# How To Make The Best Use Of Live Sessions

- Please login on time
- Please do a check on your network connection and audio before the class to have a smooth session
- All participants will be on mute, by default. You will be unmuted when requested or as needed
- Please use the “Questions” panel on your webinar tool to interact with the instructor at any point during the class
- Ask and answer questions to make your learning interactive
- Please have the support phone number (US : 1855 818 0063 (toll free), India : +91 90191 17772) and raise tickets from LMS in case of any issues with the tool
- Most often logging off or rejoining will help solve the tool related issues

# COURSE OUTLINE


## MODULE 5

INTRODUCTION TO  LINUX

INSTALLATION AND INITIALISATION

USER ADMINISTRATION

BOOT AND PACKAGE MANAGEMENT

**NETWORKING**

LINUX OVERVIEW AND SCRIPTING

LINUX FOR SOFTWARE DEVELOPMENT

SECURITY ADMINISTRATION

# Objectives

After completing this module, you should be able to:

- Understand about OSI model
- Learn about various protocols
- Understand about Packet Capturing tools
- Learn about Linux firewalls
- Know more about Linux Security

edureka!

# Networking

# OSI Layers

edureka!

# OSI Model

01 Open system Connection (OSI) is a model for how various application will interact over a network.

02 It divides the architecture in seven segments.

03 The lower layer deals with chunks of binary data and routing these data.

04 Higher levels are responsible for network requests, network representation and protocol's as seen from a user's point of view.

05 Each layer interacts with the layer above and below it.

edureka!

# OSI Layers

| | |
|---|---|
| Application | To allow access to network layers. Example – FTP, HTTP, DNS, SNMP |
| Presentation | To translate, encrypt and compress data. |
| Session | Establish, manage and terminate session. |
| Transport | Reliable process to process message delivery and error recovery. Ex – TCP, UDP |
| Network | Move packets from source to destination. Ex –IP, ICMP, ARP |
| Data Link | Organize bits into frame to provide hop-to-hop delivery. |
| Physical | To transfer bits over an electrical medium. |

# Protocols

edureka!

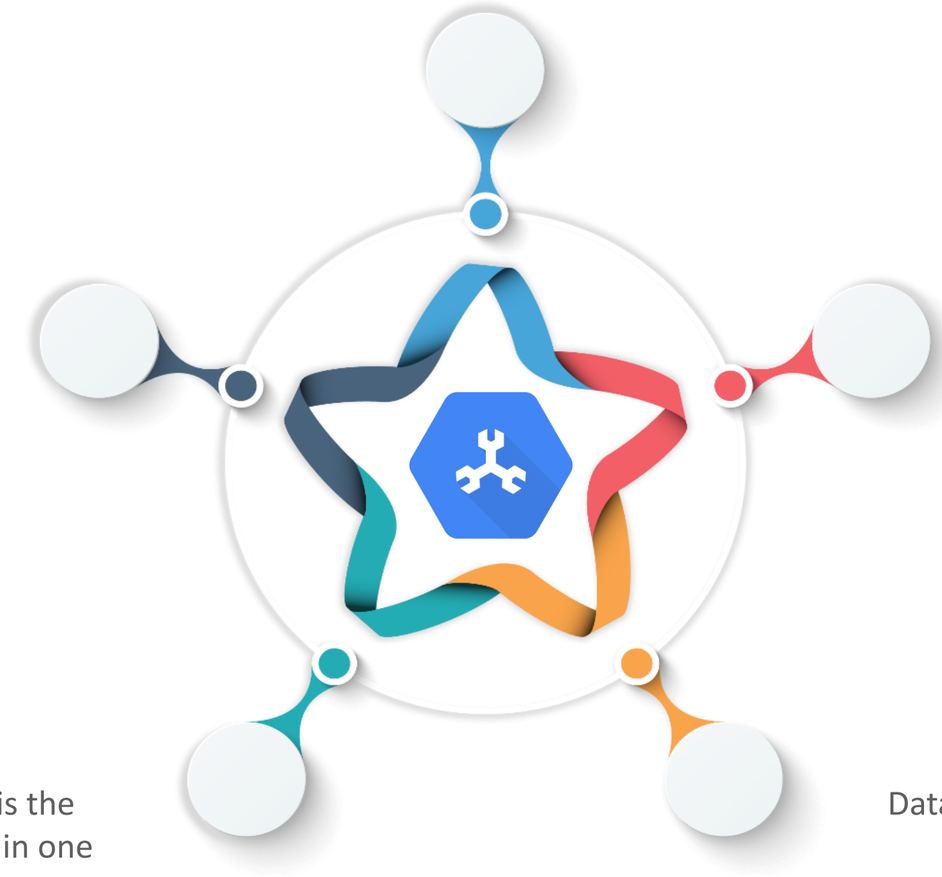
# TCP

- Transmission control protocol which delivers data from end-to-end in order, reliably.
- It establishes a virtual path between source and destination.
- It uses the underlying layer to deliver individual segments but control is with itself, and IP is unaware of the re-transmission or out-of-order packets.
- Each TCP port is bound to at most one socket, avoiding duplication.

# TCP Features

## Ordered data transfer.

It sends data as a stream of bytes with numbering.



## Rate Limits

The rate a sender sends data to guarantee reliable data.

## Retransmission of lost data

It uses an acknowledgement mechanism to check arrival of data.

Error free data transfer. Error control is received through checksum, acknowledgement and time out.

## Maximum segment size

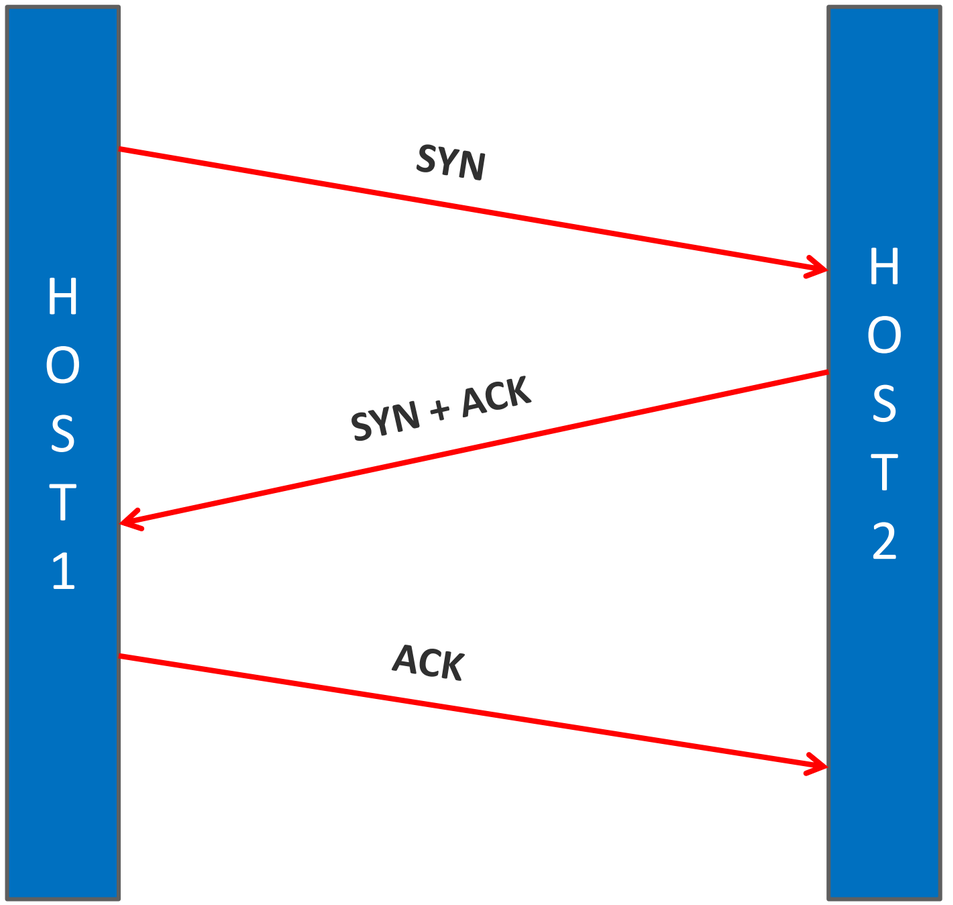
This is shared between both parties and is the largest amount of data in bytes to be sent in one single segment.

## Congestion control

Data sending rate also depends on the congestion of the network.

# TCP Connection Establishment

- SYN : It is synchronization for the sequence numbers. It carries one sequence number. No data is present.
- SYN + ACK : SYN for communication in other direction and ACK for received SYN. It consumes one sequence number.
- ACK : ACK segment for received SYN. It doesn't consume any sequence number.



# UDP

- User Datagram Protocol have a fixed size header of 8 bytes.
- Prior communications are not required to set up a communication channel.
- It has no hand-shaking, so it is unreliable. Hence, there is no guarantee of delivery.
- UD avoids the overhead of protocol processing like error detection.
- Time-sensitive application often use UDP where dropping packet is a better choice than waiting for the packet to be retransmitted.
- Live broadcast, multi-player games generally use UDP.

# UDP Features

UDP doesn't avoid congestion and it needs to be taken care at the application level.

Packets are sent individually and checked when received.

No connection establishment is done. Data is sent right away.

The messages are not ordered

It is very lightweight and fast.

It is useful for broadcast as connection establishment is not required.

# IPv4

Internet Protocol version 4 is used to identify devices on a network through an addressing system.

It is a connectionless protocol used on a packet switched network.

Ipv4 address is a 32-bit address uniquely defines the connection of a device universally.

It fragments packets when required.

It doesn’t guarantee end to end message reliability or flow control.

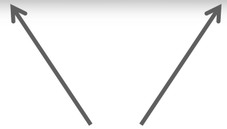
edureka!

# IPv4 Address

IPv4 are 32 bit integer and most commonly written in 4 octets with dot separated integer values.

IP address has the format Network.Host.

IP address = <network number><host number>



Each segment can range from 0-255.

Example:
123.56.234.12 is an IPv4 address.

Some of the numbers like 0.0.0.0 and 255.255.255.255 are reserved for default gateway or broadcast IP.

The network part is centrally administered by InterNIC and is unique through out the Internet.

# Issues With Ipv4

Most of the networks have already been assigned and there is a resource crunch.

Ipv4 doesn’t implement quality of service functionality.

IP packet can be easily sniffed from a network.

No authentication of user done at the server.

No encryption of data is done in packets.

Maximum packet size is of 65,535 bits which is smaller compared to the new faster networks.

# IPv6

Support for authenticity and privacy.

Simplified fixed size headers to process packets faster.

It has long addresses of 128 bits. It supports billion of users now.

IPv6 is the sixth revision to the Internet Protocol and follows IPv4.

Flow control and quality of service provides high reliability.

Encryption headers allows better security of data.

Extension headers allows large packet sizes.

Standard authorization of users to server.

edureka!

# Demo - IP Address

```
ubuntu@ubuntu#ifconfig
ens3      Link encap:Ethernet  HWaddr fa:16:3e:96:e5:1f
          inet addr:10.1.42.44  Bcast:10.1.42.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe96:e51f/64 Scope:Link
          inet6 addr: fcff:1:42:0:f816:3eff:fe96:e51f/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:27015 errors:0 dropped:5 overruns:0 frame:0
          TX packets:469 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14920237 (14.9 MB)  TX bytes:54811 (54.8 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:321 errors:0 dropped:0 overruns:0 frame:0
          TX packets:321 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:32331 (32.3 KB)  TX bytes:32331 (32.3 KB)

ubuntu@ubuntu#
```

# FTP

- File Transfer Protocol is standard protocol to transfer file over a computer network.
- It is built on client-server model and uses separate data and control connections.
- FTP is a two-way system. Transferring file from client to server is called uploading and from server to client is called downloading.
- In Unix, the service is provided by a daemon called ftpd which runs in background.
- It allows to have ownership and access restrictions.
- Come FTP client tools are Filezilla, dreamweaver, etc.

# TFTP

- Trivial file transfer protocol is a simpler version of FTP.
- It doesn’t provide user authentication.
- It works on client-sever model by establishing connection between them.
- Listing, renaming and deleting is generally not supported at TFTP.
- TFTP uses Udp port 69 while FTP uses TCP ports 20 and 21.
- There is a size restriction in TFTP which is quite smaller.

# Telnet

- It provides a bi-directional command line interface with a remote network.
- The configuration for telnet in Linux is in /etc/xinetd.d/telnet.
- The communication between client and server is handled by internal commands and which is not visible to the user.
- It is not a secure protocol.
- Once the connection is established, the client program sends commands with one character at a time.
- The Telnet server uses port 23 of TCP.

# SSH

Secure shell is a cryptographic protocol to secure data over a network.

A password sniffing attack led to development of initial version of SSH.

There is cryptographic integrity check via md5, sh1 keyed hash algorithms.

It allows users to securely login in an unsecure network and execute commands.

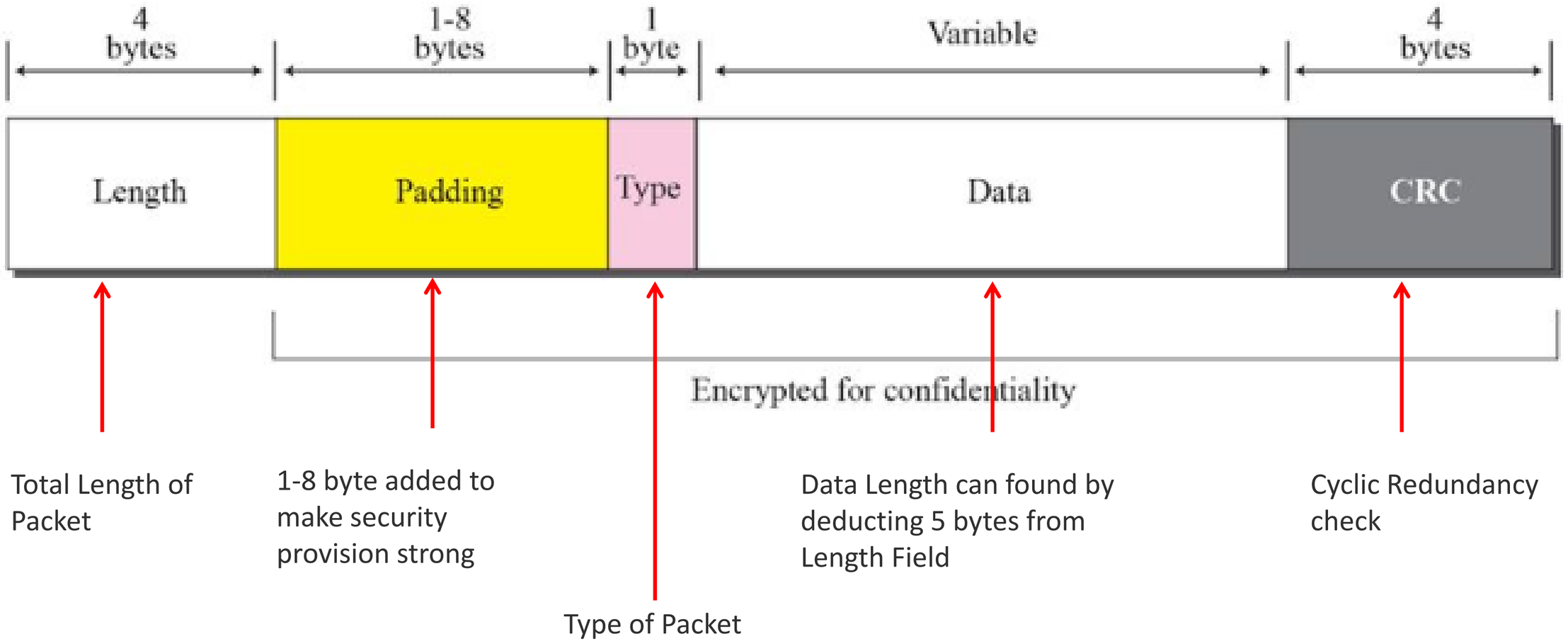
It is free for non-commercial use.

Has client-server communication over TCP and provides authentication and encryption.

It only supports the known port numbers.

edureka!

# SSH Packet Format



# HTTP - Hyper Text Transfer Protocol

An application layer protocol which is the base for data communication for World Wide Web.

Stateless.

Protocol for transferring various forms of data between server and the client like: plaintext, images, videos, sounds, etc.

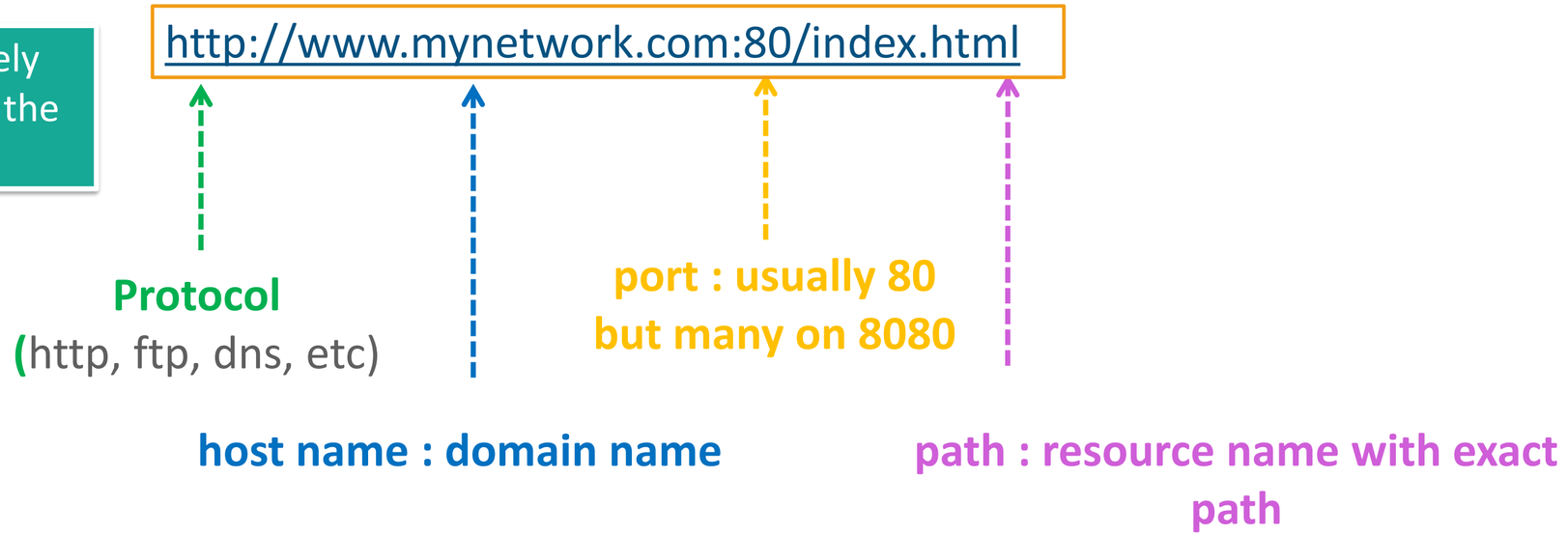
Supports authentication and stores it in the browser till the browser is closed or triggered to be deleted.

Works on port 80 of TCP.

edureka!

# HTTP - URLs

A URL is used to uniquely identify a resource over the web.




Syntax:
protocol : // <hostname> : port / path and filename

edureka!

# DNS

Domain name system provides a symbolic mapping between names and IP addresses in the world wide web.

DNS is a distributed system with many servers working worldwide.

Names are organized in a tree structure allowing faster retrieval.

A user will know the email-address but IP protocol needs IP address and DNS resolves the Email-address to corresponding IP-address to be sent to.

The name assigned to machines must be unique.

A name can consist of two or more components separated by a dot. Example – www.google.co.in

# DEMO - DNS

edureka!

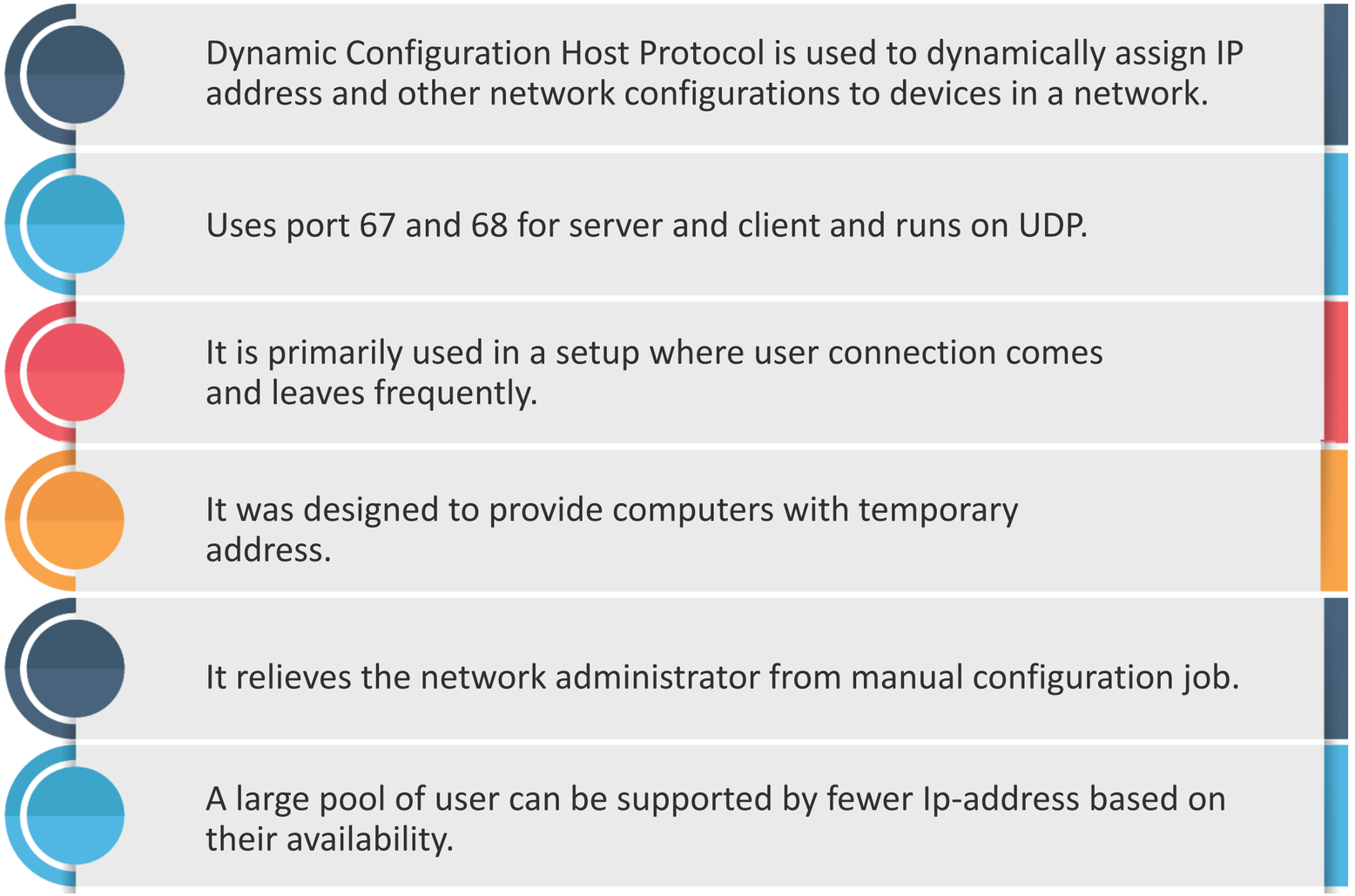
# Demo - DNS

8.8.8.8 IP has DNS google-public-dns-a.google.com

```
ubuntu@ubuntu#nslookup 8.8.8.8
Server:         2001:470:b368:1070::1
Address:        2001:470:b368:1070::1#53

Non-authoritative answer:
8.8.8.8.in-addr.arpa    name = google-public-dns-a.google.com.

Authoritative answers can be found from:

ubuntu@ubuntu#
```

edureka!

# DHCP




edureka!

# ARP

- Address Resolution Protocol is used for discovering the MAC address or link layer address.
- It associates an IP address to a physical layer address.
- ARP packet is encapsulated within an Ethernet packet.
- It keeps cache of resolved IP address for faster response to same resolution in future.
- The Arp cache is periodically flushed of all the entries.
- Proxy Arp is when a node responds on behalf of Arp request of another node. This is used to re-direct traffic to another system.

# Demo - arp

```
ubuntu@ubuntu#
ubuntu@ubuntu#arp -an | grep 20
? (172.16.20.215) at 30:e1:71:53:f1:44 [ether] on ens3
? (172.16.20.230) at 52:54:00:c1:86:d5 [ether] on ens3
? (172.16.20.154) at 00:0c:29:01:01:42 [ether] on ens3
? (172.16.20.29) at <incomplete> on ens3
? (172.16.20.1) at c4:64:13:cb:4a:c7 [ether] on ens3
? (172.16.20.218) at 52:54:00:29:c9:9d [ether] on ens3
ubuntu@ubuntu#
```

# ICMP

- IP does not have an inbuilt mechanism for sending error and control messages, so, it depends on the Internet Control Message Protocol(ICMP) to provide an error control.
- ICMP defines two types of messages:

**Error Message**

- Source quench
- Time exceeded
- Destination reachable
- Fragmentation required

**Informational Messages**

- Echo request/reply
- Address Mask request/reply
- Router Discovery

# DEMO - ICMP

edureka!

# Demo - ICMP

```
ubuntu@ubuntu#
ubuntu@ubuntu#ping 10.10.21.40
PING 10.10.21.40 (10.10.21.40) 56(84) bytes of data.
64 bytes from 10.10.21.40: icmp_seq=1 ttl=60 time=230 ms
64 bytes from 10.10.21.40: icmp_seq=2 ttl=60 time=230 ms
64 bytes from 10.10.21.40: icmp_seq=3 ttl=60 time=230 ms
64 bytes from 10.10.21.40: icmp_seq=4 ttl=60 time=230 ms
64 bytes from 10.10.21.40: icmp_seq=5 ttl=60 time=230 ms
64 bytes from 10.10.21.40: icmp_seq=6 ttl=60 time=229 ms
^C
--- 10.10.21.40 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5004ms
rtt min/avg/max/mdev = 229.969/230.114/230.214/0.486 ms
ubuntu@ubuntu#
```

# Demo - traceroute

- Packet routing from host machine to google.com(8.8.8.8)

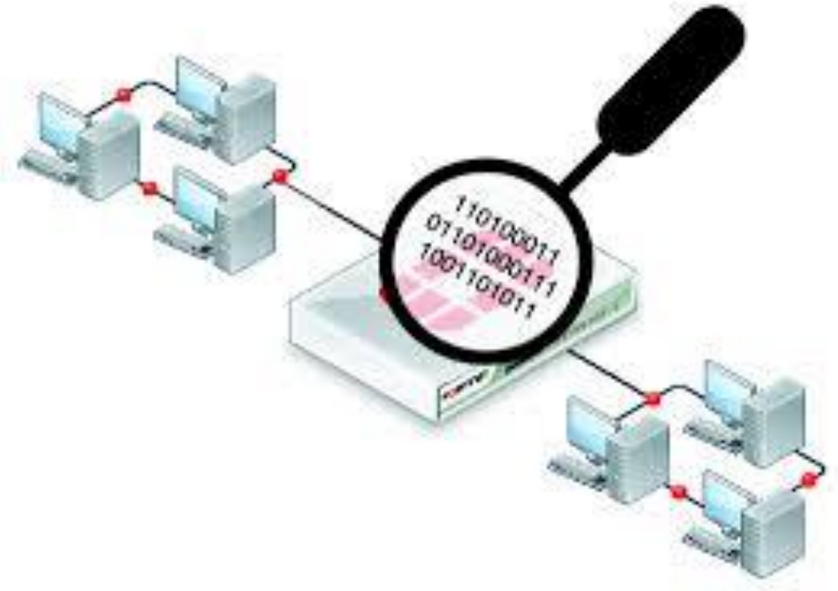
```
ubuntu@ubuntu#traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  172.16.20.1 (172.16.20.1)  2.116 ms  2.128 ms  2.263 ms
 2  10.3.9.105 (10.3.9.105)  11.784 ms  11.790 ms  11.897 ms
 3  182.74.59.185 (182.74.59.185)  2.175 ms  3.388 ms  6.257 ms
 4  182.74.25.145 (182.74.25.145)  2.084 ms  1.971 ms  2.053 ms
 5  182.79.177.241 (182.79.177.241)  7.384 ms  7.665 ms 182.79.198.218 (182.79.198.218)  12.328 ms
 6  72.14.242.178 (72.14.242.178)  7.910 ms  7.959 ms  7.783 ms
 7  108.170.253.105 (108.170.253.105)  7.717 ms 74.125.242.147 (74.125.242.147)  8.593 ms 108.170.253.122 (108.170.253.122)  8.904 ms
 8  216.239.63.213 (216.239.63.213)  43.930 ms 216.239.63.211 (216.239.63.211)  43.792 ms 216.239.48.71 (216.239.48.71)  40.289 ms
 9  216.239.48.47 (216.239.48.47)  40.804 ms  39.877 ms 216.239.48.73 (216.239.48.73)  39.732 ms
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  google-public-dns-a.google.com (8.8.8.8)  39.894 ms  39.501 ms  39.507 ms
ubuntu@ubuntu#
```

# Packet Capturing Tools

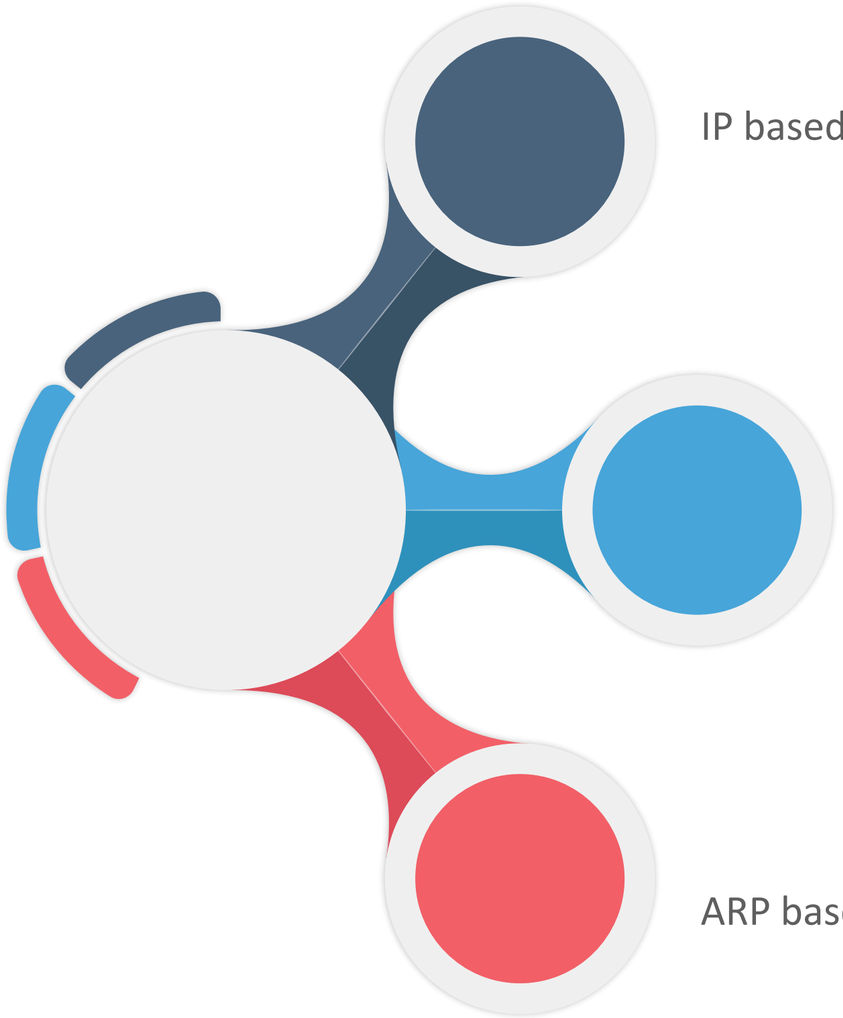
edureka!

# Packet Sniffing

- It is a technique to monitor every packet that crosses the network, in promiscuous mode
- Some common sniffing tools are tcpdump, Ip spoofing, etc.
- It can be used to trace IP address to their destination
- One can monitor all internet traffic from your system to the Internet



# Sniffing Methods



IP based sniffing

MAC based sniffing

ARP based sniffing

edureka!

# Packet Sniffing Uses

Detection of clear text password and username from network.

Conversion of data sent over the network to human readable format

Performance analysis to improve the network.

Used to make network secure as some malicious content can be filtered out.

Network intrusion detection to find if someone has unauthorized connection to our system.

Finding out the details about the different system connected to your system.

# Wireshark


1. It is the most popular network analyzer.
2. It lets you capture traffic and analyze the network on your computer.
3. It is available for both Linux and Windows.
4. Displays data with detailed protocol information.
5. Lots of filter options available to sort out the relevant information.
6. Colorize packets based on the filters.
7. It doesn’t manipulate anything and just provide measurements.

# tcpdump

It is a popular packet analyzer that runs in command line.

It is a free tool and works on most Linux like operating systems.


It uses the libpcap library to capture packets.

It is an ip-utility tool to sniff packets.

TCPdump does not display all the fields of the captured data. Use option (-x) that dumps the entire datagram captured with the default snaplen in hexadecimal.

edureka!

# Tcpdump Command Line Options

| | |
|---|---|
| -i | : tells the interface we are using. |
| -q | : stay quiet rather than printing more info. |
| -t | : remove time stamp. |
| -l | : buffers one line at a time on output. |
| -c | : count of a packet to capture. |
| -w | : write to a file than printing on the screen. |
| -r | : read the contents of the file. |

# Demo - tcpdump

```
ubuntu@ubuntu#tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens3, link-type EN10MB (Ethernet), capture size 262144 bytes
23:17:40.870230 IP 10.1.42.44.ssh > 10.3.25.88.59419: Flags [P.], seq 3863945931:3863946143, ack 345724452, win 251, length 212
23:17:40.874300 IP 10.1.42.44.ssh > 10.3.25.88.59419: Flags [P.], seq 212:392, ack 1, win 251, length 180
23:17:40.874335 IP 10.1.42.44.ssh > 10.3.25.88.59419: Flags [P.], seq 392:556, ack 1, win 251, length 164
23:17:40.874356 IP 10.1.42.44.ssh > 10.3.25.88.59419: Flags [P.], seq 556:720, ack 1, win 251, length 164
23:17:40.874375 IP 10.1.42.44.ssh > 10.3.25.88.59419: Flags [P.], seq 720:884, ack 1, win 251, length 164
^C23:17:40.882143 IP 10.1.42.236.2222 > 239.1.1.1.2222: UDP, length 1024

6 packets captured
947 packets received by filter
935 packets dropped by kernel
```

# Netstat

It is a command used to display very detailed information about how your computer is interacting with other network devices

- Netstat provides statistics for the following:
  - The name of the protocol (TCP or UDP).
  - The IP address of system and port number being used. An asterisk (*) is shown for the host if the server is listening on all interfaces.
  - If the port is not yet established, the port number is shown as an asterisk(*).
  - The IP address and port number of the remote network to which the socket is connected.
  - Indicates the state of a TCP connection.
  - The possible states are as: CLOSE_WAIT, CLOSED, ESTABLISHED, FIN_WAIT_1, FIN_WAIT_2, LAST_ACK, LISTEN, SYN_RECEIVED, SYN_SEND, and TIME_WAIT.

# Netstat Command Line Options

| | |
|---|---|
| -a | : displays all active connections. |
| -t | : displays only tcp connections |
| -l | : display all active listening ports. |
| -u | : display all udp connections. |
| -s | : display statistics by protocol. |
| -r | : display kernel routing table |
| -i | : display network interface packet transactions. |
| -p | : display which process is using which socket |
| -n | : no dns resolution done |
| -c | : to keep printing netstat information every few seconds. |

# Demo - Netstat

```
ubuntu@ubuntu#netstat -ltn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:2022            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:34886           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8008            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:2024            0.0.0.0:*               LISTEN
tcp        0      0 10.1.42.44:44555        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 10.1.42.44:10480        0.0.0.0:*               LISTEN
tcp        0      0 10.1.42.44:12112        0.0.0.0:*               LISTEN
tcp        0      0 10.1.42.44:10352        0.0.0.0:*               LISTEN
tcp        0      0 10.1.42.44:19216        0.0.0.0:*               LISTEN
tcp        0      0 10.1.42.44:12080        0.0.0.0:*               LISTEN
tcp        0      0 10.1.42.44:19184        0.0.0.0:*               LISTEN
tcp        0      0 10.1.42.44:10384        0.0.0.0:*               LISTEN
tcp        0      0 10.1.42.44:10288        0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:4565          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:26582           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:5500          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:39580           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:55679           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN
```

edureka!

# Linux Firewall

edureka!

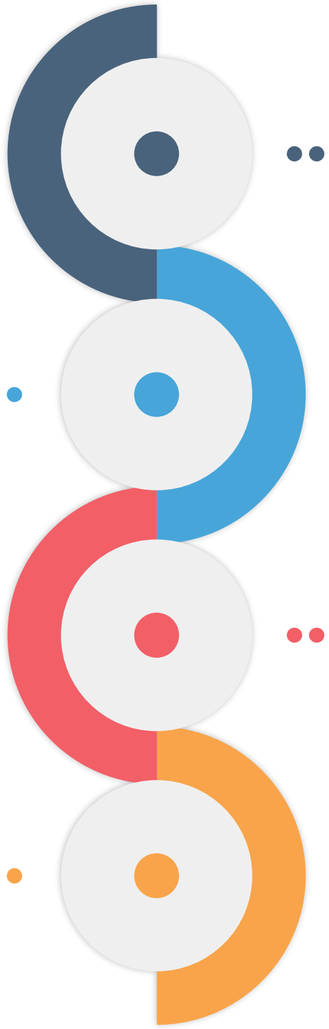
# Firewall

A firewall is a program which restricts the incoming and outgoing connections based on rules that have been pre-configured in the rule-set

- Firewall can log accesses and provide valuable information about network usage
- Firewall can be hardware level or at software level. Software level firewall are cheaper and easier to configure
- It reduces risks by blocking the insecure services reducing chances of malicious content entering system
- The default Linux firewall is **iptables**
- Multiple GUI version are also available for Linux

# iptables

It is a firewall package that comes along with the Linux distro



It is fast and reliable

Stateful packet inspection is done

Packets are filtered according to IP address and MAC Address

Rate limiting feature which helps iptables block denial of service attacks

edureka!

# Iptable

Built-Ins

- Input : packets desired for local socket
- Forward : packets routed through system.
- Output : packets generated locally.
- Prerouting : used for packets as it is received.
- Postrouting : used to packets going out.

# Iptable Subgroups

## Filter

- This is the default table and contains built-in chains.
- Supports INPUT, FORWARD, OUTPUT

## NAT

- Table consulted when packet is trying for a new connection.
- Supports PREROUTING, POSTROUTING, OUTPUT

## Mangle

- This table is used for altering thepackets.
- Supports INPUT, PREROUTING, FORWARD, OUTPUT, POSTROUTING

Three types of subcategories

# Iptable - start And stop

Use command start to start a firewall in Linux

**Syntax**

systemctl start iptables

Example : # systemctl start iptables

**To stop the firewall use ‘stop’**

# systemctl stop iptables

**Use ‘restart’ to stop and start the firewall.**

# systemctl restart iptables

# Iptables - Check Rules

To check existing Firewall rules in the system

**Syntax**

iptables –L

**For rules in a particular table use '-t' option**

# iptables –t nat –L

**Use '-n' option to get numeric representation of address and ports**

# iptables –L –n
# iptables -Ln

edureka!

# Block Specific IP

To block a specific IP append the IP in the rule

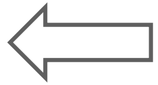
**Syntax**

iptables –A INPUT –s <ip_address> -j DROP

Example :
# iptables –A INPUT –s 10.1.23.123 –j DROP

**Syntax**

# iptables –A INPUT –p tcp –s <ip_address> -j DROP



If you want to block a particular traffic from that ip address use '-p' option.

'-A' option is used to append at the end of rule.

# Unblock IP From Rule

Use ‘-D’ to delete an entry from the rule

**Syntax**

iptables –D INPUT –s <ip_address> -j DROP

Example :
# iptables –D INPUT –s 10.1.23.123 –j DROP

**To remove all iptables rule use ‘-F’ option.**

# iptables –F

**To remove rules from a specific table use ‘-t’ option.**

# iptables –t nat -F

edureka!

# Allow Ports

- To allow incoming connections, use :

**Syntax**

iptables –A INPUT –p tcp - -dport <port_no> -j ACCEPT

Example :
# iptables –A INPUT –p tcp - -dport 80 –j ACCEPT

**To allow multiple port at once use ‘multiport’ option.**

# iptables –A OUTPUT –p tcp –m multiport - -sports 21,22,80 –j ACCEPT

**To limit a particular network on a particular port .**

# iptables -A OUTPUT -p tcp -d 192.168.100.0/24 --dport 22 -j ACCEPT

edureka!

# DEMO - iptables

edureka!

# Demo - Iptables

```
ubuntu@ubuntu#iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ubuntu@ubuntu#
ubuntu@ubuntu#iptables -A INPUT -p tcp --dport 80 -j ACCEPT
ubuntu@ubuntu#
ubuntu@ubuntu#iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

# Demo - Block A Particular Domain

- Get the Ip details of the domain

```
ubuntu@ubuntu#host google.com
google.com has address 216.58.192.14
google.com has IPv6 address 2607:f8b0:4004:811::200e
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.
google.com mail is handled by 20 alt1.aspmx.l.google.com.
google.com mail is handled by 10 aspmx.l.google.com.
google.com mail is handled by 40 alt3.aspmx.l.google.com.
ubuntu@ubuntu#
```

- Add an entry for the IP. (a particular website may run from multiple IP address too)

```
ubuntu@ubuntu#iptables -A OUTPUT -p tcp -d 216.58.192.14 -j DROP
ubuntu@ubuntu#iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             nuq04s29-in-f14.1e100.net
ubuntu@ubuntu#
```

edureka!

# Demo - Blocking A Particular Domain

- Restricting google-Ip, 8.8.8.8 and not being able to ping it.

```
ubuntu@ubuntu#iptables -A INPUT -s 8.8.8.8 -j DROP
ubuntu@ubuntu#ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
--- 8.8.8.8 ping statistics ---
25 packets transmitted, 0 received, 100% packet loss, time 24272ms

ubuntu@ubuntu#iptables -D INPUT -s 8.8.8.8 -j DROP
ubuntu@ubuntu#
ubuntu@ubuntu#ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=45 time=108 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=45 time=86.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=45 time=84.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=45 time=91.0 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 84.809/92.742/108.588/9.440 ms
ubuntu@ubuntu#
```

# Linux Security

edureka!

# SSH Security

- Ssh provides multiple levels of security as:
  - Guarantee that the data will not be altered during the transition.
  - Authentication of client and server is done.
  - Strong encryption so difficult to read message content.

# SSH Session

A client-server model is used during establishment of connection between two parties and encrypt the data between them

**Syntax**

ssh <username>@<server_ip_address>

Example :
# ssh ubuntu@10.1.10.123

- A SSH session works in two stages:
  - To sync the encryption to be used for any further communication.
  - To authenticate the user for access rights to be given.
- The server waits on the configured port for connections.
- The client initiates the TCP handshake with the server.

# Encryption For SSH Session

- When TCP connection is started, a session key is generated using diffie-Hellman algorithm.
- Procedure for Diffie Hellman :
  - A big prime number is chosen and shared by both parties.
  - An encryption mechanism(AES generally) is chosen.
  - Each party chose another prime number which is kept secret and used for private key.
  - A public key is generated by private key, encryption mechanism, and shared key and is shared with the other party.
  - Both party uses their private key, received public key and original shared key to compute a secret key. It will result in same shared secret key.
  - The shared secret key is used to encrypt all the communications.
- Public key can be shared as there is no means of deriving the private key from it.

# Authenticating User

- SSH key pairs are used for authentication which are asymmetric key function
- The procedure for authenticating the user is:
  - The client sends an ID for the key pair.
  - Server checks for authorized key file for the key ID.
  - If match found it generates a random number and uses public key to encrypt the number.
  - Server sends the client this encrypted number.
  - If client has the private key then it will be able to decrypt the message to find the original number.
  - The client finds the MD5 value for this number combined with shared key that is used to encrypt.
  - Clients sends the md5 value to the server.
  - Server does its own calculation and if the two value matches, client is authenticated.

# Configure Host Certification

**Step-1** : Generate signing key by using ssh-keygen.

# ssh-keygen –f ssh_key.

**Step-2** : It will generate two keys, ssh_key and ssh_key.pub.

**Step-3** : Sign the host key certificate:

**Syntax**

ssh-keygen –s signing_key –I key_identifier –h –n host_name –V +52w host_rsa_key

Example :
# ssh-keygen -s ssh_key -I key_server -h -n example.com -V +52w /etc/ssh/ssh_key.pub

**It will generate a ssh_key_server.pub in /etc/ssh.**

**Step-4** : We need to copy the certificate to the host server ad we can do it by file transfer like scp.

**Step-5** : We can delete public key and certificate from the server.

# Configuration To Use Host Certification

**Step-1** Open the sshd_config file (not the ssh_config) present at /etc/ssh.

**Step-2** Find the line starting with “HostCertificate” and modify it or if not found append the ssh_key_server.pub at the end of the file.

**Step-3** Restart the ssh service.

```
# service ssh restart
```

**Step-4** Copy the value from ssh_key.pub and add it in ~/.ssh/known_hosts.

**Step-5** Modify it to @cert-authority *.example.com <key_value_read_from_ssh_key.pub>

# SCP

- Secure copy(SCP) allows to copy files over ssh connections.

**Syntax**

```
scp <username>@<server_ip>:/home/ubuntu/<filename> <username2>@<server_ip2>:/home/ubuntu/<filename>
```

Example :
# scp ubuntu@10.1.10.123:/home/ubuntu/file.txt ubuntu@10.1.23.123:/tmp/file.txt

- In case if we are logged in a particular server we just need to give file path for that block.
- SCP can be used in three ways :
  - Remote server to your computer.
  - Your computer to a remote server.
  - Remote server to a remote server.

# SCP Command Options

Some of the scp command options are :

-r : recursively copy the complete directory.

-p : port to be used for the connection.

-q : disables the progress meter.

-l : limits the bandwidth rate

-4 : use ipv4 protocol only.

-6 : use ipv6 protocol only.

-c : selects the ciphering to use for encrypting the data transfer.

Scp exits and return 0 on success and > 0 is error occurs.

# Remote Log-in

edureka!

# SSH Log-in

- SSH can be used to log-in to a remote server.

**Syntax**

```
ssh <username>@<remote_ip_address>
```

Example :
# ssh ubuntu@10.1.10.123

- If the port is modified for ssh on the server side, then add port with '-p' option.

  # ssh –p 1234 ubuntu@10.1.10.123

- If wish to run a single command on remote system, then append the command to it.

  # ssh ubuntu@10.1.10.123 'reboot'

- To exit, type the keyword 'exit'

# VNC

- Virtual Network computing is used to allow a user to view and interact one computer using some tool on another computer.
- The differences in VNC and other remote displays systems are :
  - It is fully cross-platform.
  - It is small, simple and can be run from small memory devices.
  - It is open source so can be downloaded for free under GNU-GPL license.
- VNC server is the program on system which allows to share its screen.
- VNC client is the program that views and interacts with the server.

# VNC Program

VNC client needs a reliable protocol like TCP and a way of displaying pixels so is simple to create.

VNC client is present in almost all flavors of Linux.

VNC server is more complex as the OS for remote may vary and it needs to supports for all.

On Windows, it is even harder to create VNC server due to its restrictions.

VNC only displays keystrokes and mouse movements.

It doesn’t support file transfer or related services.

# More Linux Utilities

edureka!

# Dnsmasq

It is lightweight suitable for resource constrained routers and firewalls.

It is designed to provide DNS and DHCP services.

Dnsmasq's DHCP server supports static and dynamic IP allocation.

DHCP server is integrated with DNS server and allows systems to check DHCP allocated addresses in the DNS queries.

Performance is increased as it caches the DNS records.

DNS restrictions can be made by modifying /etc/hosts.

# Dndmasq Setup

Step-1

- Install dnsmasq => # apt-get install dnsmasq

Step-2

- Setup dnsmasq as DNS DHCP. Modify /etc/dnsmasq.conf as necessary.

Step-3

- In the file '/etc/dnsmasq_static_hosts.conf' you can add a list of local machines with static IP addresses in the same format as the /etc/host file.

Step-4

- Stop and start the dnsmasq service.

  # service dnsmasq stop

  # service dnsmasq start

# dnsmasq.conf Options

domain-needed : If the name is not in the local /etc/hosts file then “not found” will be returned.

bogus-priv : Reverse IP (192.168.x.x) lookups not found in /etc/hosts will be returned as “no such domain”.

no-resolv : Do not read resolv.conf to find the servers where to lookup dns.

no-poll : Do not poll resolv.conf for changes.

server=8.8.8.8 : Set one or more DNS servers to use when addresses are not local.

local=/<domain>/ : Our local domain, queries in these domains are answered from /etc/hosts.

domain : This is your local domain name.

dhcp-range : This is the range of IPs that DHCP will serve.

dhcp-lease-max=25 : defines how many leases can be active at one time.

dhcp-option=option:router,192.168.0.1 : A host is requesting IP address via DHCP, tell the gateway to use.

edureka!

# Demo - dnsmasq.conf

```
domain-needed
bogus-priv
no-resolv
expand-hosts
domain=lfc.com
local=/lfc.com/
listen-address=127.0.0.1
listen-address=10.10.1.123

dhcp-range=10.10.1.100,10.10.1.150,12h
dhcp-lease-max=25

dhcp-option=eth,3,10.10.1.50
dhcp-option=wifi,3,10.10.1.60



"/etc/dnsmasq.conf" [Modified] 675 lines --91%--
```

edureka!

# Samba Server

Samba is an open-source program which provides file and print service to client

It free under the GPL license.

With Samba, we share a Linux file system to Windows and vice-versa.

Server Message Block(SMB) is the protocol by which machines share file and printer information.

Samba sets up network shares for Linux directories and they are viewed as normal folders in Windows.

The configuration file for Samba is present in /etc/samba/smb.conf.

Each folder can have different access permissions.

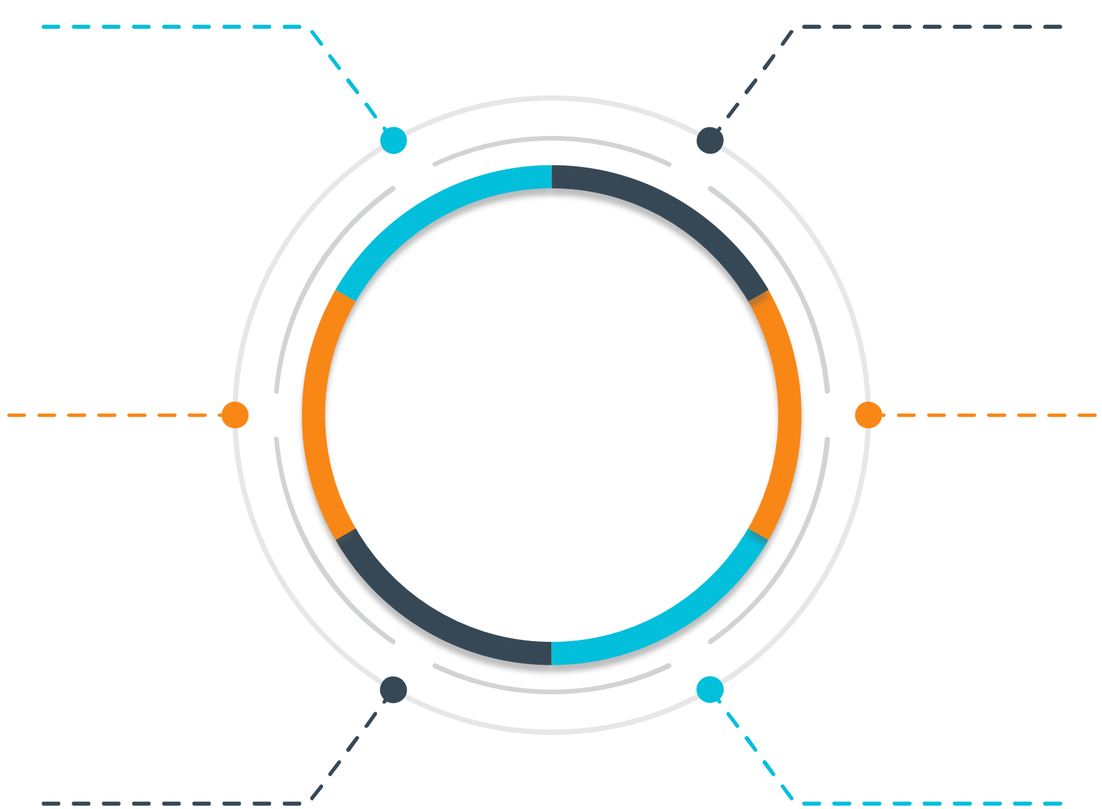
edureka!

# Webserver

A web server handles requests from internet over the HTTP protocol.

An agent, web crawler initiates a request for a specific resource over HTTP and server responds back with the data in the resource block.

A web server has a defined load limit and can serve a limited requests per second.



It stores and process web pages to the client system.

Most generic webserver has support for scripting languages like PHP, Python, etc.

The most commonly used webserver is Apache.

edureka!

# Netcat

- It is used to monitor, test and send data across network connections using TCP/UDP.

**Syntax**

netcat <option> host port

Example :
# netcat example.com 2000

- It is primarily used to :
  - Check outbound or inbound connections over TCP/UDP.
  - DNS forward/reverse checking, with appropriate warnings.
  - Can read command line arguments from standard input.
  - Hex dump of transmitted and received data.
  - Let another program service establish connections.

# Netcat Command Options

-i : delay time interval between lines of text sent and received.

-l : Should listen for an incoming connection.

-n : Do not do any DNS or service lookups.

-z : Should just scan for listening daemons, without sending any data to them.

-u : Use UDP instead of the default option of TCP.

-p : Specifies the source port to be used.

-d : Do not attempt to read from stdin.

# Quiz

1. _____ is the protocol suite for the current Internet?

   a. UNIX

   b. ISO

   c. ACM

   d. TCP/IP

# Answers

A

1. _____ is the protocol suite for the current Internet?

   a. UNIX

   b. ISO

   c. ACM

   d. TCP/IP

**Answer D : TCP/IP is used for most communication on Internet.**

# Quiz

2. What does command ‘tcpdump -w comm.pcap -i eth0 dst 10.10.1.123 and port 5555’, do ?

    a. Send messages to 10.10.1.123.

    b. Drop all packets from 10.10.1.123 on port 22.

    c. Bind 10.10.1.123 on eth0.

    d. Capture packets from 10.10.1.123 on port 5555 and write in comm.pcap.

# Answers

A

2. What does command ‘tcpdump -w comm.pcap -i eth0 dst 10.10.1.123 and port 5555’, do ?

   a. Send messages to 10.10.1.123.

   b. Drop all packets from 10.10.1.123 on port 22.

   c. Bind 10.10.1.123 on eth0.

   d. Capture packets from 10.10.1.123 on port 5555 and write in comm.pcap.

**Answer D: tcpdump captures packets from the mentioned IP and port and can write it in a particular file or display on console.**

# Quiz

3. Which of the following commands can be run to remove all the rules in an iptable table?

   a. iptables -L

   b. iptables -F

   c. iptables –A

   d. iptables -delete

# Answers

A

3. Which of the following commands can be run to remove all the rules in an iptable table?

   a. iptables -L

   b. iptables -F

   c. iptables –A

   d. iptables -delete

**Answer B: Iptables –f flushes out all entries from the iptables.**

# Summary

- In this module, you should have learnt:
  - OSI model
  - Various protocols
  - Packet Capturing tools
  - Linux firewalls
  - Linux Security


edureka!


edureka!


edureka!

# Thank You

For more information please visit our website
www.edureka.co