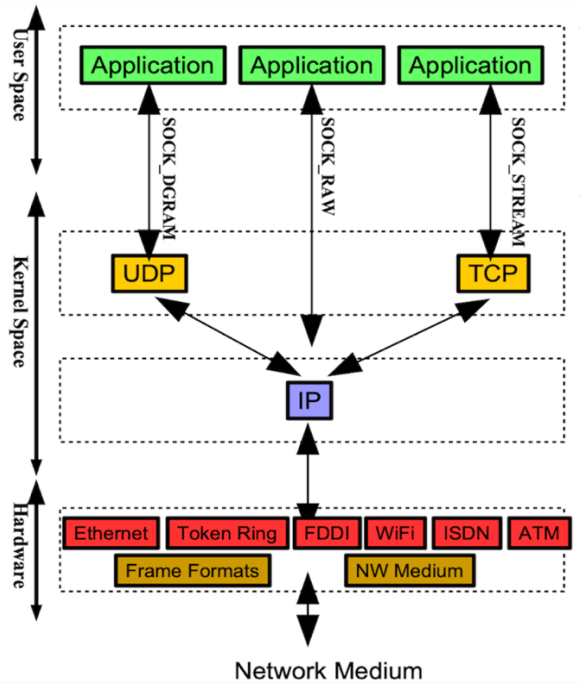
- **Error Handling and Congestion Control:** TCP/IP incorporates mechanisms for error detection, correction, and flow control. TCP manages congestion control to prevent network overload, ensuring efficient data transmission even under high traffic conditions.

- **Support for Diverse Applications:** TCP/IP supports various application layer protocols like HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), and DNS (Domain Name System), enabling a wide range of network services and applications.

- **Foundation of the Internet:** TCP/IP is the backbone of the Internet's architecture. Its design principles and protocols have shaped the development and operation of the Internet, making it possible for billions of devices worldwide to communicate and interact.

TCP/IP Layers:

The TCP/IP suite is structured into four distinct layers, each with specific functions and associated protocols. Here's a detailed breakdown:



1. Application Layer

The Application Layer is where user-interacting applications operate, facilitating communication between software processes and the network. It provides the necessary programming interfaces for building network applications.

- **Functions**
  - Provides interfaces and protocols for software to communicate over the network.
  - Manages end-user services and network data.
  - Supports network transparency, enabling applications to interact with the network infrastructure without knowing its details.
- **Protocols**
  - HTTP/HTTPS (Hypertext Transfer Protocol) Used for transferring web pages on the internet.
  - Telnet Allows remote login to another computer over a network.
  - FTP (File Transfer Protocol) Facilitates file transfer between client and server.
  - SMTP (Simple Mail Transfer Protocol) Used for sending emails.
  - SSH (Secure Shell) Provides a secure channel for operating network services remotely.

- **Addressing**
  - Uses string-based URIs (Uniform Resource Identifiers) such as URLs (Uniform Resource Locators) and URNs (Uniform Resource Names) for identifying resources.
- **Attacks:**
  - Attacking vulnerable services like telnet, ftp, ssh, mysql,smtp, dns, nfs, and so on.
  - Command injection, XSS, CSRF, SQLi. DNS attack surface include lot of attacks like local cache poisoning, fake response attack, rebinding attack, denial of service attacks and so on.

2. Transport Layer

The Transport Layer ensures reliable host-to-host communication. It is responsible for error recovery, flow control, and ensuring complete data transfer.

- **Functions**
  - Provides end-to-end communication services for applications.
  - Manages data segmentation, reassembly, and error correction.
- **Protocols**
  - TCP (Transmission Control Protocol) Ensures reliable, ordered, and error-checked delivery of data.
  - UDP (User Datagram Protocol) Provides a connectionless, lightweight communication service without guaranteed delivery.
  - RAW Allows direct sending of packets without any transport layer formatting, used in specialized scenarios.
- **Addressing**
  - Utilizes 16-bit port numbers to distinguish different services or applications on the same device.
- **Attacks:**
  - SYN Flood attack.
  - TCP reset attack.
  - TCP session hijacking (Mitnick attack).
  - Mitnick attack.

3. Internet Layer

The Internet Layer is responsible for routing packets across the network, ensuring they reach the correct destination.

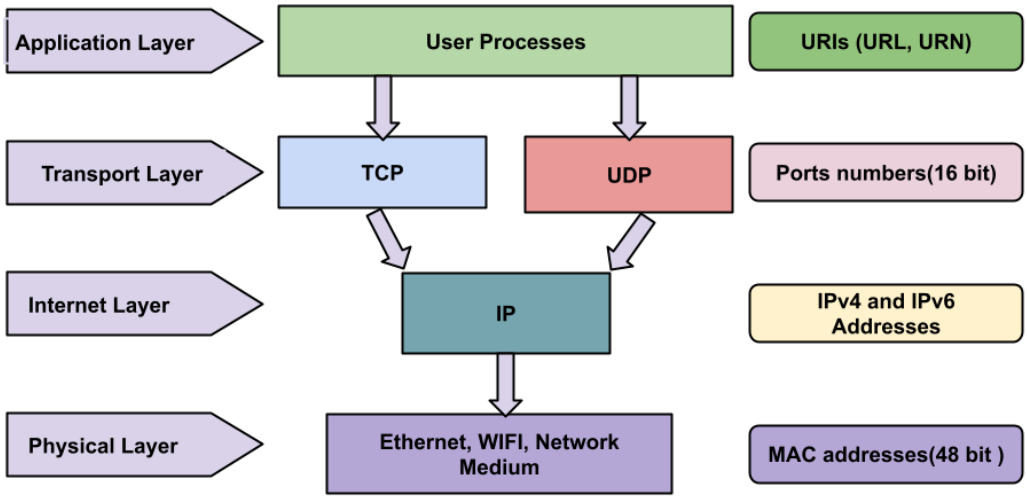
- **Functions**
  - Breaks data into manageable fragments for transmission.
  - Routes data packets across different networks to reach the destination.
  - Handles packet addressing and error checking.
- **Protocols**
  - IP (Internet Protocol) Defines addressing methods and structures the data into packets for transmission.
  - ICMP (Internet Control Message Protocol) Used for error messages and operational information exchange.
  - IGMP (Internet Group Management Protocol) Manages multicast group memberships.
- **Addressing**
  - Uses IPv4 (32-bit addresses) and IPv6 (128-bit addresses) for identifying devices on the network.
- **Attacks:**
  - Attacks using IP fragmentation.
  - ICMP (ping) Flood attack.
  - ICMP redirect attack.
  - IP spoofing attacks.
  - UDP Ping-Pong attack.
  - Smurf attack.

4. Link Layer/ Physical Layer

The Link Layer handles the physical transmission of data over network media. It places packets on the network medium and receives them.

- **Functions**
  - Manages the physical connection between network devices.
  - Ensures data packets are correctly transmitted and received over the network medium.
  - Responsible for hardware addressing and error detection.
- **Protocols**
  - Ethernet is a widely used protocol for wired LANs.
  - Token Ring is an older protocol used for LANs, where devices take turns to send data.
  - FDDI (Fiber Distributed Data Interface) is used for high-speed data transfer over fiber optic cables.
  - ISDN (Integrated Services Digital Network) supports digital transmission of voice and data over ordinary telephone copper wires.
  - SONET (Synchronous Optical Network) is a standard for optical telecommunications.
  - ATM (Asynchronous Transfer Mode) Facilitates high-speed data transfer and supports real-time voice and video.
- **Addressing**
  - Uses 48-bit MAC (Media Access Control) addresses to uniquely identify network interfaces.
- **Attacks:**
  - Man-In-The-Middle attack using arp cache poisoning.
  - MAC spoofing.
  - DHCP starvation.
  - Rogue Access Point attack.
  - Wireless Deauthentication attack.

Addressing Schemes Used in TCP/IP Layers



Addressing on the Application Layer

- The Internet Assigned Numbers Authority (IANA) oversees the assignment of domain names to organizations. These domain names can have multiple strings separated by periods. Each host on the Internet is uniquely identified by a Fully Qualified Domain Name (FQDN), which consists of two parts:

  `hostname.domain-name`

- These FQDNs are stored in a hierarchical and decentralized database that maps hostnames to their corresponding IP addresses. The service that performs the lookup is called Domain Name System (DNS) or Berkley Internet Name Domain (BIND) specified in RFC 882 and RFC 883.

- **URL (Uniform Resource Locator):** A URL identifies a resource located on a specific host within a domain. Its format is:

  `protocol://hostname.domain-name[: port]/path-to-resource`

  For Example: **`http://pucit.pu.edu.pk:80/academics/timetable-pucit.html`**

Organizations can add prefixes to their domain names to define hosts. For example, in the above example `pu.edu.pk` is the domain-name, while `pucit` is the suffix to define its subdomain.

Addressing on the Transport Layer

- **Port Numbers:** The transport layer uses port numbers, which are 16-bit integers, to identify specific processes on a host. For example, a host running both HTTP and SSH services will use port 80 for HTTP and port 22 for SSH.

- **Port Ranges:**
  - **Well-Known/Reserved Ports (0 to 1023):** Permanently assigned to specific applications or services (e.g., SSH uses port 22). These are managed by IANA.
  - **Registered Ports (1024 to 49151): Assigned to application developers less stringently.** IANA maintains a record of these ports.
  - **Dynamic/Private/Ephemeral Ports (49152 to 65535):** Intended for local applications and specified by IANA for temporary use. If an application doesn't bind its socket to a particular port, TCP and UDP assign an ephemeral port from this range.

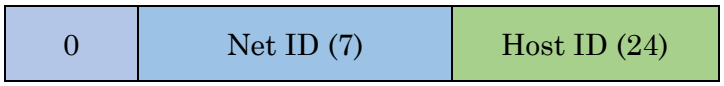
Note: To view details of port assignments, refer to the **/etc/services** file on your Linux machine

Protocol	Port	Service Description
echo	7	• used to test network connectivity by sending a message to a destination and receiving a reply • often used by the ping command to check the reachability of a host
daytime	13	• provides a simple way to obtain the current date and time from a server
ftp-data, ftp	20,21	• used to transfer files between a client and a server over a network • allowing users to upload, download, and manage files
ssh	22	• SSH (Secure Shell) provides a secure, encrypted channel for remote login and command execution • protecting data and credentials from interception • commonly used for secure remote management and file transfers
telnet	23	• protocol for remote command-line access to computers • allowing users to connect to and manage systems over a network • transmits data in plain text
smtp	25	• SMTP (Simple Mail Transfer Protocol) used for sending and routing email between servers and clients
time	37	• provides network-based synchronization of time across devices • allowing systems to set their clocks accurately based on a reference time source
DNS	53	• translates human-readable domain names into IP addresses • enabling users to access websites and services using easy-to-remember names rather than numerical IP addresses
DHCP	67,68	• automatically assigns IP addresses and other network configuration settings to devices on a network • simplifying network management and ensuring devices can communicate efficiently

http	80,8080	• HTTP (Hypertext Transfer Protocol) is the foundation of data communication on the web • facilitating the transfer of web pages and resources between web servers and browsers • defining request and response formats
https	443	• HTTPS extends HTTP with SSL/TLS encryption to secure the transmission of data • ensuring privacy and data integrity in web transactions
ARP	-	• ARP (Address Resolution Protocol) maps IP addresses to physical MAC addresses on a local network • enabling devices to communicate with each other by resolving IP addresses into hardware addresses • ARP doesn't use port number • ARP operates directly at the link layer
ICMP	-	• ICMP (Internet Control Message Protocol) is used for network diagnostics and error reporting • helping manage network communication by sending control messages such as error notifications and echo requests (ping) for connectivity testing
NFS	2049	• NFS (Network File System) allows users to access and manage files on remote servers as if they were on local storage • supporting file sharing and access across a network

Addressing on the Internet Layer

- **Classful Addressing on the Internet Layer (IPv4)**
  - **Class A IP Addresses**

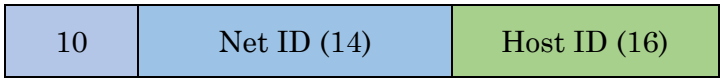


    - **Total Addresses:** $2^7-2 = 126$ *networks*
      - **Range:** 1.0.0.0 to 126.0.0.0
    - **Hosts per Network:** $2^{24}-2 = 16777214$ *hos*
    - **Subnet Mask:** 255.0.0.0 or /8

0.x.x.x and **127.x.x.x** are reserved, and is the reason for subtracting 2 from 2^7

First host address is the NW address and last host address is the broadcast address, and is the reason for subtracting 2 from 2^7

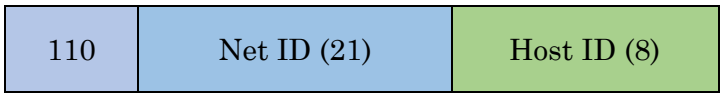
  - **Class B IP Addresses**



    - **Total Addresses:** $2^{14} = 16384$ *networks*
      - **Range:** 128.0.0.0 to 191.255.0.0
    - **Hosts per Network:** $2^{16}-2 = 65534$ *hosts*
    - **Subnet Mask:** 255.255.0.0 or /16

First host address is the NW address and last host address is the broadcast address, and is the reason for subtracting 2 from 2^7

  - **Class C IP Addresses**



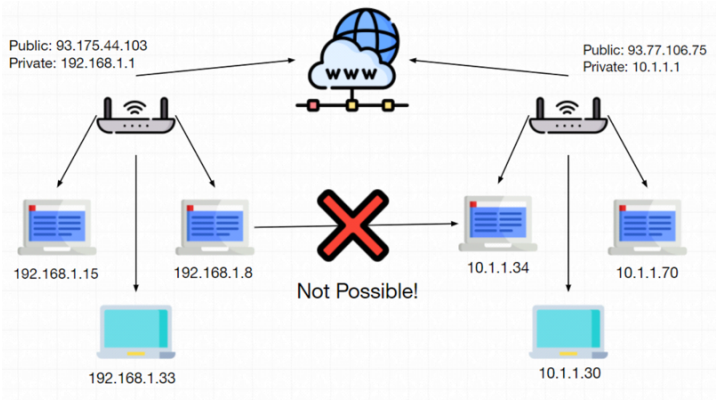
    - **Total Addresses:** $2^{21} = 2097152$ *networks*
      - **Range:** 192.0.0.0 to 223.255.255.0
    - **Hosts per Network:** $2^8-2 = 254$ *hosts*
    - **Subnet Mask:** 255.255.255.0 or /24

First host address is the NW address and last host address is the broadcast address, and is the reason for subtracting 2 from 2^7

Note:

- There exist ***class D*** and ***Class E*** addresses as well. Class D addresses (224.0.0.0 to 239.255.255.255) are used for multi-cast communication, while Class E addresses (225.0.0.0 to 255.255.255.255) are not assigned for public use rather reserved by the IETF for future use.
- The ***network address*** for a specific class is represented with all bits as ZERO in the host portion of the address.
- The ***broadcast address*** for a specific class is represented with all bits as ONES in the host portion of the address.
- Every valid IP Address of a class lie between the Network Address and the Broadcast Address of that class.
- The ***subnet mask address*** for a specific class is represented with all bits as ONES in the network portion and with all bits as ZERO in the host portion. To get the network address you just bit-wise AND the IP address with the subnet mask. All routing is performed based on the NW address.
- **Classless Internet Domain Routing (CIDR):** In 1993, CIDR was introduced that revolutionized IP address allocation and routing by eliminating the rigid boundaries of classful addressing. It offers the advantages of efficient allocation of IP addresses and flexible sub-netting. This helped to meet the growing demand of Internet and the limited address space of IPv4 (4 billion). In CIDR the address 192.168.10.0/25 means the first 25 bits of the IP address are used for the NW portion.

- **Private vs Public Addressing**

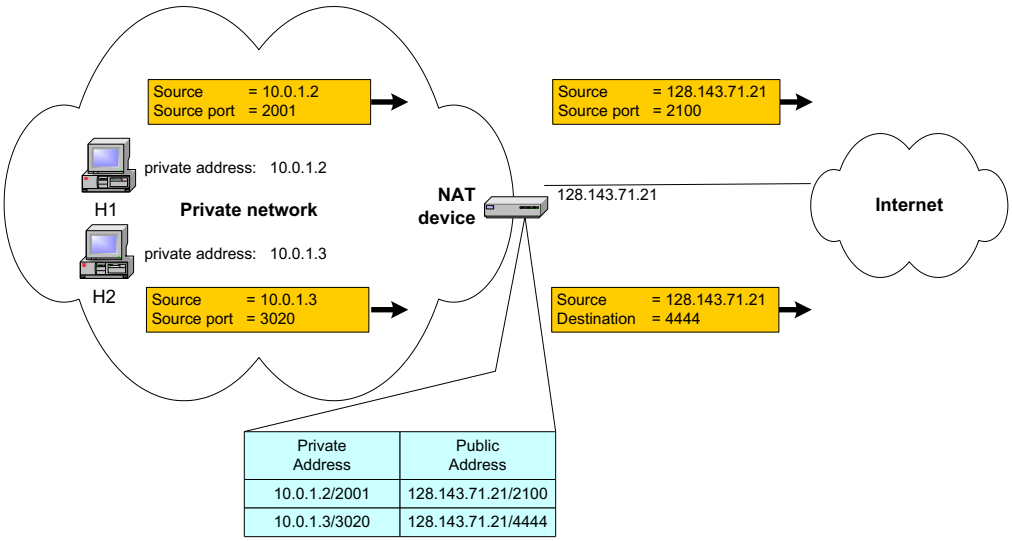


Public IP Addresses as mentioned on the previous page are unique across the entire Internet and are used for communication over the Internet, making them accessible from any device globally. Public IP addresses are routable on the internet and are assigned to devices that need to be reachable from outside the local network, such as web servers, email servers, and network gateways. Devices having public IP addresses are exposed to potential security risks as they are accessible from the Internet. It's crucial to implement proper security measures, such as firewalls and intrusion detection systems, to protect devices and services with public IP addresses.

Private IP Addresses: IETF has designed three address ranges (one for each class) as private, which are commonly used for devices within a local area network, such as computers, laptops, printers and smartphones:

- 10.0.0.0 to 10.255.255.255
- 172.16.0.0 to 172.31.255.255
- 192.168.0.0 to 192.168.255.255

The devices having private IP addresses are non-routable, i.e., not directly exposed to the public Internet, providing a layer of security by keeping internal devices hidden from external threats. They can only be used either on a fully disconnected NW or on a NW behind firewall. Now a **100$** question is: How can a device having a private IP address accesses the resources on the Internet having public IP addresses? Answer is: **NetWork Address Translation** (NATing), that allows a single device called gateway computer (router) having a public IP address to act as an agent between the Internet and the private NW. A gateway computer is an entry/exit point in a LAN, that receives incoming requests from devices having private IP addresses and send it to the Internet with its own public IP address. So, this means that a single public IP address can represent an entire group of computers on the Internet.



"CIDR and NATing has significantly extended the useful life of IPv4"

Addressing on the Network Layer (IPv6)

- **IPv6 Address Format:**
  - IPv4 addresses are 32 bits, allowing for 4 billion unique addresses.
  - IPv6 addresses are 128 bits, allowing for over 340 undecillion (340 trillion trillion trillion) unique addresses.
  - Example of an IPv6 address: **F000:0:0:0:0:0:A:1**.
  - Notational convenience: A sequence of zeros can be represented with a double colon (::), making the above address **F000::A:1**.
  - Only one instance of the double-colon notation can be used in an IPv6 address.

- **Special IPv6 Addresses:**
  - Loopback address: **::1** (equivalent to IPv4's **127.0.0.1**).
  - Wildcard address: **0::0** or simply **::** (equivalent to IPv4's **0.0.0.0**).
  - IPv4-mapped IPv6 address: Represents an IPv4 address within an IPv6 address, using the format **::FFFF:204.152.189.116** for the IPv4 address **204.152.189.116**.

How an IP address is assigned to your computer:

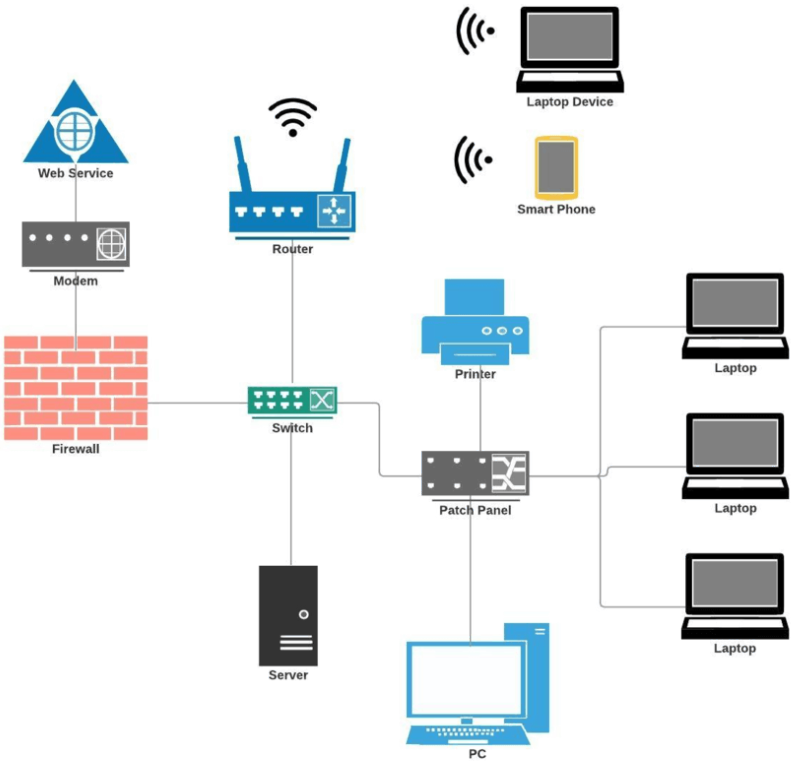
The four main TCP/IP configuration parameters, IP address, Broadcast address, Subnet mask and IP of gateway can be assigned manually using the `ifconfig/ipconfig` command or using the Dynamic Host Configuration Protocol (DHCP) service running on your PTCL broadband router. DHCP is a client/server protocol for automatically assigning network configuration parameters, such as IP addresses, subnet mask, default gateway, DNS for interfaces and services.The **DORA Process of DHCP** is a four step process as described below:

- The virtual machine (client) once booted will send a DHCP **DISCOVER** broadcast message to all the machines on LAN at port#67, to find the DHCP server in the network.
- If a DHCP service is running on any of the machines in our LAN (in our case the PTCL broadband router) which is listening at port#67 will reply with a DHCP **OFFER** message (containing IP, subnet mask, gateway), which is sent directly to the client using its MAC address.
- The client machine then sends a DHCP **REQUEST** message to the DHCP server.
- When the DHCP server receives the DHCP request, it stores the client IP address in its database and broadcast the DHCP **ACK** message to inform the client that it can use the offered information (TCP/IP parameters)

- **Addressing on the Physical Layer (MAC Addresses)**
  - **MAC Address Format:**
    - A 48-bit address is used on the physical layer.
    - Divided into two parts:
      - **Organizationally Unique Identifier (OUI):** The most significant 3 bytes (e.g., **00-50-56**).
      - **Network Interface Specific Identifier:** The least significant 3 bytes (e.g., **C0-00-01**).
  - **MAC Address Assignment:**
    - Manufacturers request an OUI from the IEEE to ensure a unique prefix for their devices.
    - The manufacturer then assigns a unique identifier to the remaining 3 bytes for each device.
    - This ensures a globally unique MAC address for every device.
  - **Routing and Address Resolution:**
    - If the destination IP address is outside the local network, the packet is sent to a configured gateway for routing.
    - If the destination IP address is within the same local network, the Address Resolution Protocol (ARP) is used to find the corresponding MAC address from the IP address.

Organizationally Unique Identifier 00-50-56	Network Interface Specific Identifier C0-00-01

Networking Devices



- **Modem:** A modem (modulator-demodulator) converts digital data from a computer into analog signals for transmission over telephone lines or cable systems, and vice versa. It acts as a bridge between the local network and the internet service provider (ISP). Typically used to connect to the internet, especially in DSL or cable broadband connections.

- **Repeater:** A repeater regenerates and amplifies signals to extend the transmission distance over a network without degrading the quality. Employed in networks where the signal needs to travel long distances or through obstacles, such as in large buildings or across geographic areas.

- **Hub:** A hub is a basic networking device that connects multiple devices in a network segment model. It broadcasts incoming data packets to all connected devices, regardless of their intended destination. This can lead to network congestion and collisions, making hubs less efficient for larger or more complex networks. Typically used in simple, small networks where high performance and security are not critical.

- **Switch:** A switch connects multiple devices in a network, and forwards data packets only to the device that needs them, based on MAC (Media Access Control) addresses. This reduces unnecessary traffic and collisions, improving network efficiency and performance.

- **Router:** A router is used to connect multiple networks, such as home network to the Internet, and directs data packets between different network segments based on their IP addresses. Routers determine the best path for data to travel from one network to another, typically within the same network type (e.g., from one subnet to another or from a local area network to the internet). Routers also provide network address translation (NAT), which allows multiple devices on a local network to share a single public IP address.

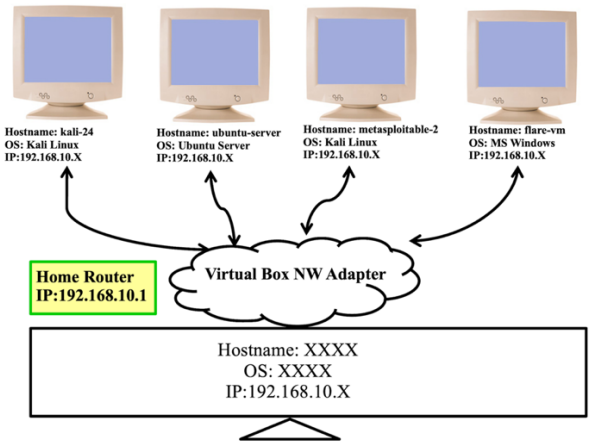
- **Gateway:** A gateway act as an entry or exit point in a network, allowing communication between different types of networks to perform protocol translation. It is a more complex device than a router, as it can perform protocol translation and traffic filtering. (A router is about directing traffic between similar networks, while a gateway facilitates communication between dissimilar networks)

- **Bridge:** A bridge is used to connect two or more network segments, filtering traffic and reducing collisions by forwarding packets only to the segment where the destination device resides (based on their MAC addresses). Remember, a bridge is used to extend or segment a single network, while a router is used to connect multiple networks and manage traffic between them.

- **Access Point (AP):** An access point connects wireless devices to a wired network using Wi-Fi standards (WEP, WPA, WPA2, WPA3). Used in wireless networks to enable devices such as laptops, smartphones, and tablets to connect to the network. It is often combined in the router of your home network.

- **Firewall:** A firewall is a security device that monitors and controls incoming and outgoing network traffic based on predetermined security rules. It can be hardware-based, software-based, or a combination of both and is used to filter traffic based on IP addresses, port numbers, and protocols.

Creating Network of Virtual Machines



We have already created virtual machines and installed different Operating Systems on them inside the Virtual Box. The next step is making a network of these machines or making these machines part of an existing network. For this, on the VirtualBox Manager, select the machine and click the Network settings, where you can see, you have the option to add up to 4 Network cards with each machine. By default, one NW card is enabled and the default NW adapter is the Bridged Adapter. The others are:

- Host-only Adapter
- Internal Network
- Bridged Adapter
- NAT Network
- Cloud Network

Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	–	–
Internal	–	–	+	–	–
Bridged	+	+	+	+	+
NAT	+	Port forward	–	+	Port forward
NATservice	+	Port forward	+	+	Port forward

For details visit: https://www.virtualbox.org/manual/ch06.html

Advantages of Bridged Adapter:

- Each machine will have its own unique private IP address, assigned by the DHCP server of your home router.
- All virtual machines can communicate with the host and the host can communicate with all the virtual machines.
- All virtual machines can communicate with each other.
- All virtual machines can access the Internet via the gateway computer
- Last but not the least the outside world, i.e., any machine that is not in the same local area network cannot access the virtual machines as they are having private IP addresses. However, using Port Forwarding this can also be done. For details about port forwarding read https://www.virtualbox.org/manual/ch06.html#natforward

Verifying the TCP/IP Configurations of VMs

a. ifconfig

ifconfig/ipconfig is a network configuration utility that is traditionally used on Unix-like and Windows operating systems respectively to configure and display network interface parameters. In modern Linux distributions, ifconfig has been largely replaced by the ip command from the iproute2 package, but it remains available for compatibility and ease of use.

- **Basic Syntax**

```
$ ifconfig [interface] [options]
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.10  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::a00:27ff:fe8e:9b3b  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:8e:9b:3b  txqueuelen 1000  (Ethernet)
        RX packets 1000  bytes 1234567 (1.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 500  bytes 654321 (654.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

  - **Output**: The output of ifconfig includes several fields for each network interface:
    - **Interface Name:** e.g., eth0, wlan0
    - **IP Address:** The assigned IP address, e.g., inet addr:192.168.1.10
    - **Netmask:** The subnet mask, e.g., Mask:255.255.255.0
    - **Broadcast Address:** The broadcast address, e.g., Bcast:192.168.1.255
    - **MAC Address:** The hardware address, e.g., HWaddr 08:00:27:8e:9b:3b
    - **MTU:** Maximum Transmission Unit, e.g., MTU:1500
    - **RX/TX Packets:** Number of packets received/transmitted, e.g., RX packets:1000
    - **Errors:** Errors encountered, e.g., errors:0
- **Display Detailed Information:** To get detailed information about a specific network interface.

```
$ ifconfig eth0
```

- **Assign an IP Address:** To set or change the IP address of a network interface

```
$ sudo ifconfig eth0 192.168.1.10
```

- **Set the Netmask:** To define the subnet mask for a network interface.

```
$ sudo ifconfig eth0 netmask 255.255.255.0
```

- **Set the Broadcast Address:** To configure the broadcast address for the network interface

```
$ sudo ifconfig eth0 broadcast 192.168.1.255
```

- **Enable or Bring Up an Interface:** To activate a network interface, making it operational

```
$ sudo ifconfig eth0 up
```

- **Disable or Bring Down an Interface:** To deactivate a network interface, stopping its operations

```
$ sudo ifconfig eth0 down
```

b. Ping:

ping (Packet Internet Gropper) is a basic network utility used to test the reachability of a host on a network and to measure the round-trip time for messages sent from the originating host to a destination computer. It is commonly used for troubleshooting and network diagnostics.

- **Purpose of ping**
  - **Network Reachability**: Determines whether a host (e.g., a computer or server) is reachable over the network.
  - **Round-Trip Time Measurement**: Measures the time it takes for a packet to travel from the source to the destination and back.
  - **Network Troubleshooting**: Helps diagnose network issues such as latency, packet loss, or connectivity problems.

- **How ping Works**

  It uses the Internet Control Message Protocol (ICMP) to send Echo Request messages to a target host. The host then responds with Echo Reply messages. The basic steps are:

  - **Send Echo Request**: ping sends ICMP Echo Request packets to the target host.
  - **Receive Echo Reply**: The target host replies with ICMP Echo Reply packets.
  - **Measure Round-Trip Time**: ping measures the time taken for the Echo Reply to return and displays it.
  - **Display Results**: Provides statistics on packet loss and round-trip time.

- **Basic Syntax**

```
$ ping [options] [hostname or IP address]
```

- **Usage**
  - **Ping a Host**: Sends ICMP Echo Request packets to example.com and displays the response time as output. On Linux/macOS, ping will continue sending packets until you stop it with Ctrl+C. On Windows, you need to specify the number of packets with -n to stop.

```
$ ping google.com
```

  - **Output**

```
PING google.com (142.250.72.14) 56(84) bytes of data.
64 bytes from 142.250.72.14: icmp_seq=1 ttl=117 time=12.3 ms
64 bytes from 142.250.72.14: icmp_seq=2 ttl=117 time=11.8 ms
64 bytes from 142.250.72.14: icmp_seq=3 ttl=117 time=12.0 ms
64 bytes from 142.250.72.14: icmp_seq=4 ttl=117 time=11.9 ms
64 bytes from 142.250.72.14: icmp_seq=5 ttl=117 time=12.1 ms
^C
--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 11.810/12.061/12.309/0.233 ms
```

- **PING google.com (142.250.72.14) 56(84) bytes of data.**
  - This indicates that the ping command is sending ICMP Echo Request packets to google.com with a payload size of 56 bytes (which expands to 84 bytes when including headers).

- **64 bytes from 142.250.72.14: icmp_seq=1 ttl=117 time=12.3 ms**

- 64 bytes from 142.250.72.14: This shows the size of the ICMP Echo Reply packet and the IP address of the destination.
- icmp_seq=1: This is the sequence number of the ICMP request (used to track the order of packets).
- ttl=117: The Time To Live (TTL) value of the packet, which decrements with each hop (router) it passes through.
- time=12.3 ms: The round-trip time it took for the packet to travel from your machine to the destination and back.

- **^C**
  - This indicates that the ping command was interrupted manually (usually by pressing Ctrl+C).

- **--- google.com ping statistics --- :** This section provides a summary of the ping results.
  - **5 packets transmitted, 5 received, 0% packet loss, time 4004ms**
  - 5 packets transmitted: The number of ICMP Echo Requests sent.
  - 5 received: The number of ICMP Echo Replies received.
  - 0% packet loss: Indicates that no packets were lost during transmission.
  - time 4004ms: The total time for the ping operation.

- **rtt min/avg/max/mdev = 11.810/12.061/12.309/0.233 ms**
  - min: The minimum round-trip time observed.
  - avg: The average round-trip time.
  - max: The maximum round-trip time observed.
  - mdev: The mean deviation (standard deviation) of the round-trip times, indicating variability.

- **Ping an IP Address**: Pings the device with the IP address 192.168.1.1.

```
$ ping 192.168.1.1
```

- **Ping with a Specific Number of Packets**: On Linux/macOS, -c 4 sends 4 packets. On Windows, use -n 4.

```
$ ping -c 4 google.com
```

- **Ping with a Specific Packet Size**: On Linux/macOS, -s 128 specifies a packet size of 128 bytes. On Windows, use -l 128

```
$ ping -s 128 google.com
```

- **Ping with a Specific Timeout**: On Linux/macOS, -W 5 sets a timeout of 5 seconds for each response. On Windows, use -w 5000.

```
$ ping -W 5 google.com
```

- **Ping with a Custom Interval**: On Linux/macOS, -i 2 sets an interval of 2 seconds between packets, Sends packets every 2 second

```
$ ping -i 2 google.com
```

c. arp

The arp command in Linux is a network utility tool that is used to display, add and remove entries in the Address Resolution Protocol (ARP) cache, which is a temporary storage for IP to MAC address mappings. These mappings are crucial for network communications.

```
$ arp

Address          HWtype         HWaddress        Flags  Mask  Iface
192.168.193.53   ether        3a:48:8e:6f:8b:4c  C             eth0
```

The output shows the IP address (192.168.193.53), the corresponding MAC address (3a:48:8e:6f:8b:4c), and the network interface (eth0) on which the ARP entry is located. You can add an entry, remove/modify an existing entry in the arp cache. For details read the man pages.

d. route

We know that router is a device responsible for forwarding NW traffic. When datagrams arrive at a router, the router must determine the best way to route them to their destination. The route command in Linux is used to show and manipulate the Kernel routing table. Running the route command without any options displays the routing table entries as shown below:

```
$ route

Kernel IP routing table

Destination     Gateway         Genmask         Flags  Metric  Ref   Use   Iface
default         192.168.193.53  0.0.0.0         UG     100     0     0     eth0
192.168.193.0   0.0.0.0         255.255.255.0   U      100     0     0     eth0
```

The above output shows us how the system is currently configured. If a packet comes into the system and has a destination in the range 192.168.193.0 through 192.168.193.255, then it is forwarded to the gateway 0.0.0.0, i.e., our system will not route these packets. If the destination is not in this IP address range, then it is forwarded to the default gateway (in this case 192.168.193.53), and the system will determine how to forward the traffic on the next step towards its destination.

You can use the add and del options of the route command to add, delete, modify the routing table. For details read the man pages.

DNS/BIND

Overview:

This sub-section deals with something that is known as Domain Name System. The way humans can be identified in many ways; by their first name, last name, nick name as well as by their National Identity numbers. Similarly, the hosts on the Internet can also be identified either by their host names or by their IP addresses. The string-based hostnames are easy to remember by humans, but difficult to process by machines. Therefore, the Internet need to have a directory system that can map a hostname to an IP address, which is analogous to a telephone directory which maps the names of our relatives and friends to their respective telephone numbers. DNS is a hierarchical decentralized naming service that runs on hundreds and thousands of computers all around the globe and is responsible for mapping hostnames with their IP addresses and vice versa.

In 1969, when Internet was born; four hosts located at

- University of California Loss Angeles,
- University of Santa Barbra,
- University of Utah, and
- Stanford Research Institute

were connected via 56 Kbps phone lines. Remembering those four IP addresses was not an issue. In 1971, just after three years, the count of Internet hosts moved to thousands. At that time the idea of a host lookup table was introduced to keep the name and IP mapping. On UNIX based machines this look up table is kept in the file **`/etc/hosts`**, while on Microsoft Windows based machines it is there in the **`C:\Windows\System32\drivers\etc\hosts`** file. Each line of these files contains `<ip> <hostname>` mapping. You can populate these files with all the ip-name mappings in all the machines of your lab setup, and then you can ping the machines using the hostnames as well other than their IPs.

This technique works fine in the early days of Internet, but suffers with limitations like:

- If you want to add a new host, you need to put its hostname and IP mapping entry in the lookup tables of all the existing hosts.
- If the IP or name of a host changes, you need to reflect the change in the lookup tables of all the hosts.

Lot of improvements were proposed but due to sheer size of Internet and its rate of growth, no promising solution came up till 1983. In 1983, the first RFC for DNS was published. Request for Comments are actually documents that specify and discuss current and developing standards for the Internet. These documents are first published as drafts and are made available to all Internet users for review and feedback. After review some of these become Standard. To dig out the real details about how DNS works, you can read **`RFC:882`** and **`RFC:883`** from the following links of Internet System Consortium web site. The Internet Engineering Task Force (IETF) has ultimate responsibility for RFC documents, and maintains a complete list of them in their online RFC directory.

https://www.isc.org/rfcs/

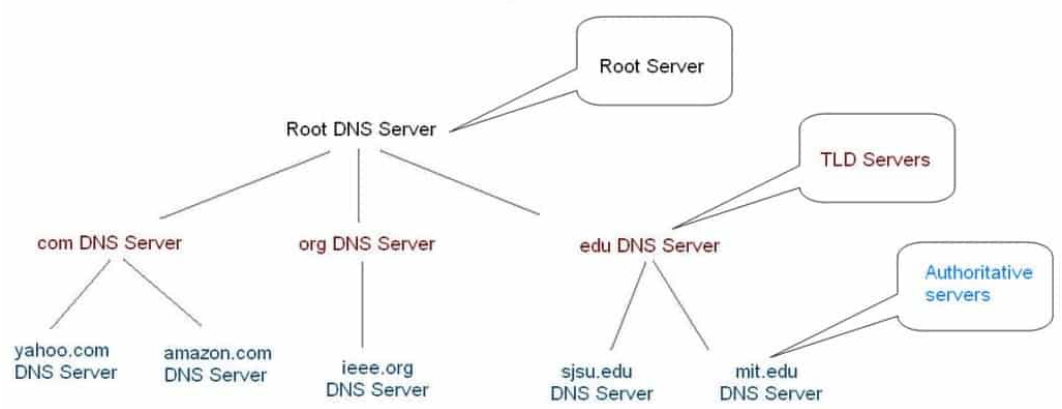
https://powerdns.org/dns-camel/

Fully Qualified Domain Name:

- Simply speaking, a Fully Qualified Domain Name (FQDN) can be divided into two parts `<hostname.domain-name>`, the hostname identifies a host on the Internet, while domain name can be two or more strings separated by a period.
- An example domain name is *`pu.edu.pk`*.
- Internet Assigned Numbers Authority (https://www.iana.org/) is a central authority that manages the assignment of these domain names to organization.
- An organization (like Punjab University) can add prefixes to its domain name to define its hosts, e.g., *`ds.pu.edu.pk`*.
- Similarly, an organization can add suffixes to its domain name to define its resources, e.g., *`pu.edu.pk/faculties`*

DNS Hierarchical Tree:

There are millions and millions of IP addresses and all the names and IP addresses of these hosts need to be saved somewhere. It will be very inefficient and also not reliable to have just one computer store such a huge amount of information. The solution is to distribute the information among many computers called Domain Name Servers.



- You can think of the Domain Name Space as an inverted tree with the root at the top, represented by a *period* at the very end of a domain name, which we normally omit. There are thirteen root servers that maintain information about the TLD servers (i.e., root servers know where the TLD servers are).
- Below the root we have over 700 Top Level Domains. There exists *generic Top Level Domains (gTLDs)* like `.com, .edu, .org, .gov, .mil` and so on.
- Similarly, there are *country code Top Level Domains (ccTLDs)* like `.pk, .uk, .us, .cn, .in, .me` and so on.
- Below TLDs, we have Second Level Domains (SLDs), also called Domain Name Servers, which maintains hostname-IP mapping information in files called zone files. These Domain Name Servers can be Authoritative or Non-Authoritative. If a DNS server has the completed updated zone files of a domain, then that DNS server is called *Authoritative name server* for that domain.

DNS Related Utilities:

In order to install DNS related utilities, use the following Linux command:

```
$ sudo apt-get install dnsutils
```

a. host

It is a utility that performs DNS lookups. Normally used to convert names to IP address and vice versa. For details read the man pages.

```
$ host arifbutt.me
arifbutt.me has address 68.65.120.238
arifbutt.me mail is handled by 0 mail.arifbutt.me

$ host google.com
google.com has address 142.250.181.14
google.com has IPv6 address 2a00:1450:4019:809::200e
google.com mail is handled by 10 smtp.google.com
```

b. nslookup

Name Server Lookup is a utility that is available for all UNices as well as for Microsoft platforms:

```
$ nslookup arifbutt.me
Server: 192.168.10.1
Address: 192.168.10.1#53

Non-Authoritative answer:
Name: arifbutt.me
Address: 68.65.120.238

$ nslookup google.com
Server: 192.168.10.1
Address: 192.168.10.1#53

Non-Authoritative answer:
Name: google.com
Address: 142.250.181.14
Name: google.com
Address: 2a00:1450:4019:809::200e
```

Description of output: In the above outputs 192.168.10.1 is the address of the DNS server that our system is configured to use to translate domain names to IP addresses and 53 is the standard port number on which DNS servers accept the queries. The Non-Authoritative answer means the DNS server which has responded does not have the zone file, rather have answered the query form its cache.

c. dig

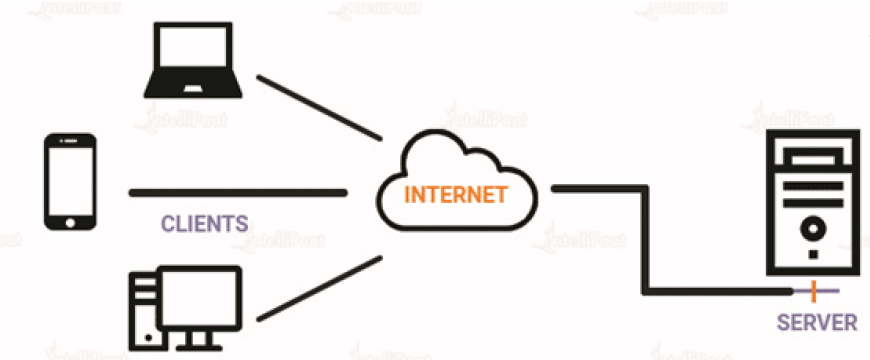
The third DNS lookup utility is dig that stands for Domain Information Groper. It is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use, and clarity of output. Other lookup tools tend to have less functionality than dig. For details read the man pages.

```
sara@pnap:~$ dig google.com

; <<>> DiG 9.18.18-0ubuntu0.22.04.2-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6286
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             287     IN      A       142.250.180.238

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Tue May 21 15:34:04 CEST 2024
;; MSG SIZE  rcvd: 55

sara@pnap:~$
```

Description of output:

- The first line displays the dig command version
- The HEADER section summarizes the DNS query and response details.
- The OPT PSEUDOSECTION displays advanced data
- The QUESTION SECTION displays the query data that was sent:
    - The first column is the domain name queried.
    - The second column is the query type (IN = Internet).
    - The third column specifies the record (A means IPv4, AAAA means IPv6).
- The ANSWER SECTION displays the response:
    - The first column lists the domain name that was queried.
    - The second column is the Time to Live in seconds for which a receiving server can cache this information. After this the mapping becomes invalid and any query must be sent again to the authoritative server.
    - The third column shows the query class. In this case, IN stands for Internet.
    - The fourth column displays the query type. In this case, A stands for IPv4 address, AAAA means IPv6 address.
    - The final column displays the IP address associated with the domain name.

A hands-on Practice of Different Client-Server Services



Before proceeding any further, let us try to make our hands dirty by using different services that runs on our Ubuntu Server machine and access them using respective client programs from our Kali Linux machine. I will be referring the machine running Ubuntu Server OS as Server machine, since it will be running different Server programs (services) like `netcat`, `xinetd`, `telnetd`, `sshd`, `vsftpd`, `httpd`, `mysql` and so on. Similarly, I will be referring the machine running Kali Linux OS as Client machine, since it will be running different corresponding client programs like `netcat`, `telnet`, `ssh`, `ftp`, `curl`, `fire-fox` and so on. The TCP/IP parameters of both machines can be checked using `ifconfig` command and the screenshots are shown below:

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.8.112  netmask 255.255.255.0  broadcast 192.168.8.255
        inet6 fe80::7f59:d39:b815:649  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:78:1c:8a  txqueuelen 1000  (Ethernet)
        RX packets 145  bytes 47296 (46.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 73  bytes 9098 (8.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
root@ubuntuserver:~# ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:d2:4a:cf
          inet addr:192.168.8.111  Bcast:192.168.8.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fed2:4acf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11666 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3686 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10329305 (10.3 MB)  TX bytes:264319 (264.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:553 errors:0 dropped:0 overruns:0 frame:0
          TX packets:553 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:62381 (62.3 KB)  TX bytes:62381 (62.3 KB)
```

The Extended Internet Daemon (Super Server)

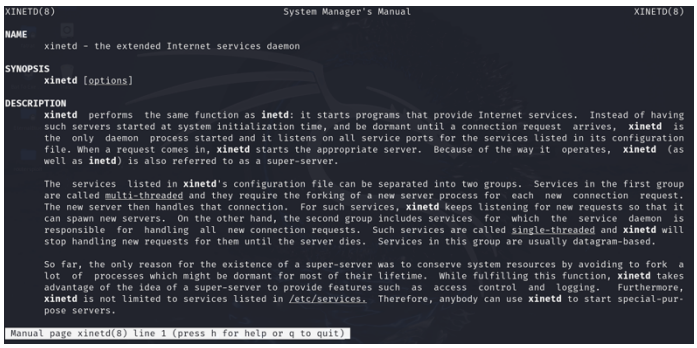
- Before proceeding, let me check the contents of the `/etc/services` file, which (on all Linux based machines) provides a mapping between different Internet services and their underlying assigned port numbers and protocol types. Let us check the contents of this file:
  `# less /etc/services`
- A portion of the output is shown in the screenshot. Students are advised to go the entire file and view different services running on different ports for better understanding.
- On all Linux based machines there is a service called **`xinetd`** (Extended Internet Daemon) also called Super Daemon that is responsible to control different services like:
  - **echo(7):** Upon connection the server returns an identical copy of data to the client.

```
root@ubuntuserver:~# cat /etc/services
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, officially ports have two entries
# even if the protocol doesn't support UDP operations.
#
# Updated from http://www.iana.org/assignments/port-numbers and other
# sources like http://www.freebsd.org/cgi/cvsweb.cgi/src/etc/services .
# New ports will be added on request if they have been officially assigned
# by IANA and used in the real-world or are needed by a debian package.
# If you need a huge list of used numbers please install the nmap package.

tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
```

- **discard(9):** The data sent to the server is discarded (like `/dev/null`) and nothing is returned.
- **daytime(13):** Upon connection the server returns ASCII character string of current date and time (Number seconds passed since UNIX epoch, i.e., 00:00 (midnight) 1st Jan 1970 GMT)
- **chargen(19):** Upon connection the server starts sending arbitrary characters to the host until connection terminates
- **telnet(23):** A remote login protocol
- **time(37):** Upon connection the server returns time as 32-bit unsigned integer in binary format containing the number of seconds passed since Internet epoch, i.e., 00:00 (midnight) 1st January 1900 GMT

- The **Extended Internet Daemon (`xinetd`)** is also known as super server. If you do not want the services (like daytime, echo, telnet, etc) to be started at system initialization time by `systemd` or `init`, and be dormant until a connection request arrives, `xinetd` is the only daemon process started. When a request comes in for any one of the above services, `xinetd` (does not run on any port rather listens on the ports specified for the services it manages) starts the appropriate service. Moreover, there is a utility called `systemctl` that is used to manage different services running on your Linux machines. Let us check the man pages of `xinted` and `systemctl` on our Ubuntu Server machine:

  `# man xinetd`
  `# man systemctl`



- Let us check the status of `xinted` super server on our Ubuntu Server machine, and if it is not running, we need to start it using the following command:

  `# systemctl status/start/stop/enable/disable xinetd`

```
root@ubuntuserver:~# systemctl status xinetd
● xinetd.service - LSB: Starts or stops the xinetd daemon.
     Loaded: loaded (/etc/init.d/xinetd; bad; vendor preset: enabled)
     Active: active (running) since Mon 2024-09-02 09:30:55 PKT; 36min ago
       Docs: man:systemd-sysv-generator(8)
    Process: 1198 ExecStart=/etc/init.d/xinetd start (code=exited, status=0/SUCCESS)
      Tasks: 1
     Memory: 2.4M
        CPU: 27ms
     CGroup: /system.slice/xinetd.service
             └─1351 /usr/sbin/xinetd -pidfile /run/xinetd.pid -stayalive -inetd_compat -inetd_ipv6
```

- To check out what all services are managed by `xinetd` on your machine, you can check the contents of the `/etc/xinetd.d/` directory, that contains *configuration files* of all the services which are being controlled by `xinetd` server.

```
# ls /etc/xinetd.d
chargen    daytime    discard    echo  telnet     time  vsftpd
```

- Let us check out the configuration files of these services, and *enable* them by setting the `disable=no`, in their configuration files (if it is set to yes)

```
# vim /etc/xinetd.d/echo
```

```
# default: off
# description: An xinetd internal service which echo's characters back to
# clients.
# This is the tcp version.
service echo
{
        disable         = no
        type            = INTERNAL
        id              = echo-stream
        socket_type     = stream
        protocol        = tcp
        user            = root
        wait            = no
}

# This is the udp version.
service echo
{
        disable         = no
        type            = INTERNAL
        id              = echo-dgram
        socket_type     = dgram
        protocol        = udp
        user            = root
        wait            = yes
}
```

- Now, let us verify if these services are running on Ubuntu Server. For this we can check the status of the ports on which these services are running using **netstat** utility. Netstat (network statistics) is a command-line tool used for monitoring network connections and statistics. It provides detailed information about network interfaces, routing tables, active connections, and network statistics, making it useful for network troubleshooting, system monitoring, and security analysis. You can install

```
# apt-get install net-tools
# man netstat
# netstat -tup
```

```
root@ubuntuserver:~# netstat -tup
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 192.168.8.111:ssh       192.168.8.112:34586     ESTABLISHED 2766/0
udp6       0      0 localhost:54374         localhost:51650         ESTABLISHED 2606/(pinger)
udp6       0      0 localhost:51650         localhost:54374         ESTABLISHED 1397/(squid-1)
```

```
# netstat -tl
```

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:time                  *:*                     LISTEN
tcp        0      0 *:echo                  *:*                     LISTEN
tcp        0      0 *:discard               *:*                     LISTEN
tcp        0      0 *:mysql                 *:*                     LISTEN
tcp        0      0 *:55788                 *:*                     LISTEN
tcp        0      0 *:daytime               *:*                     LISTEN
tcp        0      0 *:sunrpc                *:*                     LISTEN
tcp        0      0 *:chargen               *:*                     LISTEN
tcp        0      0 *:48053                 *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 *:telnet                *:*                     LISTEN
tcp        0      0 *:smtp                  *:*                     LISTEN
```

- We are all set now to check out the working of the `chargen, daytime,discard,echo,time` services running on the Ubuntu Server from our Kali Linux machine using some client program. Let us use `netcat` utility, famously known as **Swiss Army Knife** of networking due to its wide range of functionalities. `Netcat` (often abbreviated as nc) is a versatile networking tool that can read and write data across network connections using the TCP/IP protocol suite. `Netcat` is used for network debugging, security testing, and data transfer, among other tasks. You need to install it on your Kali Linux as well as Ubuntu Server machines
  # apt-get update
  # apt-get install netcat-openbsd
  # apt-get install netcat-traditional
- The first command will install `nc.openbsd`, while the second will install `nc.traditional`. You may need to set the appropriate version, which in our case is `nc.traditional` using the following command on both the machines:
  # update-alternatives --config nc
- To make above change permanent, we need to make a soft link using the following command. This way nc will always points to netcat-traditional.
  # ln -sf /bin/nc.traditional /etc/alternatives/nc
- When you use `nc` to connect to server, by default it uses the TCP connection, you can use the **-u** switch to use UDP connection. From your Kali Linux machine, you can use the following command to connect to the appropriate service running on the specific port of ubuntu server:
  $ nc <ip of ubuntuserver> <port>

Note: Before proceeding any further, students must practice starting/stopping above services on ubuntu server machine and accessing them from the kali Linux machine using **nc**.

- In order to make a network connection between two nodes, one of them will need to be listening on a specific port, while the other initiates the connection to that port. Students should also try to use `nc` as a server to listen on a specific port to be connected by a client program. Run the listener on ubuntu server and then try to connect from Kali machine using nc. Whatever you write on Ubuntu Server's terminal will be written/echoed on Kali's terminal and vice versa. This proves that the `netcat` utility is used to read and write data across network connections. ☺

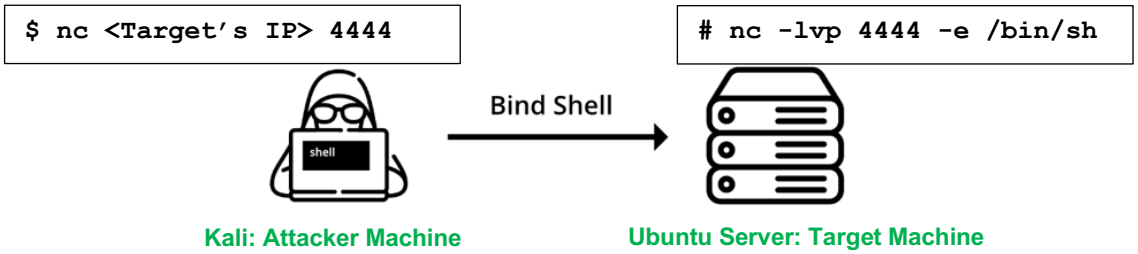
  # nc -l -p 54154 **$ nc <ip of ubuntuserver> 54154**

- Once you are communicating using `nc` with above commands, check the status of ports using netstat on both machines to have a clear understanding of the two processes communicating via TCP sockets.

Bind Shell vs Reverse Shell using **nc**

Bind Shell:

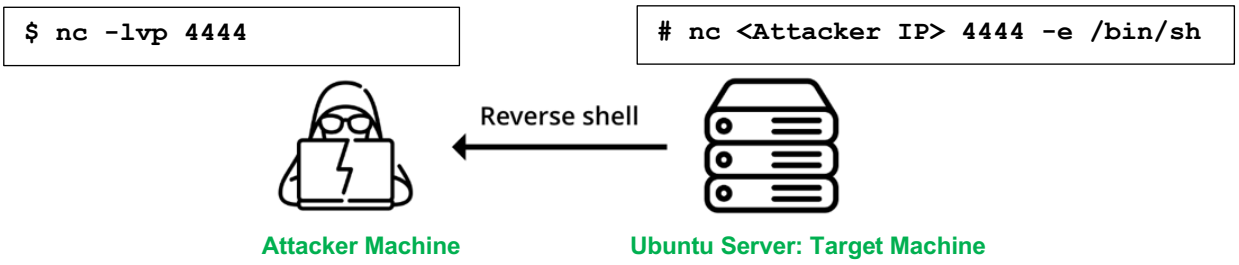
- Bind Shells have the listener running on the target and the attacker connects to the listener in order to gain remote access to the target system.
- In Bind shell, the attacker finds an open port on the server/ target machine and then tries to bind his/her shell to that port using say `netcat` utility.
- The attacker must know the IP address of the victim before launching the Bind Shell.
- In Bind shell, the listener is ON on the target machine and the attacker connects to it.
- Bind Shell sometimes will fail, because modern firewalls don't allow outsiders to connect to open ports.



Attacker executing bind from his machine to server

Reverse Shell:

- In the reverse shell, the attacker has the listener running on his/her machine and the target connects to the attacker with a shell. So that attacker can access the target system.
- In the reverse shell, the attacker opens his own port as a server, so that victim can connect to that port for successful connection.
- The attacker doesn't need to know the IP address of the victim, because the attacker is going to connect to our open port.
- The Reverse shell is opposite of the Bind Shell, in the reverse shell, the listener is ON on the Attacker machine and the target machine connects to it.
- Reverse Shell can bypass the firewall issues because the target machine tries to connect to the attacker, so the firewall doesn't bother checking packets.



Server tries to connect to Attacker machine

The Telnet Service

- Telnet is an application layer protocol that can be used on Local Area Network or on the Internet as a remote login facility. It provides a bidirectional, interactive, text-oriented communication facility. It was developed in 1969 and transmits all the data including the passwords in clear over the network, i.e., no encryption is done. Infect it is very easily exploitable and all the traffic can be intercepted by the man in the middle attack. So due to these serious security concerns, the use of `telnet` has been replaced by ssh (Secure Shell Server) these days.
- To install telnet daemon on our ubuntu server, you need to use following commands:
  `# apt-get install telnetd, xinetd`
  - To check the status of a service you can use the `systemctl` command on your ubuntu server machine:
    `# systemctl status/start/stop xinetd`
  - Once you have started the telnet service, you can connect to it from the kali machine:
    `$ telnet <ip of ubuntuserver>`

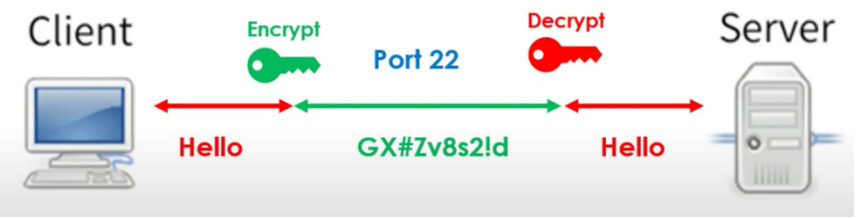
```
┌──(kali㉿kali)-[~]
└─$ telnet 192.168.8.111 23
Trying 192.168.8.111 ...
Connected to 192.168.8.111.
Escape character is '^]'.
Ubuntu 16.04.3 LTS
ubuntuserver login: arif
Password:
Last login: Mon Sep  2 10:59:45 PKT 2024 from 192.168.8.112 on pts/1
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

96 packages can be updated.
49 of these updates are security updates.
To see these additional updates run: apt list --upgradable

New release '18.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


You have new mail.
arif@ubuntuserver:~$
```

  - Remember you need to give the credentials of a user who exist on the ubuntu server machine. The credentials are compared with the username:password stored in `/etc/passwd` and `/etc/shadow files` and if matched a login session is created. ☺
  - From windows machine you can use the putty to connect to ubuntu server ☺

To Do:

- Students can verify that telnet is insecure, by running `wireshark` on Kali Linux and then connect to ubuntu server (Wireshark is a packet sniffer that captures NW packets in real time and display them in human readable format ☺
- Students should also use netstat (Network Statistics) utility to check out the open connections on the two machines with different status of the related ports (Listen, Established, Closed, Time_wait).

The SSH Service



- SSH (Secure Shell) is a network protocol, and command-line utility used to securely connect to remote systems over an unsecured network. It provides encrypted communication for logging into remote machines and executing commands, making it a fundamental tool for system administrators, developers, and anyone needing secure remote access. It can be used on all UNIXs, Windows, Linux, rather on any device running Linux like Mac, Android, iPhone and almost all sort of routers.

- On ubuntu server machine you need to install openssh server
  # apt-get install openssh-server

- On kali machine (client) you need to install openssh-client
  $ sudo apt-get install openssh-client

- In order to login to Ubuntu Server as a user arif who must exist on Ubuntu Server, use the following command:
  $ ssh arif@<ip of US>

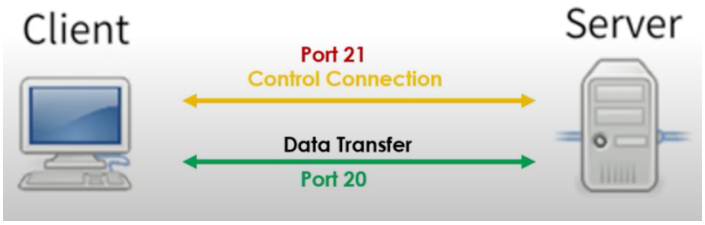
```
┌──(kali㉿kali)-[~]
└─$ ssh arif@192.168.8.111
arif@192.168.8.111's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

96 packages can be updated.
49 of these updates are security updates.
To see these additional updates run: apt list --upgradable

New release '18.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


You have new mail.
Last login: Mon Sep  2 11:07:27 2024 from 192.168.8.112
arif@ubuntuserver:~$
```

- If you want to login to your Metasploitable2 machine from Kali Linux using `ssh`, the simple login command might not work. There is a work around in which you need to mention the `HostKeyAlogirthms` as shown in the following command: ☺
  $ ssh -oHostKeyAlgorithms=+ssh-dss msfadmin@<ip of M2>

- In order to login to execute a command on Ubuntu Server without logging into an interactive session:
  $ ssh arif@<ip of US> 'ls -l'

- In order to securely copy file/directory to and from a remote host, you can use the scp command. The following command will copy a file from Kali Linux to Ubuntu Server:
  $ scp myfile.txt arif@<ip of US>:/home/arif/file1.txt

The `vsftpd` Service



- File Transfer Protocol (FTP) is used to upload and download files between two computers in a Local Area Network or on the Internet. Like SSH, FTP is also a client/server model. The server component called the FTP daemon or service, listens for FTP requests from remote clients. When a request is received it:
  - Authenticate the user,
  - Establish the control channel,
  - Establish the data channel and for the duration of the session it executes any of the valid commands sent by FTP client, and finally
  - Disconnect the connection.
- There exist various **`ftp client`** programs, some of which are GUI based (Filezilla) and some are CLI based (`ftp`). On our Kali Linux machine, we will install `ftp`:
  `$ sudo apt-get install ftp`
- Similarly, there exist various **`ftp server`** programs, some of which are GUI based (Filezilla-Server) and some are CLI based (`vsftpd`). On our Ubuntu server machine, we will install `vsftpd`:
  `# apt-get install vsftpd`

  Notes:
  - After installing students are advised to read the man pages of `vsftpd(8)` as well as the man pages of its configuration file `vsftpd.conf(5)`
  - In order to **configure vsftpd server**, you need to open its self-explanatory and very well commented configuration file (`/etc/vsftpd.conf`) in some editor and make necessary changes in the values of various directives. For example, `listen=YES` means that `vsftpd` will work in stand-alone mode and listens on port 21, while `listen=NO` means that `vsftpd` will work under supervision of `xinetd`, in which case you have to create its configuration file `/etc/xinetd.d/vsftpd`
  - Finally, you need to run the `vsftpd` service using the `systemctl` command on Ubuntu Server and to verify if the service is running and listening at port 21, you can run this `# netstat -ant | grep 21` command on Ubuntu Server ☺
- Now with everything set, let us connect to the `vsftpd` server running on Ubuntu Server machine from our Kali Linux machine:
  `$ ftp <ip of US>`

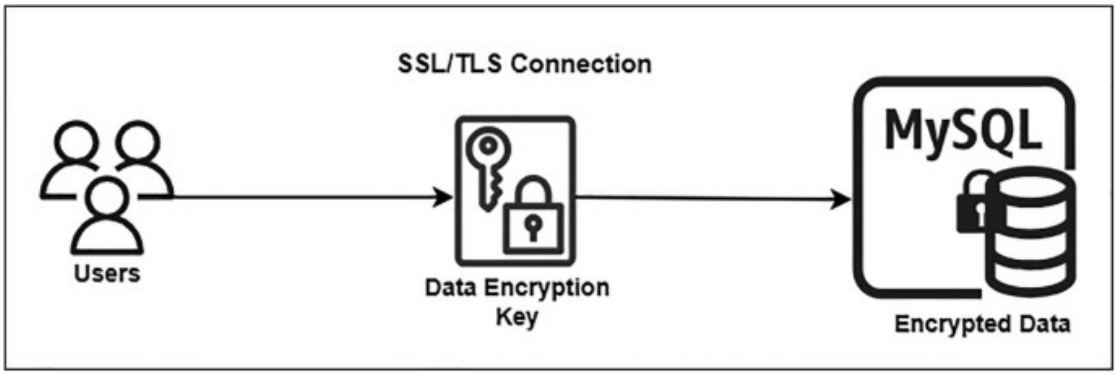
```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.8.111
Connected to 192.168.8.111.
220 Welcome to Arif FTP service configured for Learning purpose. User name is 'anonymous' and Password is blank
Name (192.168.8.111:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
Remote directory: /
ftp> !pwd
/home/kali
ftp> ls
229 Entering Extended Passive Mode (|||23717|)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Nov 16  2017 d1
-rw-r--r--    1 0        0               0 Nov 16  2017 f1.txt
226 Directory send OK.
ftp> exit
221 Goodbye.

┌──(kali㉿kali)-[~]
└─$
```

To Do:

The anonymous user is jailed inside /`var/ftp/` directory inside US machine, to use different commands inside this interactive session, type `ftp> help`. Students should also try to login as a local user instead of anonymous user by making changes in the configuration file and see how can you secure your ftp server by jailing your ftp users in their home directories ☺

The `mysql` Service

- MySQL Server is an open-source relational database management system (RDBMS) that stores and manages data using SQL (Structured Query Language). It supports multi-user access to databases and is commonly used for web applications and data storage.



Installing MySQL (mariadb-server): Both MySQL (owned by Oracle) and MariaDB (community driven fork of MySQL) are open-source RDBMS, but they differ in terms of licensing, features, and performance. Both stores and manages data using SQL (Structured Query Language) and supports multi-user access to databases. For the practice sake, you can install mariadb-server on Ubuntu server machine, while can install the client on Kali machine:

```
# apt-get install mariadb-server

# mysql --version
mysql from 11.4.3-MariaDB, client 15.2 for Debian-linux-gnu

# mariadb --version
mariadb from 11.4.3-MariaDB, client 15.2 for Debian-linux-gnu

# systemctl start mariadb.service

# systemctl enable mariadb.service

# netstat -ant | grep 3306

tcp      0    0    127.0.0.1:3306      0.0.0.0   LISTEN
```

The configuration file of Mariadb is `/etc/mysql/mariadb.conf.d/50-server.cnf`. By default, MySQL only listens on local host for security reasons, as shown in the above output. If you want to allow remote connections, open its configuration file with root privileges, and comment the line `bind-address = 127.0.0.1` by putting a hash symbol before it. After this. You need to restart the service.

```
# systemctl restart mariadb.service
# netstat -ant | grep 3306
tcp      0    0    0.0.0.0:3306        0.0.0.0:*       LISTEN
tcp6     0    0    :::3306             :::*            LISTEN
```

You can access using graphic tools like **phpMyAdmin** or **MySQL Workbench**, however, I will be using command line client.

- Accessing Locally:
  ```
  $ mysql -u [username] -p
  ```
- Accessing Remotely:
  ```
  $ mysql -h [hostname or IP] -u [username] -p
  ```

The root user is the default super user created during MySQL or Mariadb installation. For the first time when you login as root you use the following command to set the password of root user:

```
        # mariadb -u root
MariaDB [(none)]> alter user 'root'@'127.0.0.1' identified by 'pucit';
MariaDB [(none)]> flush privileges;
MariaDB [(none)]> exit
```

Practicing SQL: Please practice by creating a sample database at your own time. ☺

```
CREATE DATABASE db1;
SHOW DATABASES;
USE db1;
DROP DATABASE db1;
```

1) **DDL:** Data Definition Language commands for defining a database schema. They are used for creating, modifying, dropping and renaming the structure of database objects.

```
a) CREATE TABLE users (first_name VARCHAR(20), last_name VARCHAR(20));
b) ALTER TABLE users ADD COLUMN address varchar(20);
c) DROP TABLE users;
d) TRUNCATE TABLE users;
e) RENAME TABLE users TO members;
```

2) **DML:** Data Manipulation Language deals with retrieving, storing, modifying and deleting data.

```
a) SELECT * FROM users WHERE first_name='Arif';
b) INSERT INTO users (first_name, last_name) VALUES ('Rauf', 'Butt');
c) UPDATE users SET last_name='Khan' WHERE first_name='Arif';
d) DELETE FROM users WHERE first_name='Arif' AND last_name='Butt';
```

3) **DCL:** Data Control Language is used to implement access control logic on database objects.

```
a) GRANT ALL PRIVILEGES on users TO 'arif'@'localhost';
b) REVOKE ALL PRIVILEGES on users FROM 'arif'@'localhost';
c) SHOW GRANTS FOR 'arif'@'localhost';
```

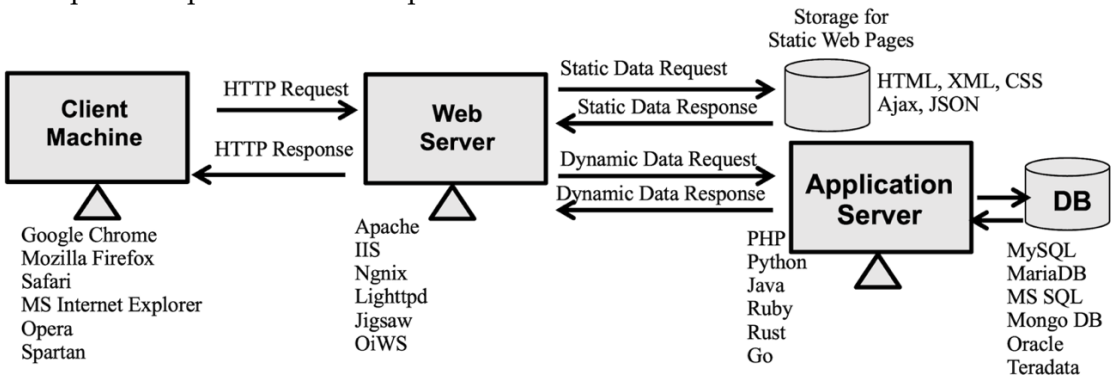
4) **TCL:** Transaction Control Language is used to manage transactions in a database. You can commit to save the changes and rollback to undo the changes as shown below. You can create savepoint and can rollback to a specific savepoint.

```
   START TRANSACTION;
   -  Perform DML operations -
COMMIT/ROLLBACK
```

Configuring and Using httpd Service under Apache Web Server

How does HTTP Request and Response Work?

- We all know that the World Wide Web is a repository of web pages distributed all over the world on different devices or computers. The http/https protocol is a vehicle used to move web pages over the Internet. Before getting into details of configuring **`httpd`** service and accessing a website using Apache Web Server, let us quickly have a recap of what HTTP protocol is, its methods, header, request and response. We will be talking about these concepts a lot, so it is better to get used to these concepts.
    - **Hyper Text Transfer Protocol (HTTP/HTTPS)** is the part of TCP/IP protocol family that works on the Application layer at port 80/443 and consists of a set of rules for transferring data between computers and servers on the Internet. It consists of text only, however, it transfers all kinds of file formats, such as video, audio, and text.
    - It is a **client/server architecture** where a web browser, robot, search engine, and so on act as HTTP clients, and the web server acts as a server. The session is a sequence of request-response network operations.



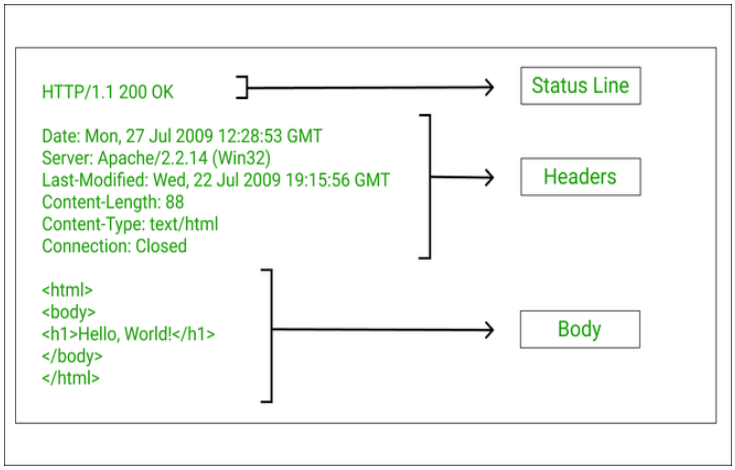
- Now let us understand the steps involved in the working of HTTP protocol at an abstract level

1. **User Input:** The user enters a URL into the browser's address bar or clicks on a link, initiating an HTTP request.
   `protocol://hostname.domain-name[:port]/pathtoresource`
   `http://pucit.pu.edu.pk:80/academics/timetable-pucit.html`
2. **HTTP Request Generation:** The browser generates an HTTP request based on the entered URL. The structure of an HTTP request is shown below:
    - *Status Line:* Specifies the request method, the requested resource URL and protocol version.
    - *Headers:* Provide additional info about the request `(key:value)`, such as the client's user agent, accepted content types, and cookies.
    - *Body:* Contains optional data sent by the client, such as form data or file uploads (for POST request). Remember for GET the data is sent as part of the URL as `key1=value1&key2=value2` pairs.

3. **Transmission to Server:** The request is sent from the client's device to the Internet Service Provider (ISP) and then over the Internet to the appropriate server.

4. **Server Processing:** Upon receiving the request, the server interprets it as an HTTP request and identifies the requested resource based on the URL.
5. **HTTP Response Generation:** The server generates an HTTP response, which includes a status code indicating the success or failure of the request, along with the requested resource (if applicable).

- *Status Line:* Includes the HTTP version, a status code of outcome of request like 200 (OK), 404 (Not found), 500 (internal server error), and a textual description of the status code.
- *Headers:* Provide information about the response (`key:value`), such as date, server type, content type, content length etc.
- *Body:* Contains the actual response data that the client requested, e.g., an HTML file, JSON data, an image, or may be an error message.



6. **Transmission to Client:** The server sends the HTTP response back to the client through the ISP and over the Internet.
7. **Browser Interpretation:** The browser receives the HTTP response and interprets it based on the status code and content type specified in the response headers.
8. **Data Display:** Finally, the browser displays the retrieved data (e.g., web page, image, video) to the user based on the information received in the HTTP response.

Proof of above Concepts using curl: The Client URL, famously known as **curl** is a command-line tool and library for transferring data to and from a server. It's support for wide variety of protocols and options like http(s), ftp(s), sftp, and so on make it a versatile tool for interacting with web services, APIs, and other network resources.

- **Installation**

```
$ sudo apt install curl
```

- **Usage**
  - **Fetch the Contents of a URL**:

```
$ curl -v http://example.com
```

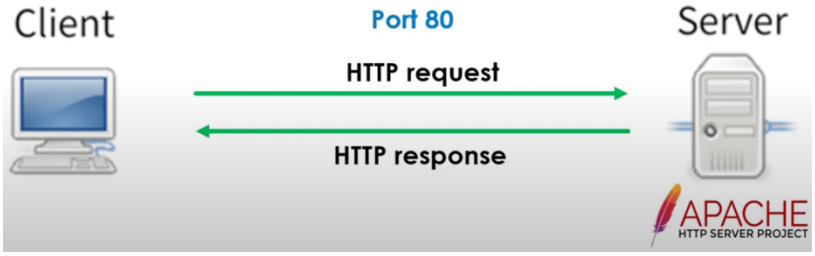
  - **Download a File**: The `-O` option saves the file with its original name in pwd, while `-o` is used to specify a custom name and path to the file.

```
$ curl -O http://example.com/file.zip
$ curl -o ~/d1/myfile.zip http://example.com/file.zip
```

  - **Resume a Download**: -C - resumes a download from where it left off.

```
$ curl -C - -O http://example.com/file.zip
```

Hands-On Practice of Hosting a Website using Apache Web Server



- **Web Hosting:**
  - Suppose we have created a website or may be a single web page and we want the entire world to access it using their favourite browser, from their homes or work places. For this to work we need to host all the web pages and other necessary files of our website on a machine having a public IP address and running a web server on it.
  - There exist free as well as paid hosting services that are available to us, but in both the cases, while selecting a hosting server, we must decide:
    - Whether we need a Linux or a Windows based server
    - Technologies our website requires like Java, PHP, Python, Perl
    - Requirement of any DBMS like MySQL or Oracle
    - Processing and connection speed we need
    - Disk space required
    - Email accounts
    - Backup facilities and so on.
  - After we have finalized our web hosting server, we just need to copy all the required text, image, audio and video files that comprise our website on the web server using let's say ftp service.
  - In this part of our handout, we are going to host our web pages on our Ubuntu Server machine which will be running the Apache web server and will of course access our website using a browser from our Kali Linux machine.
- **Installing and Starting Apache on Ubuntu Server:**
  - The Apache HTTP Server is the most widely used web server software and runs on 67% of all web sites in the world. It is open source and is available for all UNICES, Mac, Linux and Microsoft platforms. The name "Apache" has been taken from a Native American tribe, that is famous for its skills in warfare and strategy making. To install Apache Web Server on Ubuntu Server machine, give the following command:

```
# apt-get install apache2
# apt-get install apache2-doc
# apache2 -v
```

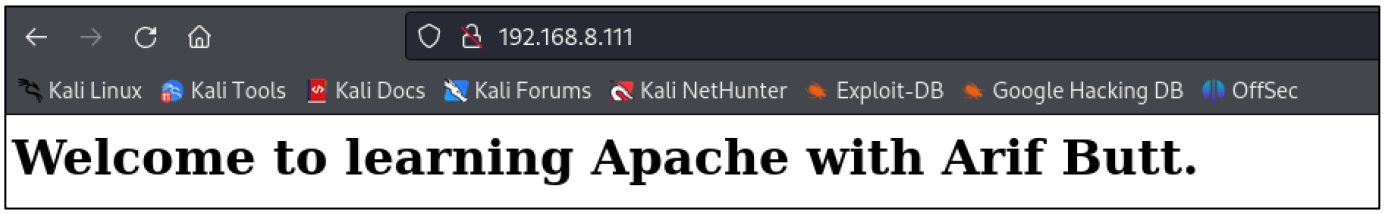
  - Do read the man pages of apache2, start the service and check the status of port 80 on Ubuntu server machine.

```
# man apache2
# systemctl start apache2.service
# systemctl enable apache2.service
# systemctl status apache2.service
# netstat -antp | grep 80
```

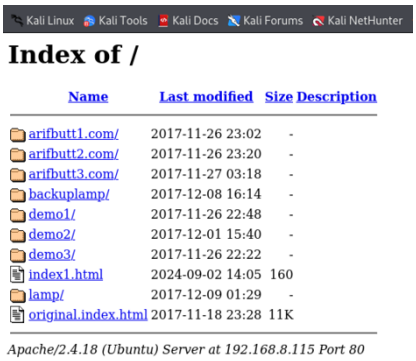
```
root@ubuntuserver:~# netstat -antp | grep 80
tcp        0      0 0.0.0.0:48053           0.0.0.0:*               LISTEN      1171/rpc.mountd
tcp6       0      0 :::80                   :::*                    LISTEN      29252/apache2
root@ubuntuserver:~#
```

- In above screenshot, note that on Ubuntu Server machine, there is one port listening on port 80 for incoming connections from any machine and from any port.

- **Accessing a Static Web Page from Kali Linux:**
  - Now let us move on to our Kali Linux machine, open a browser and type the following address: http://<ip of US>:80
  - On Ubuntu Server if the *Document Root directory*, (`/var/www/html/`) contains a simple HTML file named `index.html`, its contents will be sent to the browser on Kali, which will render the html page and display it as shown below:



  - Do access this page using `$ curl -v <ip of US>`
  - On Ubuntu Server if the *Document Root directory*, (`/var/www/html/`) do not contain a file named `index.html`, then the directory listing of that directory will be sent to the browser on Kali as shown:



  - Note: Students should try to make changes in the configuration file `/etc/apache2/apache2.conf` to ensure that the directory listing of a directory should not be sent to the client even if there is no `index.html` file in that directory.
  - After the connection is established, let us check the status of ports on Ubuntu Server machine using the netstat command. Do note that now there exist two ports, one is still in listen state, while the other is in the established state with the Kali Linux machine ☺

```
root@ubuntuserver:~# netstat -antp | grep 80
tcp        0      0 0.0.0.0:48053           0.0.0.0:*               LISTEN      1171/rpc.mountd
tcp6       0      0 :::80                   :::*                    LISTEN      29252/apache2
tcp6       0      0 192.168.8.111:80        192.168.8.112:51048     ESTABLISHED 29255/apache2
root@ubuntuserver:~#
```

- **Dynamic Web Pages:**
  - This was a demo of a static web page, let us now change gear and see a demo of a dynamic web page and let us use PHP for creating our dynamic web page. PHP is an opensource server-side scripting language that runs. On all UNICES, Linux, Mac and Microsoft platforms and. Is compatible with most web servers like Apache, IIS, and so on. It can do lot of tasks for us like:
    - Generating dynamic page contents
    - Can collect form data
    - Can create, open, read, write, delete and close files on server
    - Can connect to a variety of databases and add, modify, delete records
    - Can send and receive cookies
    - Can encrypt data
  - Let us install php on our Ubuntu Server machine:

 # apt-get install php libapache2-mod-php php-mysql php-mcrypt

- Now that we have installed php, let us write a very basic dynamic web page that displays the current date and time whenever requested.

 # cat /var/www/html/demo1/index.php

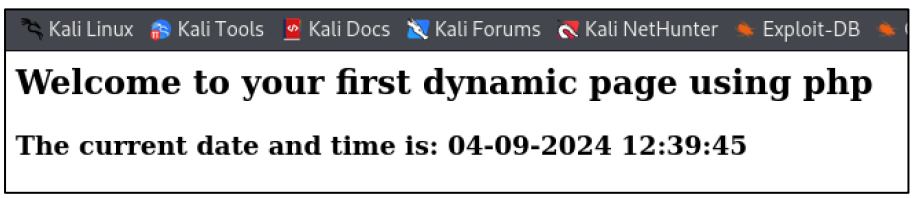
```
root@ubuntuserver:/var/www/html/demo1# cat index.php
<html>
   <head>
      <title>
         Demo2
      </title>
   </head>
   <body>
      <h2> Welcome to your first dynamic page using php</h2>
        <?php
           $date = date('d-m-Y H:i:s');
            echo "<h3> The current date and time is: $date </h3>";
        ?>
   </body>
</html>
root@ubuntuserver:/var/www/html/demo1#
```

Code Description:

- A php file can contain HTML, CSS, JS, and PHP code.
- In PHP, you don't need to declare a variable, you just create it and assign it a value. A variable name must start with a $ sign and is case sensitive
- There are tons of functions available and `date()` is one of them which returns the current date and time in the specified format
- One can use `print()` or `echo()` functions to display strings. The `echo()` function can be used with or without parenthesis

Accessing a Dynamic Web Page from Kali Linux:

- On Ubuntu Server `/var/www/html/demo1` directory contains a above file named `index.html`. Let us now open a browser on Kali Linux machine and give the following address to access this dynamic web page: http://www<ip of US>:80/demo1



- Each time we request the page, the PHP script is executed on the server, and the plain HTML result is sent back to the browser.
- Do access this page using `$ curl -v <ip of US>`

Session Management

We all know that HTTP(S) protocol is stateless, however, a good web application needs to make it stateful. For example, once a user has been authenticated by the server, the user may like to visit other pages of the application, e.g., his/her profile page, or change password page and so on. Now all these pages are authenticated pages, so one way is that the user has to give his/her `username:password` every time he/she wants to visit an authenticated page and the other way is using Session ID (cookies). ***Session Management*** *is a process that involves maintaining state between a user and a web application across multiple requests, as HTTP itself is a stateless protocol.* Here is a breakdown of how session management is done at an abstract level:

- **Session Initialization:** A user is authenticated by a login form, and upon successful authentication the server creates a unique session ID, which is sent back to the client and is stored in cookies (a small piece of alphanumeric data sent by the server and stored on the client's browser), URL parameters or HTML hidden fields. The HTTP response sent by the above php code is shown below:

```php
<?php
  setcookie('cookieA', 'aaaaaa');
  setcookie('cookieB', 'bbbbbb', time() + 3600);

  echo "<h2>Cookies are set</h2>"
?>
```

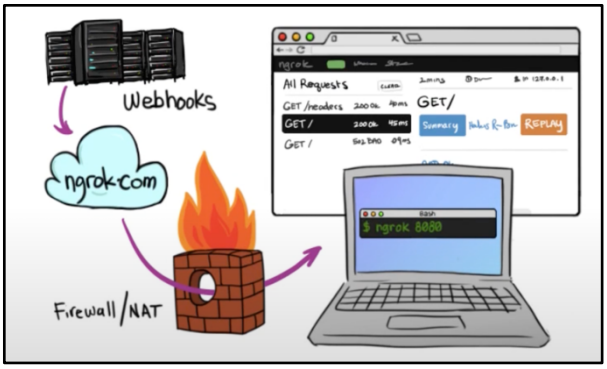
```
GET: HTTP/1.1 200 OK
Date: Wed, 25 Aug 2021 20:40:15 GMT
Server: Apache/2.4.41 (Ubuntu)
Set-Cookie: cookieA=aaaaaa
cookieB=bbbbbb; expires=Wed, 25-Aug-2021 21:40:15 GMT; Max-Age=3600
Content-Length: 28
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

- **Session Continuation:** The next time the user sends a HTTP request to the server, it keeps the cookie value in the header portion of the HTTP request to make the server know that it is a continuation of previous requests. The server uses this cookie value to retrieve the user's session data from its memory or a database.
- **Session Termination:** The session ends when either the user logs out, or due to inactivity by the user. Two related types of cookies in this context are:
    - Non-persistent cookies, also called session cookies, are stored temporarily in the browser's memory and are deleted when the browser is closed. An example of non-persistent cookie is **cookieA** in the example above.
    - Persistent Cookies are stored on the user's device even after the browser is closed. They remain valid until their expiration date or until the user manually deletes them. An example of persistent cookie is **cookieB** in the example above.

Note:

- Be watchful, while working inside a public WiFi network as someone might be able to steal your sessionID and can impersonate you to the webserver. More on this later ☺
- In your web engineering course, you might have studied the three ways to communicate with a web server, which are normal http requests, Asynchronous JavaScript and XML (AJAX) and websockets. Please revise these concepts these as you will be needing them in the later part of the course ☺

Expose your localhost using **ngrok**



It is a tool that exposes local networked services behind NATs and Firewalls to the public Internet over a secure tunnel. Essentially, it enables you to expose a local web service (or any other type of service running on your machine) to the outside world by providing a public URL. This is especially useful for testing and development when you need to share your local environment with others or access it remotely.

Installing ngrok:

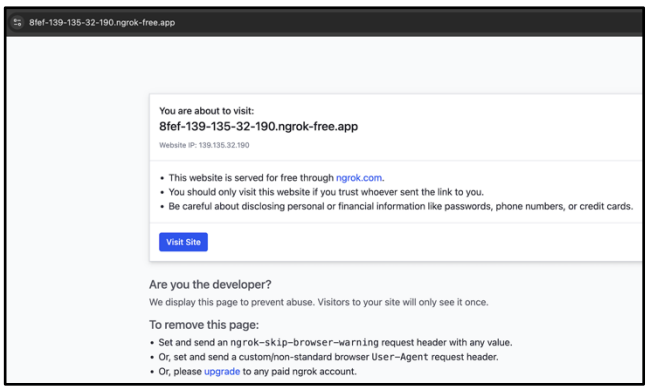
- Download and install ngrok: https://ngrok.com/downloads
- Sign-up and get a token: https://dashboard.ngrok.com/signup
- Add the authtoken to `ngrok` client using this command: `ngrok config add-authtoken`

Exposing you Web Server running on Kali Linux to Internet:

- Web server (port 8080): `ngrok http 8080`
- SSH server (port 22): `ngrok tcp 22`
- Postgres server (port 5432): `ngrok tcp 5432`

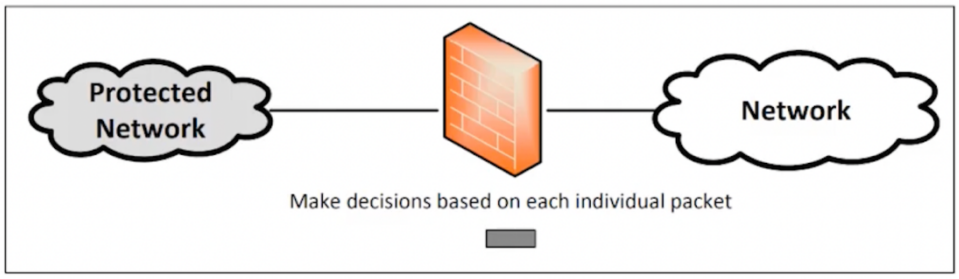
```
ngrok

👋 Goodbye tunnels, hello Agent Endpoints: https://ngrok.com/r/aep

Session Status                online
Account                       Arif Butt (Plan: Free)
Update                        update available (version 3.22.0, Ctrl-U to update)
Version                       3.20.0
Region                        Asia Pacific (ap)
Latency                       108ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://8fef-139-135-32-190.ngrok-free.app -> http://localhost:80

Connections                   ttl     opn     rt1     rt5     p50     p90
                              5       0       0.07    0.02    0.27    5.42

HTTP Requests
-------------

19:10:17.167 PKT GET /icons/unknown.gif         200 OK
19:10:17.067 PKT GET /icons/blank.gif           200 OK
19:10:17.070 PKT GET /icons/text.gif            200 OK
19:10:17.453 PKT GET /favicon.ico               404 Not Found
19:10:17.171 PKT GET /icons/binary.gif          200 OK
19:10:17.171 PKT GET /icons/folder.gif          200 OK
19:10:16.657 PKT GET /                          200 OK
```

Accessing the Web Server running on Kali Linux:
Open your browser

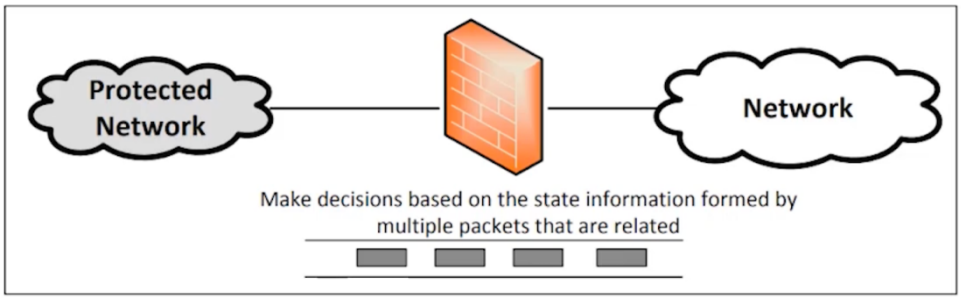


Firewalls

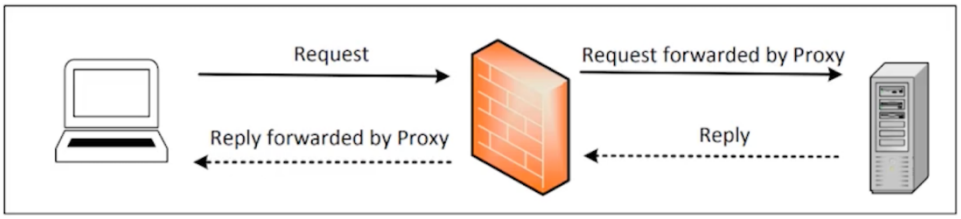
A **firewall** is a security system that monitors and controls incoming (Ingress) and outgoing (Egress) network traffic based on predetermined security rules. It acts as a barrier between a trusted internal network and untrusted external networks, such as the Internet, to block malicious traffic while allowing legitimate communication. A firewall operates by inspecting packets of data moving through a network and deciding whether to allow or block them based on predefined rules. These rules can filter traffic based on IP addresses, port numbers & protocols, or specific applications behaviour. Based on the contents a firewall examines, it can be categorized as Packet Filter Firewall, Stateful Firewall, and Application Firewall.



(A) Packet Filter Firewall



(B) Stateful Firewall



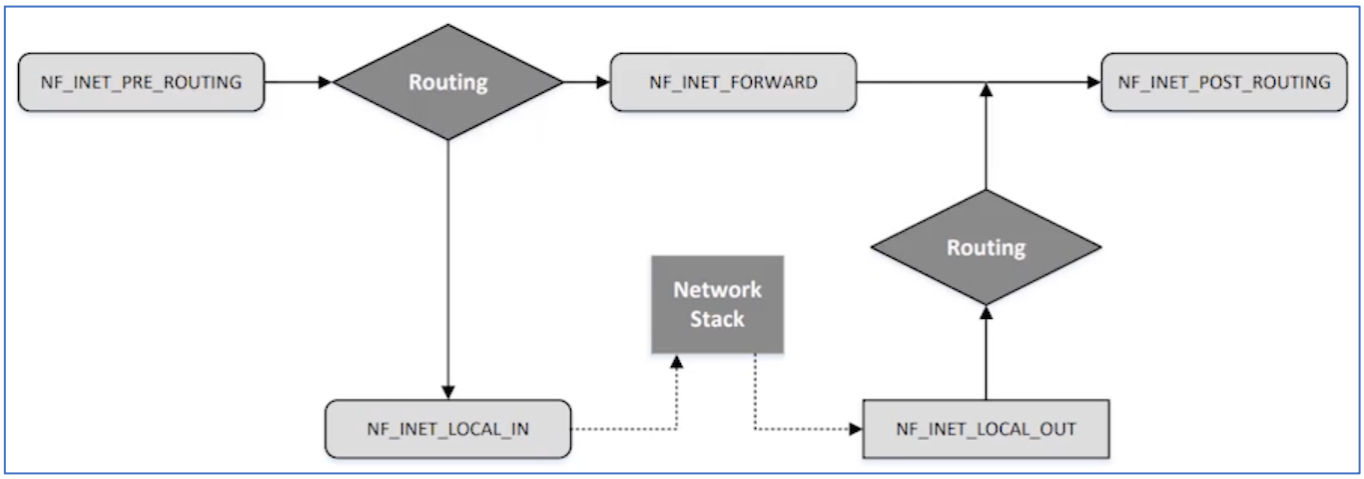
(C) Application/Proxy Firewall

At an abstract level, here are the three categories of firewalls based on their deployment:

- Hardware Firewalls: A separate dedicated physical device placed at NW's edge (gateway) that filter network traffic (e.g., Cisco ASA, Palo Alto Networks, Fortigate).
- Software Firewalls: Installed on operating systems to filter traffic (e.g., on Linux iptables/nftables/ufw are firewalls, Windows Defender is a AV and a firewall as well).
- Cloud Firewalls: Firewalls provided as a cloud-based service (e.g., AWS Network Firewall, Azure Firewall, Google Cloud Armour, Cloudflare for webapps).

Software Firewalls for Linux: `iptables`, `nftables`, and `ufw`

- The **`iptables`** is a command-line firewall utility introduced in 2000 in Linux kernel 2.4, while **`nftables`** is its replacement introduced in 2014 in Linux kernel 3.13. **`ufw`** (Uncomplicated Firewall) is a user-friendly firewall management tool and is the default firewall on Ubuntu and other Debian-based distributions. At the backend all use **`netfilter`**, which is a packet filtering framework in the Linux kernel. The key components of `netfilter` framework are hooks, tables, chains and rules. The following figure illustrates how packets traverse the five ***netfilter hooks*** at different stages packet processing in the Linux kernel:
  - **NF_INET_PRE_ROUTING** → Before routing decision
  - **NF_INET_LOCAL_IN** → Packet is destined for local system
  - **NF_INET_LOCAL_OUT** → Packet is generated by the local system
  - **NF_INET_FORWARD** → Packet is forwarded through the system
  - **NF_INET_POST_ROUTING** → Packet is about to leave the system



In the next two pages, I have described use of the Linux **`nft`** utility to create a basic firewall on our Ubuntu Server machine, however, interested students having good C programming background can explore the usage of `netfilter` framework by writing their own Loadable Kernel Modules (LKM). **LKM** in Linux is a piece of code that can be dynamically loaded into or unloaded from the kernel at runtime without requiring a system reboot to extend the functionality of the kernel without modifying its core. In MS Windows we have Kernel-Mode Drivers (KMDs).

To Do:

Use Loadable Kernel Module (LKM) to allow dynamic manipulation of network/security policies enforced at the kernel level, e.g., circumventing SELinux constraints. For this, you need to write down your own hook and unhook functions and register them using the `module_init()` and the `module_exit()` functions respectively. (In the definition of the hook function, you need to mention the hook location, which can be one of the five locations illustrated in the above figure). Once done, compile the C-source file using the Linux `make` utility to generate the `.ko` file. Now use the `insmod` command to insert the `.ko` file (LKM) into the running kernel. You can use the `lsmod` command to list the loaded modules and `dmesg` command to look at the kernel buffer. Once done testing, do not forget to use the `rmmod` command to remove the specified module from the kernel. Happy Learning ☺

Hands On Practice: nftables

Installation

- Let us install the iptables, nftables and ufw on our Ubuntu Server machine:

```
# apt update && sudo apt install iptables nftables ufw
```

- Let us check out the versions:

```
# iptables --version
iptables v1.6.0
# nft --version
nftables v0.5
# ufw --version
ufw 0.35
```

- Before we start working with nftables, just make sure that iptables rules should not exist on Ubuntu Server machine. If they do, we need to flush them to avoid any conflicts:

```
# iptables -L      (List all firewall rules in all chains)
# iptables -F      (Delete all rules from all chains, but keep the chains intact)
# iptables -X      (Delete all user defined chains)
# iptables -Z      (Reset packet and byte counters to zero)
```

- Unlike iptables, nftables is a service that we need to start using systemctl command:

```
# systemctl start nftables
```

- It is recommended to start with nftables with a clean slate, so use the following command to flush all the existing rules:

```
# nft flush ruleset
# nft list tables|chains|ruleset
```

Creating a firewall

- **Create a new table:** A table in a firewall is a container that holds chains and rules. A firewall can have more than one tables. The address family must be one of ip, ip6, inet, arp, bridge, or netdev. When no address family is specified, ip is used by default. The syntax/example to create a new table, is shown below:

```
nft {add|delete|list|flush} table {ip|ip6|inet} {tbl-name}
# nft add table inet fw1
```

- **Add a chain in the table:** Chains are containers for rules. A chain inside a table defines how and when rules are applied to network traffic. The syntax to create a new chain inside a table is shown below. The example creates a new chain inside the fw1 table with the name of input:

```
nft {add|delete|rename|list|flush} chain {ip|ip6|inet} {table} {chain}
# nft add chain inet fw1 input   {type filter hook input priority 0 \; policy accept \;}
# nft add chain inet fw1 output  {type filter hook output priority 0 \; policy accept \;}
# nft add chain inet fw1 forward {type filter hook forward priority 0 \; policy drop \;}
```

  - **type** filter is used for packet filtering or firewalling. Other options are nat for Network Address Translation, mangle for modifying packet headers, raw for connection tracking exemptions, and security for mandatory access control in SELinux.
  - **hook** input attaches this chain to the input hook, i.e., it will process the incoming packets destined to the local machine (other options are output, forward, prerouting, postrouting). The hook actually determines when this chain processes packets.
  - **priority** 0 determines the priority of this chain, when multiple chains exist. Lower/negative numbers runs earlier, higher number runs later.
  - The ; is used by bash as command separator. Each statement inside {} must be separated by semicolon. So, to prevent bash from interpreting it prematurely, we use back slash as escape sequence.

- **Syntax to add a rule in the chain:** We add rules inside a chain of a table that defines how traffic is handled. There can be multiple rules inside a chain and they are checked from top to bottom. Do not forget to place more specific rules at the top and general rules at the bottom. Following is the general syntax of adding a rule:
  `nft {add|insert|delete} rule [addr-family] {table} {chain} {statement}`

  - Let us first add two rules that allow communication on the loopback interface, and allow all incoming traffic that is part of already established connections:
    ```
    # nft add rule inet fw1 input iif lo accept
    # nft add rule inet fw1 input ct state established,related accept
    ```
  - **Block ICMP (ping) incoming requests but allow everything else:**
    ```
    # nft add rule inet fw1 input ip protocol icmp drop
    ```
  - **Rule to drop telnet traffic on port 23:**
    ```
    # nft add rule inet fw1 input tcp dport 23 drop
    ```
  - **Rule to allow http and https traffic:**
    ```
    # nft add rule inet fw1 input tcp dport {80, 443} accept
    ```
  - **Rule to drop a specific IP address:**
    ```
    # nft add rule inet fw1 input ip saddr <IP> drop
    ```
  - **Rule to drop all input traffic:**
    ```
    # nft add rule inet fw1 input drop
    ```

- **Display entire firewall ruleset:**
  ```
  # nft list ruleset
  table inet fw1 {
      chain input {
          type filter hook input priority 0; policy accept;
          iif lo accept
          ct state established,related accept
          ip protocol icmp drop
          tcp dport telnet drop
          tcp dport {http, https} accept
          ip saddr 10.0.2.55 drop
          drop
      }
      chain output {
          type filter hook output priority 0; policy accept;
      }
      chain forward {
          type filter hook forward priority 0; policy drop;
      }
  }
  ```
- All the tables, their chains and the rules in the chains are there in memory. If you flush them or reboot, they all will be lost and you need to redefine them. So, to make these rules permanent write them in the following file.
  `# nft list ruleset > /etc/nftables.conf`
- This is just a hello world about firewalls. Expand your learning at your own… ☺

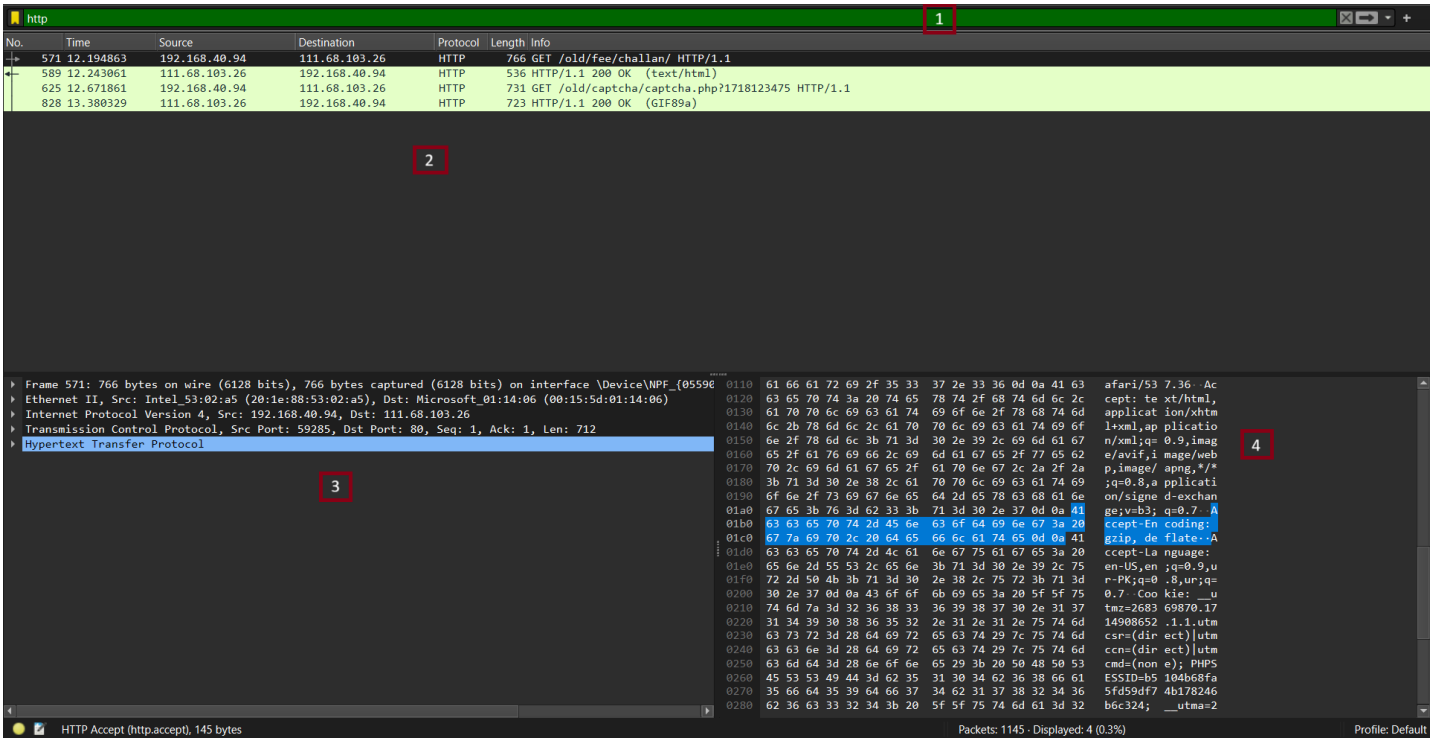
Wireshark

Wireshark https://www.wireshark.org/ is a free and open-source packet analyzer. It's used for network troubleshooting, analysis, software and communications protocol development. In simple terms, Wireshark captures data packets on a network and displays them in detail for you to analyze.

Installation

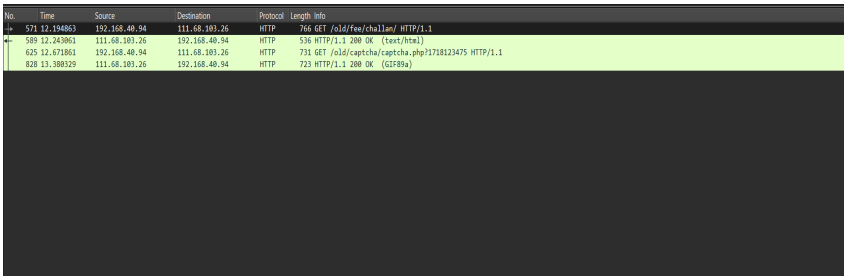
```
$ sudo add-apt-repository ppa:wireshark-dev/stable
$ sudo apt update
$ sudo apt install wireshark
```

Wireshark Interface

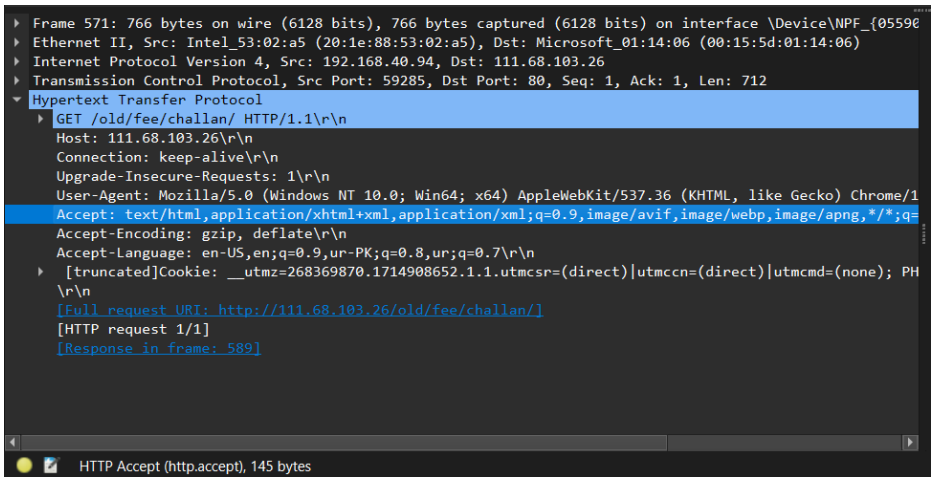


1. **Filter Toolbar:** The Filter Field is used to enter filters to narrow down the displayed packets. For example, typing "http" will only show HTTP packets. ip.addr == 192.168.1.1 will show the traffic from specified IP address only.

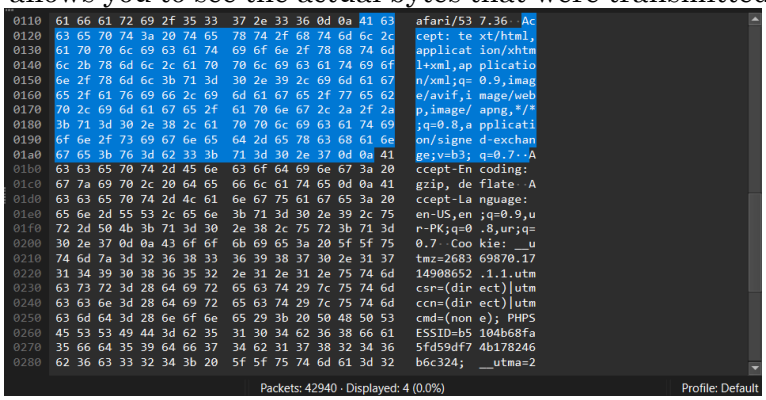
2. **Packet List Pane:** This pane provides a one-line summary of each captured packet. The columns typically include:
   - **No.:** Packet number in the capture file.
   - **Time:** Time when the packet was captured.
   - **Source:** Source IP address.
   - **Destination:** Destination IP address.
   - **Protocol:** Protocol used (e.g., TCP, UDP, HTTP).
   - **Length:** Length of the packet.
   - **Info:** Brief information about the packet.



3. **Packet Details Pane:** When a packet is selected in the Packet List Pane, detailed information about that packet is displayed here. It is divided into expandable sections for each layer of the network stack (e.g., Frame, Ethernet, IP, TCP/UDP).

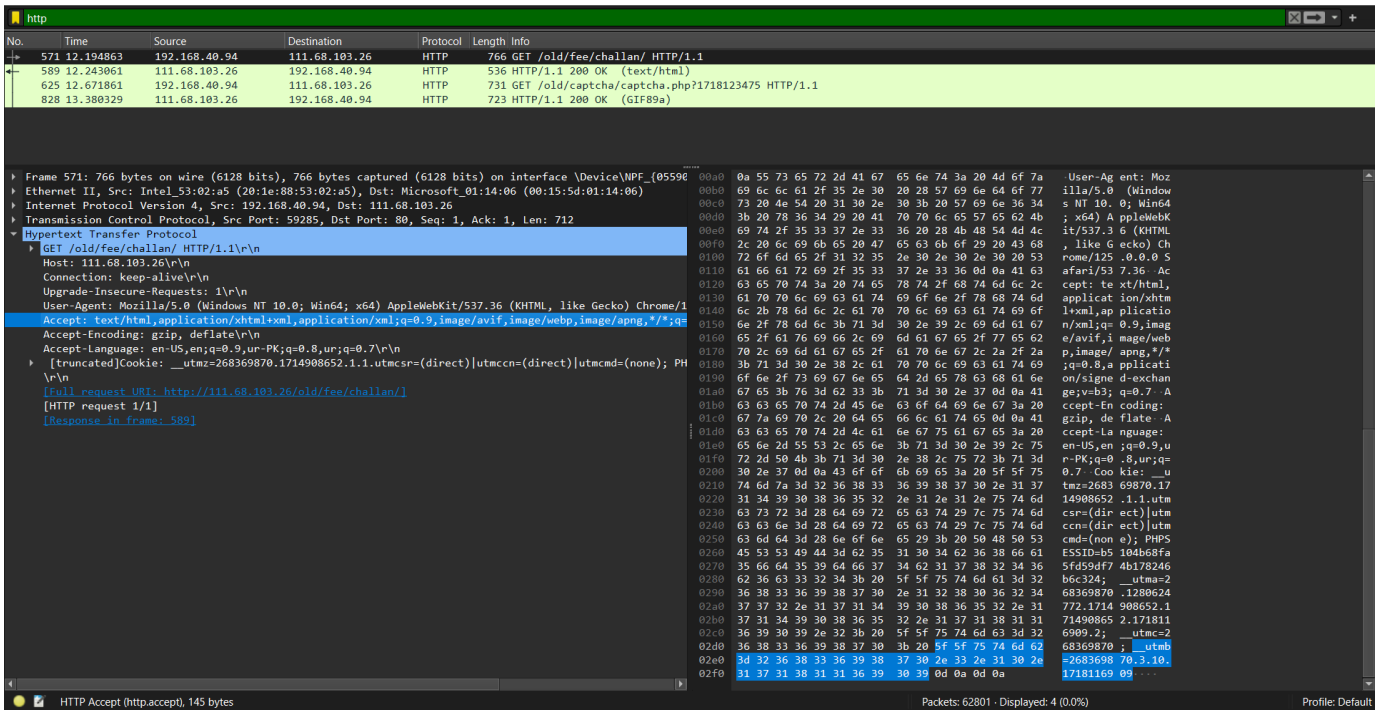


4. **Packet Bytes Pane:** Displays the raw data of the selected packet in both hexadecimal and ASCII formats. This pane allows you to see the actual bytes that were transmitted over the network.



- **Using Wireshark**
  - Once you open the Wireshark interface, in the main window, you'll see a list of network interfaces (like Ethernet, Wi-Fi).
  - Choose the interface you want to capture data from. If you're not sure, select the one with the most traffic (often Wi-Fi for laptops).
  - Once you select the interface, you'll see the packets being captured in real-time in the main window.
  - You can apply filters to focus on specific types of traffic. Type expressions in the filter bar at the top of window. Figure shows the packets filtered with http protocol only

- **Analyzing a captured packet:** Once you select a packet for analysis, you can view details of the selected packet and its raw data in packet detail pane and packet bytes pane respectively.



The bottom-left pane shows the details of the selected packet, and the bottom-right pane shows the data being transferred/ packet payload.

Disclaimer

The series of handouts distributed with this course are only for educational purposes. Any actions and or activities related to the material contained within this handout is solely your responsibility. The misuse of the information in this handout can result in criminal charges brought against the persons in question. The authors will not be held responsible in the event any criminal charges be brought against any individuals misusing the information in this handout to break the law.