

Project Members and Distribution of Work:

Sr.No	Name	Work Done
1)	Jeevanandh Ravi	DNS, IPsec VPN tunnelling, Backup server
2)	Chetan Pavan Sai Nannapaneni	Web server, MITM using scapy, NFS file sharing
3)	Nilraj Mayekar	DHCP, Firewall

Index:

Sr.No	Content Description
1)	Introduction
2)	Design of the project (Block Diagram)
3)	Ip addressing scheme
4)	Brief Introduction on Topics: <ul style="list-style-type: none">a) DHCP<ul style="list-style-type: none">- Definition- Protocol Behavior- Signaling- Commands Used- Testing example and screenshotsb) DNS<ul style="list-style-type: none">- Definition- Protocol Behavior- Signaling- Commands Used- Testing example and screenshotsc) Webserver<ul style="list-style-type: none">- Definition- Protocol Behavior- Signaling- Commands Used- Testing example and screenshotsd) Firewall<ul style="list-style-type: none">- Definition- Protocol Behavior

	<ul style="list-style-type: none">- Signaling- Commands Used- Testing example and screenshots e) Backup <ul style="list-style-type: none">- Definition- Protocol Behavior- Signaling- Commands Used- Testing example and screenshots
5)	Brief Introductions on Topics: <ul style="list-style-type: none">a) File System in Linux, Tar, Gunzipb) Memoryc) Processes
6)	Bonus Part <ul style="list-style-type: none">a) Man in the Middle attack using scapyb) IPsec VPN between 2 VMs using OpenSwan/Strongswan-3c) Implement NFS for file sharing
7)	Future Scope
8)	References

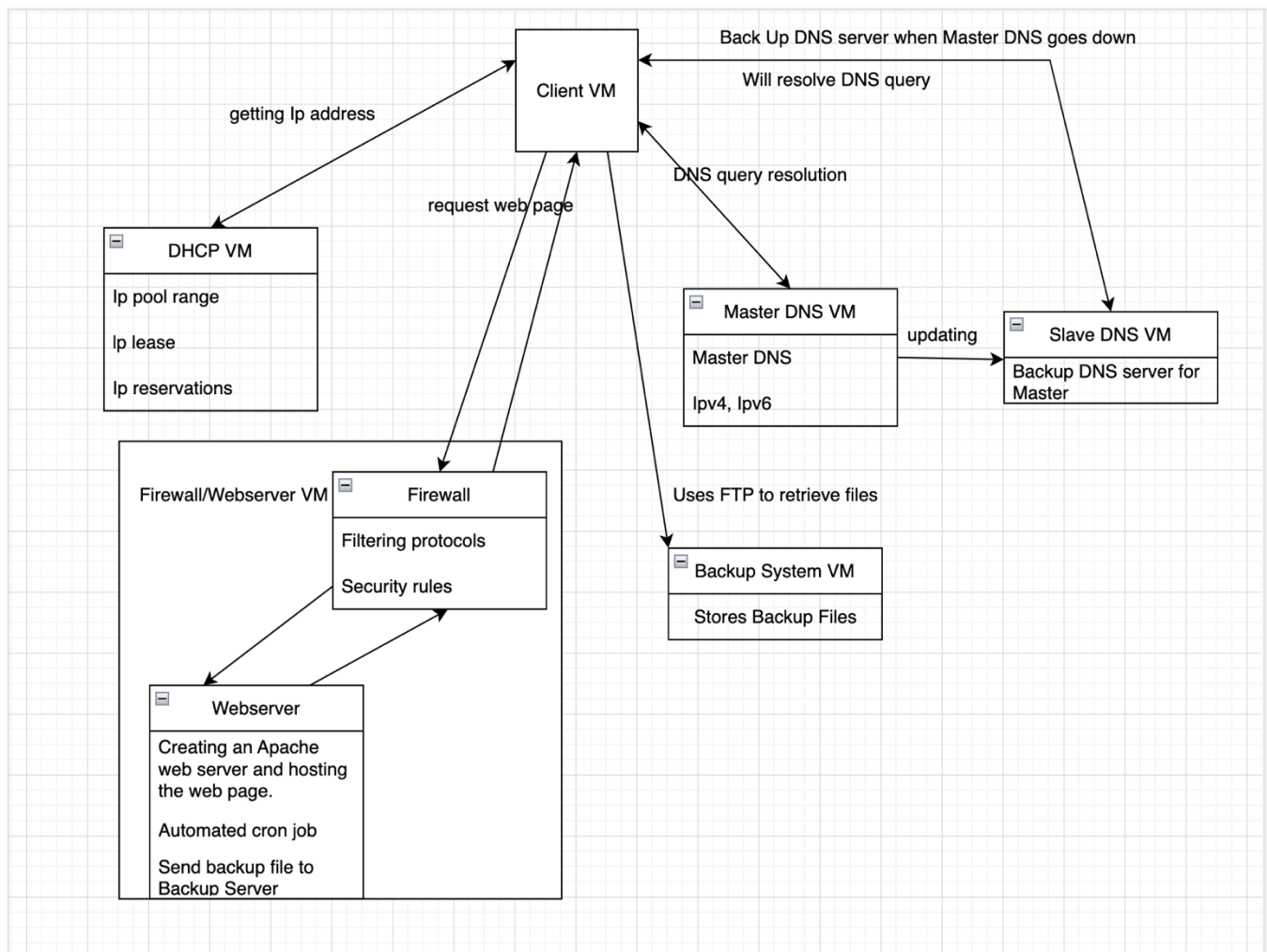
Introduction:

This telecom project provides a detailed architecture of the network infrastructure for a Boston-based start-up. It includes DNS implementation, DHCP server configuration, web server deployment, firewall protection, and automatic backup systems as part of the core network services. It tries to provide a robust and secure digital ecosystem that showcases some useful concepts in telecoms engineering. The DNS implementation is the core of the project, which requires establishing a Bind9 DNS Server with master and slave configurations. This includes creating reverse domains; using IPv4 and IPv6 addresses to create DNS records; and ensuring that automatic updates of servers are affected. The DHCP focuses on effective IP address management through the development of a server that can handle both IPv4 and IPv6 addressing-including scope setup, pool management, and reservations. The web server and firewall part of the project involves providing participants with the challenge of making a secure server environment with very limited resources, which in itself makes security a key component. This includes configuring a web server using command-line utilities and implementing various security measures such as protocol and IP filtering. The project also incorporates an automated backup system to ensure data preservation and system dependability by compiling zipped backup files and transferring them to an alternate server. The project focuses on integration and practical implementation; all the elements must work together as a whole. Participants are supposed to demonstrate web page access, firewall blocking, DNS

resolution, DHCP IP leasing, and backup file transfer. This holistic approach guarantees the student's deep understanding of how different network services interact throughout an entire ecosystem. Other optional projects involve setting up an IPsec VPN between Linux workstations, installing and configuring NFS to share files, and conducting a Man-in-the-Middle attack using ARP poisoning. Such challenging assignments provide opportunities for exploring more complex networking concepts and security protocols.

This project needs to have every step documented, along with any design, IP addressing, protocol behavior, and results of any testing, since at the culmination will be a full report and presentation of the findings. With this project, participants especially receive practical exposure to telecommunication engineering with the virtual build and management of corporate network infrastructures.

Design of the Project:



Ip addressing scheme:

Role	Static (IPv4)	Static (IPv6)	Subnet Mask	Gateway
DHCP	10.0.2.3/24	fd00::5/64	255.255.255.0	10.0.2.1
Master DNS	10.0.2.7	fd00:1	255.255.255.0	10.0.2.1
Slave DNS	10.0.2.8	fd00:2	255.255.255.0	10.0.2.1
Webserver	10.0.2.9	fd00:3	255.255.255.0	10.0.2.1
Backup server	10.0.2.10	-	255.255.255.0	10.0.2.1

DHCP pool range for the connecting clients will be from 10.0.2.2 to 10.0.2.50

Brief Introduction on Topics:

a) Dynamic Host Configuration Protocol:

The Dynamic Host Configuration Protocol, or DHCP, is a networking protocol developed to allow the automatic assignment of IP addresses and other configuration to devices on a network. It reduces human error in configuring devices for access to the network. In environments where devices join and leave the network quite often, such as businesses, schools, and public hotspots, DHCP is essential. DHCP implements a client-server architecture. A pool of IP addresses is maintained by the DHCP server, which allocates them to clients—those devices seeking network access—based on their needs. Besides the IP addresses, the server can share crucial configurations such as the subnet mask, default gateway, and DNS server details. This ensures that not only will devices be able to reach outwards but also communicate effectively within the network, all without manual intervention.

Protocol Behavior:

The DORA process—Discover, Offer, Request, and Acknowledge—is a four-step sequence that controls how the DHCP protocol operates. Devices can dynamically acquire IP addresses thanks to this methodical approach:

- 1) **DISCOVER:** To discover the availability of any DHCP servers, on first connecting to a network a device broadcasts a DHCPDISCOVER message. As a device doesn't have an IP address initially, it will broadcast this message to the network's broadcast address.
- 2) **Offer:** Any DHCP server that receives this broadcast sends out a DHCPOFFER message. This message includes the subnet mask, lease period, available IP address, and any other required configuration information. The server temporarily reserves the offered IP address for the device.
- 3) **Request:** The device further sends a DHCPREQUEST message to show its acceptance of the IP address given. This message, at the same time, notifies other servers that their offers shall not be considered to avoid conflicts.

4.) Acknowledge: Finally, the DHCP server sends a DHCPACK message to confirm the lease. By this message, the process is complete, and the device is now officially assigned an IP address, and thus can function on the network.

If any of these steps fail, the client may attempt to reconnect or temporarily assign itself an Automatic Private IP Address, APIPA, until such time as it successfully obtains a DHCP lease.

Signaling:

It consists of the series of messages exchanged by both client and server to develop and maintain network configurations - a process called DHCP signaling. These messages use User Datagram Protocol (UDP). The server uses port number 67, and for the client, it uses 68.

1) Initial Communication: A client uses broadcast messages, like DHCPDISCOVER and DHCPREQUEST, to reach all available DHCP servers when it first connects to a network. It cannot send tailored messages since the client does not have an IP address.

2) Unicast: After assigning an IP address, subsequent messages, such as DHCPOFFER and DHCPACK, are sent as unicast. This helps in streamlining the communication process since it ensures the configuration parameters are only released to the correct client.

3) Lease Renewal and Rebinding: DHCP leases are valid for a certain amount of time. To extend the lease, the client first tries to contact the original server directly. In case the server is unreachable, the client broadcasts a renewal request on the network to other DHCP servers to maintain connectivity over the network.

4) Fallback Mechanism: The client assigns itself an APIPA address in the range 169.254.0.0/16 if no DHCP server responds, thus allowing limited local communication until such time as a server becomes available.

Because UDP is used, DHCP signaling is lightweight and effective. While the subsequent unicast communication maximizes resource consumption, the initial broadcast ensures that new devices can find DHCP servers without requiring prior configuration. This organized way of signaling is how DHCP can guarantee dependable and flexible network setup.

Commands Used:

Install DHCP server package: `sudo apt update && sudo apt install isc-dhcp-server -y`

Edit dhcp server configuration file for Ipv4: `sudo nano /etc/dhcp/dhcpd.conf`

Edit dhcp server configuration file for Ipv6: `sudo nano /etc/dhcp/dhcpd6.conf`

Configure interfaces for the dhcp server: `sudo nano /etc/default/isc-dhcp-server`

Restart dhcp server: `sudo systemctl restart isc-dhcp-server`

Enable dhcp server on the startup: `sudo systemctl enable isc-dhcp-server`

Check dhcp service status: `sudo systemctl status isc-dhcp-server`

Verify dhcp leases: `cat /var/lib/dhcp/dhcpd.leases`

Allow dhcp traffic through the firewall: `sudo ufw allow 67/udp`

Test dhcp server functionality on client: `sudo dhclient -v eth0`

Testing screenshots:

```
jeeva@dhcp1:~$ sudo systemctl status isc-dhcp-server6
● isc-dhcp-server6.service - ISC DHCP IPv6 server
   Loaded: loaded (/lib/systemd/system/isc-dhcp-server6.service; enabled; vendor
   Active: active (running) since Mon 2024-12-02 01:34:51 EST; 6min ago
     Docs: man:dhcpcd(8)
    Main PID: 2080 (dhcpcd)
      Tasks: 1 (limit: 2318)
    CGroup: /system.slice/isc-dhcp-server6.service
            └─2080 dhcpcd -user dhcpcd -group dhcpcd -f -6 -pf /run/dhcp-server/dhc

Dec 02 01:34:51 dhcp1 dhcpcd[2080]: Sending on Socket/5/enp0s3/fd00::/64
Dec 02 01:34:51 dhcp1 sh[2080]: Sending on Socket/5/enp0s3/fd00::/64
Dec 02 01:34:51 dhcp1 dhcpcd[2080]: Server starting service.
Dec 02 01:37:50 dhcp1 dhcpcd[2080]: Solicit message from fe80::a00:27ff:fe20:f80
Dec 02 01:37:50 dhcp1 dhcpcd[2080]: Picking pool address fd00::50
Dec 02 01:37:50 dhcp1 dhcpcd[2080]: Advertise NA: address fd00::50 to client wit
Dec 02 01:37:50 dhcp1 dhcpcd[2080]: Sending Advertise to fe80::a00:27ff:fe20:f80
Dec 02 01:37:50 dhcp1 dhcpcd[2080]: Request message from fe80::a00:27ff:fe20:f80
Dec 02 01:37:50 dhcp1 dhcpcd[2080]: Reply NA: address fd00::50 to client with du
Dec 02 01:37:50 dhcp1 dhcpcd[2080]: Sending Reply to fe80::a00:27ff:fe20:f80f po
lines 1-19/19 (END)
```

```
jeeva@dhcp1:~$ sudo cat /var/lib/dhcp/dhcpd6.leases
# The format of this file is documented in the dhcpd.leases(5) manual page.
# This lease file was written by isc-dhcp-4.3.5

# authoring-byte-order entry is generated, DO NOT DELETE
authoring-byte-order little-endian;

server-duid "\000\001\000\001.\340\021\213\010\000'E\356)";

ia-na "\017\370 '\000\001\000\001.\336\200\207\010\000' \370\017" {
    cltt 1 2024/12/02 06:37:50;
    iaaddr fd00::50 {
        binding state active;
        preferred-life 604800;
        max-life 2592000;
        ends 3 2025/01/01 06:37:50;
    }
}
```

```
jeevanandh@jeevanandh-VirtualBox:~$ ping6 fd17:625c:f037:2::1
PING fd17:625c:f037:2::1(fd17:625c:f037:2::1) 56 data bytes
64 bytes from fd17:625c:f037:2::1: icmp_seq=1 ttl=255 time=1.32 ms
64 bytes from fd17:625c:f037:2::1: icmp_seq=2 ttl=255 time=0.771 ms
64 bytes from fd17:625c:f037:2::1: icmp_seq=3 ttl=255 time=0.598 ms
64 bytes from fd17:625c:f037:2::1: icmp_seq=4 ttl=255 time=0.710 ms
64 bytes from fd17:625c:f037:2::1: icmp_seq=5 ttl=255 time=0.610 ms
64 bytes from fd17:625c:f037:2::1: icmp_seq=6 ttl=255 time=3.65 ms
^Z
[9]+  Stopped                  ping6 fd17:625c:f037:2::1
jeevanandh@jeevanandh-VirtualBox:~$
```

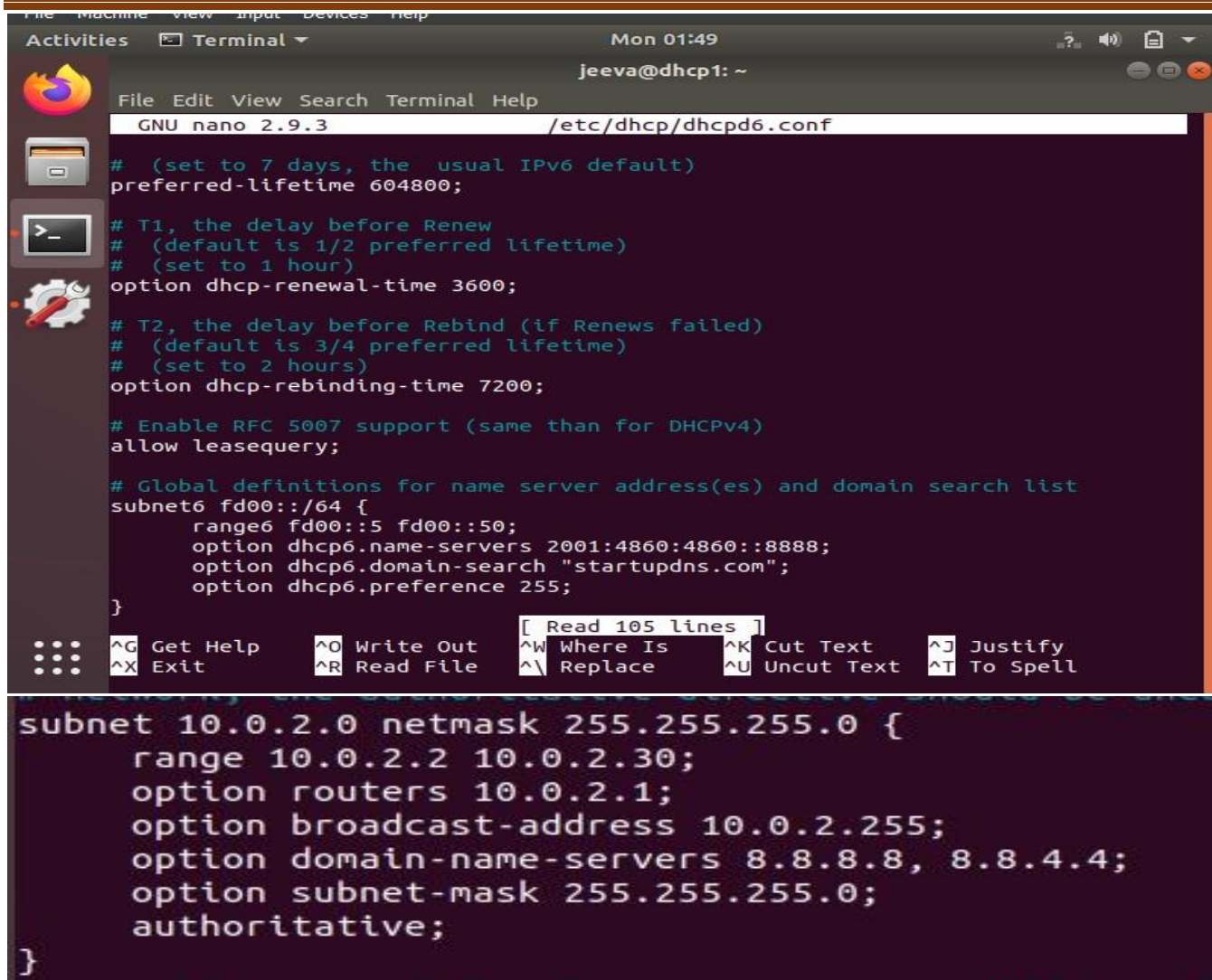


```
jeeva@dhcp1:~$ sudo nano /etc/dhcp/dhcpd6.conf
jeeva@dhcp1:~$ sudo systemctl restart isc-dhcp-server6
jeeva@dhcp1:~$ sudo systemctl status isc-dhcp-server6
● isc-dhcp-server6.service - ISC DHCP IPv6 server
   Loaded: loaded (/lib/systemd/system/isc-dhcp-server6.service; enabled; vendor
   Active: active (running) since Mon 2024-12-02 01:34:51 EST; 2s ago
     Docs: man:dhcpd(8)
   Main PID: 2080 (dhcpd)
    Tasks: 1 (limit: 2318)
   CGroup: /system.slice/isc-dhcp-server6.service
           └─2080 dhcpd -user dhcpd -group dhcpd -f -6 -pf /run/dhcp-server/dhc

Dec 02 01:34:51 dhcp1 sh[2080]: PID file: /run/dhcp-server/dhcpd6.pid
Dec 02 01:34:51 dhcp1 dhcpd[2080]: Wrote 0 NA, 0 TA, 0 PD leases to lease file.
Dec 02 01:34:51 dhcp1 sh[2080]: Wrote 0 NA, 0 TA, 0 PD leases to lease file.
Dec 02 01:34:51 dhcp1 dhcpd[2080]: Bound to *:547
Dec 02 01:34:51 dhcp1 sh[2080]: Bound to *:547
Dec 02 01:34:51 dhcp1 dhcpd[2080]: Listening on Socket/5/enp0s3/fd00::/64
Dec 02 01:34:51 dhcp1 sh[2080]: Listening on Socket/5/enp0s3/fd00::/64
Dec 02 01:34:51 dhcp1 dhcpd[2080]: Sending on Socket/5/enp0s3/fd00::/64
Dec 02 01:34:51 dhcp1 sh[2080]: Sending on Socket/5/enp0s3/fd00::/64
Dec 02 01:34:51 dhcp1 dhcpd[2080]: Server starting service.
lines 1-19/19 (END)
```

```
jeeva@dhcp1:~$ sudo systemctl status isc-dhcp-server
● isc-dhcp-server.service - ISC DHCP IPv4 server
   Loaded: loaded (/lib/systemd/system/isc-dhcp-server.service; enabled; vendor
   Active: active (running) since Sun 2024-12-01 13:16:28 EST; 2min 12s ago
     Docs: man:dhcpd(8)
   Main PID: 2347 (dhcpd)
    Tasks: 1 (limit: 2318)
   CGroup: /system.slice/isc-dhcp-server.service
           └─2347 dhcpd -user dhcpd -group dhcpd -f -4 -pf /run/dhcp-server/dhc

Dec 01 13:17:16 dhcp1 dhcpd[2347]: DHCPOFFER on 10.0.2.5 to 08:00:27:20:f8:0f (
Dec 01 13:18:11 dhcp1 dhcpd[2347]: DHCPDISCOVER from 08:00:27:20:f8:0f (jeevana
Dec 01 13:18:11 dhcp1 dhcpd[2347]: DHCPOFFER on 10.0.2.5 to 08:00:27:20:f8:0f (
Dec 01 13:18:11 dhcp1 dhcpd[2347]: DHCPREQUEST for 10.0.2.5 (10.0.2.3) from 08:
Dec 01 13:18:11 dhcp1 dhcpd[2347]: DHCPACK on 10.0.2.5 to 08:00:27:20:f8:0f (je
Dec 01 13:18:24 dhcp1 dhcpd[2347]: DHCPRELEASE of 10.0.2.5 from 08:00:27:20:f8:
Dec 01 13:18:30 dhcp1 dhcpd[2347]: DHCPDISCOVER from 08:00:27:20:f8:0f via enp0
Dec 01 13:18:30 dhcp1 dhcpd[2347]: DHCPOFFER on 10.0.2.5 to 08:00:27:20:f8:0f (
Dec 01 13:18:30 dhcp1 dhcpd[2347]: DHCPREQUEST for 10.0.2.5 (10.0.2.3) from 08:
Dec 01 13:18:30 dhcp1 dhcpd[2347]: DHCPACK on 10.0.2.5 to 08:00:27:20:f8:0f (je
lines 1-19/19 (END)
```



The screenshot shows a terminal window with the nano text editor open, editing the file /etc/dhcp/dhcpd6.conf. The editor's status bar at the top indicates 'GNU nano 2.9.3' and the file path. The configuration file content is as follows:

```
# (set to 7 days, the usual IPv6 default)
preferred-lifetime 604800;

# T1, the delay before Renew
# (default is 1/2 preferred lifetime)
# (set to 1 hour)
option dhcp-renewal-time 3600;

# T2, the delay before Rebind (if Renews failed)
# (default is 3/4 preferred lifetime)
# (set to 2 hours)
option dhcp-rebinding-time 7200;

# Enable RFC 5007 support (same than for DHCPv4)
allow leasequery;

# Global definitions for name server address(es) and domain search list
subnet6 fd00::/64 {
    range6 fd00::5 fd00::50;
    option dhcp6.name-servers 2001:4860:4860::8888;
    option dhcp6.domain-search "startupdns.com";
    option dhcp6.preference 255;
}

subnet 10.0.2.0 netmask 255.255.255.0 {
    range 10.0.2.2 10.0.2.30;
    option routers 10.0.2.1;
    option broadcast-address 10.0.2.255;
    option domain-name-servers 8.8.8.8, 8.8.4.4;
    option subnet-mask 255.255.255.0;
    authoritative;
}
```

The bottom of the terminal window shows a status bar with various keyboard shortcuts: ^G Get Help, ^O Write Out, ^R Read File, ^W Where Is, ^K Cut Text, ^J Justify, ^X Exit, ^_ Replace, ^U Uncut Text, and ^T To Spell. A message '[Read 105 lines]' is also visible.

b) Domain Name System (DNS):

DNS is an essential network function that makes finding and connecting to other devices over the internet or private networks easier; it turns readable, human-friendly domain names, such as www.startup.com, into IP addresses, such as 192.168.2.1. While it may be easy for a human mind to remember the previous name, the latter address in this case is what the computer and networking device use to know and identify each other; thus, this translation must take place. DNS acts much like a distributed, hierarchical database. At the top of the hierarchy are root servers, TLD servers, such as www.com and www.org, and authoritative name servers responsible for specific domains. A client requests that DNS servers resolve a domain name by first querying the closest cache resolver and iteratively working its way through the hierarchy if necessary. Despite the immense scale of the internet, the technology ensures effective resolution and scalability.

Protocol Behavior:

This whole process is based upon the DNS protocol answering queries, responding with answers. It uses recursive, iterative, and non-recursive DNS queries.

- 1) Recursive Query: In this type of query, the DNS client requests a recursive response from the DNS resolver. The resolver continues to query other DNS servers until it retrieves the correct IP address and returns it to the client if it does not have the requested information cached.
- 2) Iterative Query: Here, the DNS resolver will return to the client with an address of a different DNS server that is closer to where the answer is; there, the client makes a query again to reach out to the DNS it has been referred to until the IP address is retrieved.
- 3) Caching and Optimization: DNS resolvers temporarily cache answers from previously made queries to save query time and network usage for recurring queries. The data will be held in the cache if the time-to-live number in the DNS records says.

This, in turn, guarantees speed; hence, for standard communications, UDP is used for DNS queries and answers at port 53. DNS may then switch to TCP in such transactions that have to be reliable or in case the answers are bigger. Such a dual-protocol approach ensures performance and dependability in a range of application situations.

Signaling:

During name resolution, DNS clients (resolvers) exchange messages with servers in a process called DNS signaling. The following is included within the signaling process:

- 1) Query Initiation: A client, to visit a website or resource, sends a DNS query to the DNS resolver configured on it. The type of record needed, such as A for the IPv4 address, AAAA for the IPv6 address, or MX for the mail server, along with the domain name that must be resolved, is included in this query.
- 2) Resolver Recursion: The resolver sends the result directly to the client if it has any of the requested data cached. If not, it initiates a series of queries going from root servers to TLD servers to authoritative servers, each moving closer to returning the correct answer.
- 3) Response Delivery: The client can access the target resource once the resolver, after getting the IP address, sends the response back to them.
- 4) Error Handling: It returns an error code from the resolver, such as NXDOMAIN for a Non-Existent Domain, in case the requested domain is invalid or can't be resolved. This allows the client to handle invalid or unreachable domains gracefully.

Because DNS signaling uses a hierarchical query mechanism and cache, it is effective. DNS minimizes latency, reduces network traffic by caching frequently requested domain resolves, and directs clients to authoritative sources. For the different types of requests, a combination of UDP and TCP ensures a balance between speed and reliability. In fact, this controlled signaling process forms the backbone of the smooth functioning of modern internet and network services.

Commands Used:

Install bind9: `sudo apt update && sudo apt install bind9 -y`

Edit main bind9 configuration file: `sudo nano /etc/bind/named.conf.options`

Create zones in the local configurations: `sudo nano /etc/bind/named.conf.local`

Create the forward zone file: `sudo nano /etc/bind/zones/db.yourdomain.com`

Create the reverse zone file: `sudo nano /etc/bind/zones/db.10.0.2`

Check bind9 configurations for errors: `sudo named-checkconf`

Restart bind9 services: `sudo systemctl restart bind9`

Check status for the bind9 service: `sudo systemctl status bind9`

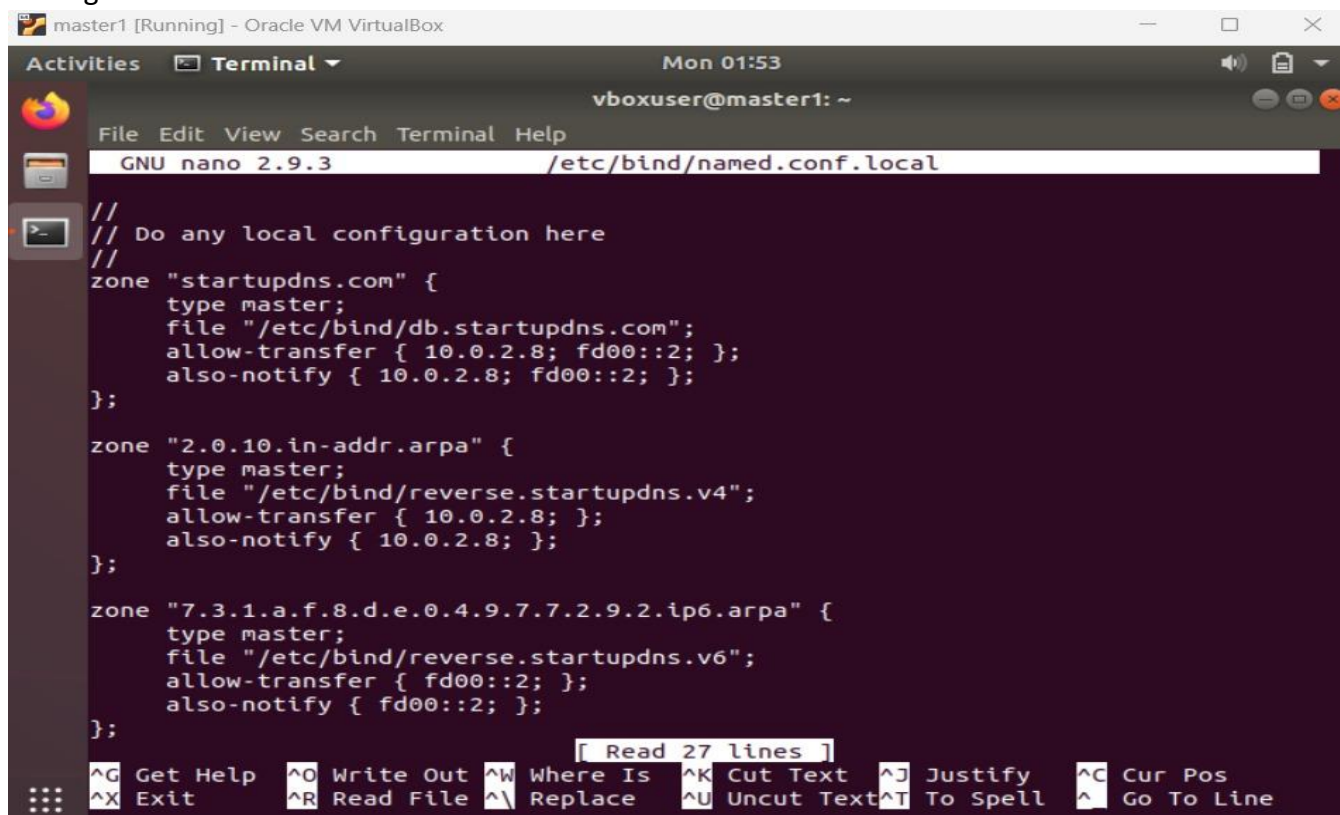
Test forward DNS resolution: `dig @127.0.0.1 startupdns.com`

Test reverse DNS resolution: `dig -x 192.168.1.9 @127.0.0.1`

Monitor DNS logs: `sudo tail -f /var/log/bind/default.log`

Allow DNS traffic through the firewall: `sudo ufw allow 53/tcp && sudo ufw allow 53/udp`

Testing screenshots:



The screenshot shows a terminal window titled "master1 [Running] - Oracle VM VirtualBox". The window contains a nano editor editing the file `/etc/bind/named.conf.local`. The editor shows the following configuration:

```
//  
// Do any local configuration here  
//  
zone "startupdns.com" {  
    type master;  
    file "/etc/bind/db.startupdns.com";  
    allow-transfer { 10.0.2.8; fd00::2; };  
    also-notify { 10.0.2.8; fd00::2; };  
};  
  
zone "2.0.10.in-addr.arpa" {  
    type master;  
    file "/etc/bind/reverse.startupdns.v4";  
    allow-transfer { 10.0.2.8; };  
    also-notify { 10.0.2.8; };  
};  
  
zone "7.3.1.a.f.8.d.e.0.4.9.7.7.2.9.2.ip6.arpa" {  
    type master;  
    file "/etc/bind/reverse.startupdns.v6";  
    allow-transfer { fd00::2; };  
    also-notify { fd00::2; };  
};
```

At the bottom of the terminal, there is a status bar showing "Read 27 lines" and a list of keyboard shortcuts: `^G` Get Help, `^O` Write Out, `^W` Where Is, `^K` Cut Text, `^J` Justify, `^C` Cur Pos, `^X` Exit, `^R` Read File, `^_` Replace, `^U` Uncut Text, `^T` To Spell, `^` Go To Line.

```
GNU nano 2.9.3 /etc/bind/db.startupdns.com

;
; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA      localhost. root.localhost. (
        2024120101      ; Serial
        604800          ; Refresh
        86400           ; Retry
        2419200         ; Expire
        604800 )        ; Negative Cache TTL
;
@         IN      NS       ns1.startupdns.com.
@         IN      NS       ns2.startupdns.com.
;
ns1       IN      A        10.0.2.7
ns1       IN      AAAA     fd00::1
;
ns2       IN      A        10.0.2.8
ns2       IN      AAAA     fd00::2
;
www       IN      A        10.0.2.9
www       IN      AAAA     fd00::3

[ Read 22 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

```
vboxuser@master1:~$ sudo systemctl status bind9
[sudo] password for vboxuser:
● bind9.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/bind9.service; enabled; vendor preset: ena
   Active: active (running) since Sun 2024-12-01 21:56:39 EST; 1min 1s ago
     Docs: man:named(8)
   Main PID: 1241 (named)
    Tasks: 4 (limit: 2318)
   CGroup: /system.slice/bind9.service
           └─1241 /usr/sbin/named -f -u bind

Dec 01 21:56:39 master1 named[1241]: zone 255.in-addr.arpa/IN: loaded serial 1
Dec 01 21:56:39 master1 named[1241]: zone 7.3.1.a.f.8.d.e.0.4.9.7.7.2.9.2.ip6.ar
Dec 01 21:56:39 master1 named[1241]: zone localhost/IN: loaded serial 2
Dec 01 21:56:39 master1 named[1241]: zone startupdns.com/IN: loaded serial 202412
Dec 01 21:56:39 master1 named[1241]: all zones loaded
Dec 01 21:56:39 master1 named[1241]: running
Dec 01 21:56:39 master1 named[1241]: zone 2.0.10.in-addr.arpa/IN: sending notifie
Dec 01 21:56:39 master1 named[1241]: zone startupdns.com/IN: sending notifies (se
Dec 01 21:56:39 master1 named[1241]: zone 7.3.1.a.f.8.d.e.0.4.9.7.7.2.9.2.ip6.ar
Dec 01 21:57:11 master1 named[1241]: client @0x7f4eb40c7310 10.0.2.8#47754: recei

[1]+  Stopped                  sudo systemctl status bind9
vboxuser@master1:~$
```

```
;; Connection timed out; no servers could be reached
jeevanandh@jeevanandh-VirtualBox:~$ dig @10.0.2.8 www.startupdns.com

; <<>> DiG 9.11.3-1ubuntu1.18-Ubuntu <<>> @10.0.2.8 www.startupdns.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31222
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: c535a2b5833b3b91834cb69e674cf8a4d8547418c9dc6387 (good)
;; QUESTION SECTION:
;www.startupdns.com.                IN      A

;; ANSWER SECTION:
www.startupdns.com.                604800  IN      A      10.0.2.9

;; AUTHORITY SECTION:
startupdns.com.                    604800  IN      NS      ns1.startupdns.com.
startupdns.com.                    604800  IN      NS      ns2.startupdns.com.

;; ADDITIONAL SECTION:
ns1.startupdns.com.                604800  IN      A      10.0.2.7
ns2.startupdns.com.                604800  IN      A      10.0.2.8
ns1.startupdns.com.                604800  IN      AAAA   fd00::1
ns2.startupdns.com.                604800  IN      AAAA   fd00::2

;; Query time: 4 msec
;; SERVER: 10.0.2.8#53(10.0.2.8)
;; WHEN: Sun Dec 01 19:00:36 EST 2024
;; MSG SIZE rcvd: 215
```

c) Web Server:

The web server is a hardware or software device that responds to the client's request, usually a web browser, and delivers images, web pages, and other files over the intranet or the internet. It serves as an intermediary between users and the backend resources, ensuring that the requested material is effectively delivered. The primary function of a web server is to handle HTTP or HTTPS requests and return the requested content—usually an HTML document, CSS files, JavaScript, or some form of multimedia. Web servers can run both static and dynamic content. Content that is stored in files on the server and served directly to the requesting user, such as HTML pages or images, are called static content. Dynamic content, on the other hand, is generated on the fly in response to user input or back-end activities. Applications that require database access, like social media platforms or e-commerce websites, do this quite often. Microsoft IIS, Nginx, and Apache HTTP Server are the most popular web server programs tailored for particular use cases and performance requirements.

Protocol Behavior:

The Hypertext Transfer Protocol, or HTTP, and its secure version, HTTPS, define the way a web server should behave. A client, usually a browser or an application, interacts with a web server in the following steps:

- 1) Handling the Request: The client requests a resource from the web server through an HTTP/HTTPS request. It includes, but is not limited to, headers, the method such as GET or POST, and sometimes data, like form inputs.
 - 2) Resource Lookup: The web server, after analyzing the request, finds the resource that was asked for. If this is a static file, the server retrieves it directly. In the case of dynamic content, the server may interact with databases or backend applications to generate a response.
 - 3) Response Generation: After processing the request, the web server generates an HTTP response. This consists of the requested content or an error message, headers, and the status code, for example, 200 OK, 404 Not Found.
 - 4) Session Management: In dynamic applications, the server may track user sessions for a consistent and customized experience for the users' using cookies, session tokens, or other means.
- The reason being modern web servers are designed to optimize performance by using features such as compression, load balancing, and caching. These reduce the overall load on the server and network and ensure faster response times, especially during high-traffic periods.

Signaling:

Signaling on a web server is generally done via a series of communications between the client and server using the HTTP/HTTPS protocol. The process ensures that resources are provided in due time and that data exchange occurs in an effective manner.

- 1) Establishment of Connection: In the case of HTTP, the client connects to the server, usually via port 80 (for HTTP) or port 443 (for HTTPS). To guarantee secure communication, a TLS/SSL handshake is a part of the HTTPS process.
 - 2) Request Dispatch: The client sends an HTTP request to the server, which includes the resource required-GET /index.html, for instance-and other information, such as headers or POST payload.
 - 3) Server Processing: This is the processing at the server side, which may include database queries, execution of scripts-like PHP, Python, or any Java-based applications, or reading static files. It generates the proper answer with metadata, such as encoding and content type.
 - 4) Response Delivery: The client receives the HTTP response from the server. This response, including the appropriate HTTP status codes to indicate the outcome-such as 200 for success or 500 for server failures-includes the requested content or an error message.
 - 5) Connection Closure: After the response is sent, the connection may either be closed or kept open for more requests, depending on the version of the protocol and configuration of the server.
- Advanced capabilities include support for WebSocket connections that allow real-time communication, the handling of asynchronous requests, and content negotiation to provide resources in the best format for the client. Web servers ensure reliable and efficient delivery of web information to consumers by following the HTTP protocol and leveraging contemporary optimization strategies.

Commands used:

Install Apache webserver: `sudo apt update && sudo apt install apache2 -y`
Start Apache service: `sudo systemctl start apache2`

Enable Apache to start on boot: `sudo systemctl enable apache2`

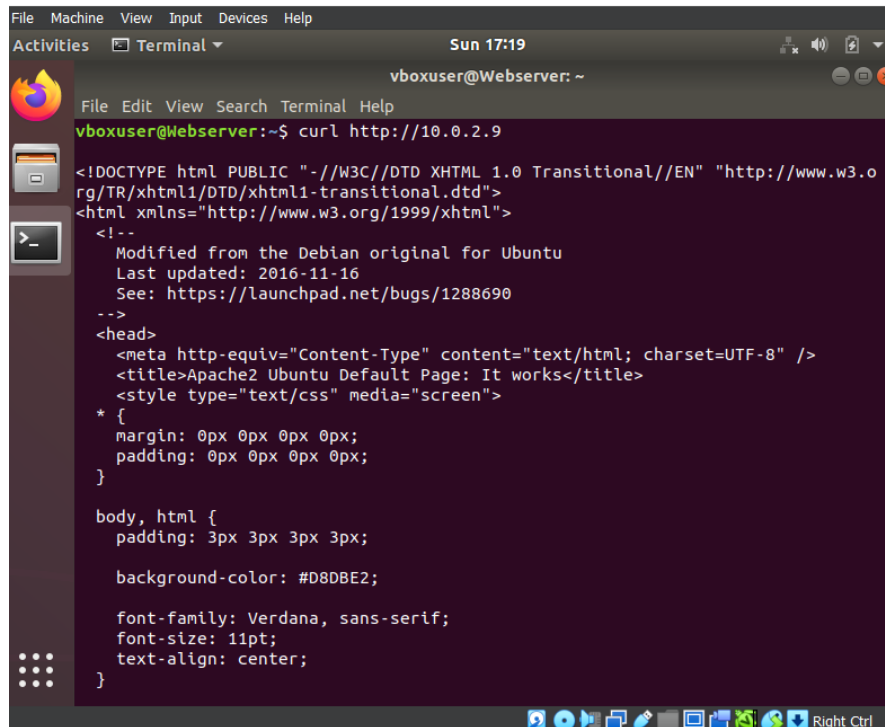
Check status of Apache: `sudo systemctl status apache2`

Test apache configurations for syntax errors: `sudo apache2ctl configtest`

Allow HTTP and HTTPS traffic through firewall: `sudo ufw allow 'Apache Full'`

Installed Fail2Ban to make the server more secure:

Testing screenshots:



```
File Edit View Search Terminal Help
Sun 17:19
vboxuser@Webserver: ~
vboxuser@Webserver:~$ curl http://10.0.2.9
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
  Modified from the Debian original for Ubuntu
  Last updated: 2016-11-16
  See: https://launchpad.net/bugs/1288690
-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Apache2 Ubuntu Default Page: It works</title>
<style type="text/css" media="screen">
* {
  margin: 0px 0px 0px 0px;
  padding: 0px 0px 0px 0px;
}

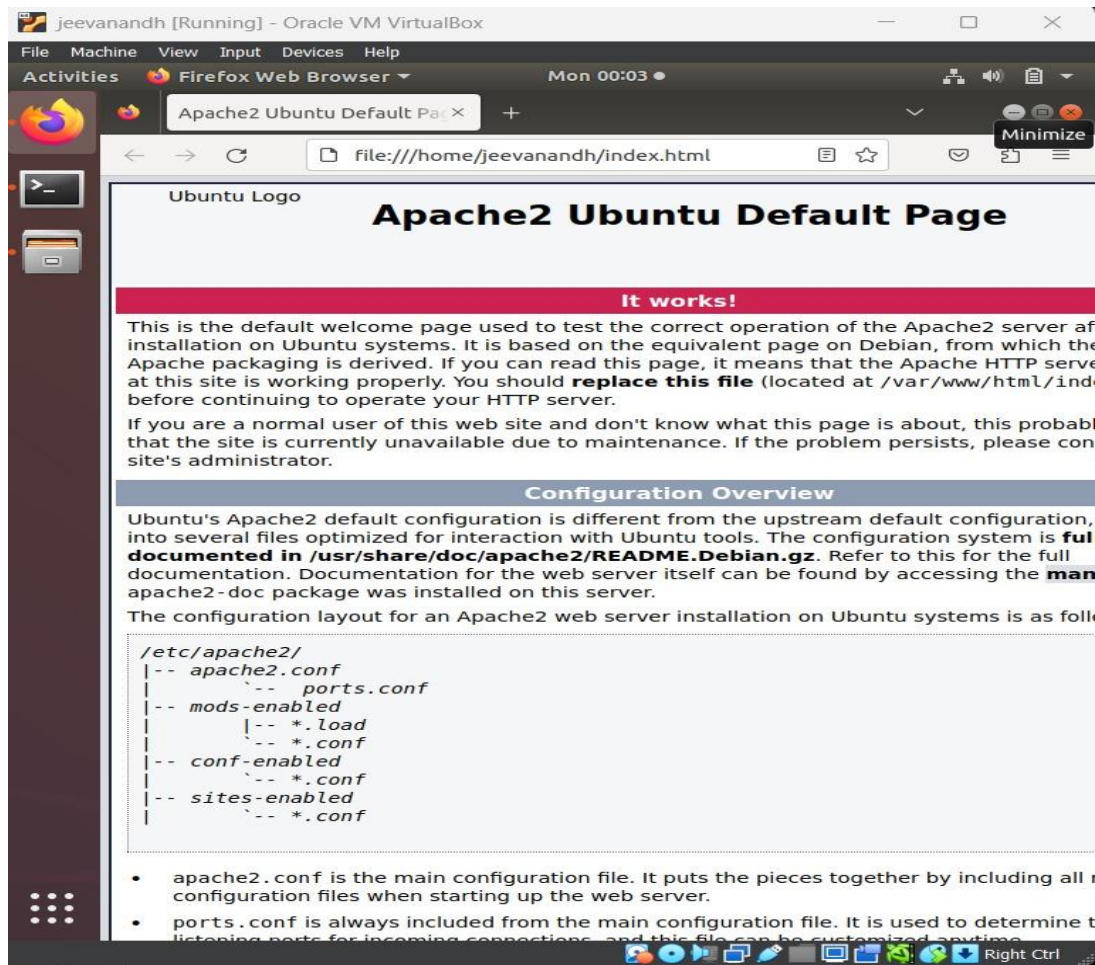
body, html {
  padding: 3px 3px 3px 3px;

  background-color: #D8DBE2;

  font-family: Verdana, sans-serif;
  font-size: 11pt;
  text-align: center;
}
```

```
vboxuser@Webserver:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset:
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Sun 2024-12-01 16:02:33 EST; 6s ago
     Process: 2195 ExecStop=/usr/sbin/apachectl stop (code=exited, status=0/SUCCESS)
     Process: 2200 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
    Main PID: 2204 (apache2)
       Tasks: 55 (limit: 2318)
     CGroup: /system.slice/apache2.service
             └─2204 /usr/sbin/apache2 -k start
               2205 /usr/sbin/apache2 -k start
               2206 /usr/sbin/apache2 -k start

Dec 01 16:02:33 Webserver systemd[1]: Stopped The Apache HTTP Server.
Dec 01 16:02:33 Webserver systemd[1]: Starting The Apache HTTP Server...
Dec 01 16:02:33 Webserver systemd[1]: Started The Apache HTTP Server.
lines 1-17/17 (END)
```



d) Firewall:

A firewall is a software or hardware network security tool that monitors and controls all incoming and outgoing traffic based on predetermined security rules. Firewalls become very important in systems' defense mechanisms against malicious activities, cyber-attacks, and unauthorized access by interposing a barrier between trusted internal networks and probably untrusted external networks, such as the internet. They ensure that only secure and valid packets of data are allowed to pass through, maintaining the integrity and security of the network. Some examples of the various layers at which firewalls can function include network-layer packet filtering and application-layer traffic inspection. Firewalls can also come in different forms: cloud-based, software-based, and hardware-based. Modern firewalls often include advanced features such as VPN compatibility, intrusion detection and prevention systems, and deep packet inspection for added security.

Protocol Behavior:

Network administrators set up a set of rules that define how a firewall behaves. These rules define how the firewall will handle data packets and ensure that legitimate communication continues uninterrupted while undesired traffic is not allowed. The basic actions include:

- 1) Packet filtering: Firewalls analyze the data packets at the network layer to allow or reject them individually. Accordingly, a decision to admit or reject is made based on parameters such as source and destination IP addresses, port, and protocols. For instance, a firewall can deny all traffic on a certain port that is known commonly to be associated with any kind of vulnerability, which could be port 23 (Telnet).
- 2) Stateful Inspection: Unlike the simple packet filtering mechanism, stateful firewalls keep a record of the open connections. Therefore, this makes it easier to identify packets that are part of an established session and ensure that only expected responses are allowed through. If a client initiates a web request, for example, the firewall will only allow the response of the server and block packets from sources that were not requested.
- 3) Application Layer Filtering: Firewalls working at the application layer can inspect the contents of data packets to identify and block potentially dangerous activity. For instance, they can block users from downloading dangerous files or accessing unauthorized websites.
- 4) Monitoring and Logging: Firewalls log traffic information, which provides administrators with valuable data regarding attempted intrusions or other anomalies. These logs help in identifying threats and adjusting firewall rules to provide better security.

Signaling:

The signaling of the firewall refers to the way in which the firewall responds to data packets as they go across the network. While possible threats are caught, legal communication is smoothly passed thanks to this signaling procedure. The process of signaling involves a number of crucial phases:

- 1) Packet Inspection: Upon arrival, the packet is inspected by the firewall with a set of predefined rules. The source and destination IP addresses, ports, and protocols of a packet can be checked within the headers. For example, packets from an IP range already known to be malicious get deleted immediately.
- 2) Decision Making: Based on the rule set, the firewall decides whether to allow, block, or redirect the packet. In the case of stateful firewalls, this involves determining where in an active session the packet falls.
- 3) Execution of Action: The firewall can decide to let the packet flow, completely drop it, or log it for further examination. For instance, packets may be lost and marked in the logs as part of a DDoS attack.
- 4) Dynamic Adaptation: Modern firewalls can react to an attack as and when it happens. Suppose a certain IP address shows dubious activity; the very instance it is detected, the firewall automatically blocks that IP address for some time to reduce the intrusion chance.

The Firewall Signaling ensures the network is secure and doesn't tamper with normal communications. Firewalls have become able to constantly update their rules, advanced capability integrations like threat intelligence make them up to date with ever-evolving cyber threats. Thus, a firewall should

marry the right rule application along with complete inspection so as to preserve this delicate security and performance balance in any given network.

Testing screenshots:

```
Rule added (v6)
vboxuser@Webserver:~$ sudo ufw status
Status: active

To                Action            From
--                -
80/tcp            ALLOW             Anywhere
443/tcp           ALLOW             Anywhere
Apache Full       ALLOW             Anywhere
22/tcp            LIMIT             Anywhere
2202/tcp          ALLOW             Anywhere
80/tcp (v6)       ALLOW             Anywhere (v6)
443/tcp (v6)      ALLOW             Anywhere (v6)
Apache Full (v6)  ALLOW             Anywhere (v6)
22/tcp (v6)       LIMIT             Anywhere (v6)
2202/tcp (v6)     ALLOW             Anywhere (v6)
```

```
vboxuser@Webserver:~$ sudo ufw status
Status: active

To                Action            From
--                -
80/tcp            ALLOW             Anywhere
443/tcp           ALLOW             Anywhere
Apache Full       ALLOW             Anywhere
80/tcp (v6)       ALLOW             Anywhere (v6)
443/tcp (v6)      ALLOW             Anywhere (v6)
Apache Full (v6)  ALLOW             Anywhere (v6)

vboxuser@Webserver:~$
```

e) Backup:

A backup system is a crucial part of data management that ensures the recovery and preservation of important data in case of loss, corruption, or disaster. Backups provide a safety net for people and businesses against natural disasters, cyberattacks, hardware failures, and inadvertent deletions by making duplicate copies of the data and keeping them in safe places. Backups are implemented in many ways: differential, incremental, and complete. Whereas an incremental backup stores only the changes made since the last backup, in a full backup, which makes a complete copy of all data, less storage is required. Differential backups, as their name suggests, capture a balance between speed and completeness by storing changes since the last full backup. Automation, encryption, and cloud storage are often integrated into modern backup solutions to enhance accessibility, security, and reliability.

Protocol Behavior:

The behavior of a backup system focuses on the efficiency of the data recovery, storage, and copying processes. Backup procedures ensure minimum consumption of resources while ensuring accessibility and integrity of the data. Key aspects of the behavior include:

- 1) Data Selection: Backup systems use user-specified parameters, including file types, directories, or servers, to determine which data should be backed up. This enables businesses to filter out superfluous files and prioritize important data.
- 2) Scheduling and Automation: To ensure that newer changes are recorded automatically, backups are often scheduled at periodic intervals. Automating this reduces the possibility of human error and thus ensures continuous protection of valued data.
- 3) Data Compression and Encryption: Backup systems first compress the data before storing it; this has the effect of full utilization of storage. Encryption, even on storage or in transit, ensures backed-up data is secure, safe, and inaccessible to unauthorized individuals.
- 4) Verification and Integrity Checks: The system verifies that the data in the backup matches the source. This step ensures the backup is accurate and complete, ready for use in case it becomes necessary.
- 5) Storage and Redundancy: The backups are made to be kept in different locations to minimize the likelihood of data loss. It could also mean cloud platforms, offsite servers, or storage devices on-site. Redundancy simply means making sure that at any moment there will be an available backup copy should a single location where the backup was kept fails.

Signaling:

The process of coordination and communication between the backup system and the source system for data recovery and transfer is what is referred to as backup signaling. This ensures smooth and efficient running of the backup processes. The processes involved in signaling include:

- 1) Initialization: After the establishment of a connection with the source system, the backup system confirms the identity and authorization of the users. This ensures that the backup procedure is authorized and secure.
- 2) Data Transfer: The system transfers data from source to the backup location based on the chosen backup technique, whether it is full, incremental, or differential. Advanced backup systems speed up and make this process more efficient by transferring only modified areas of huge files using techniques like block-level copying.
- 3) Real-time Monitoring: The system monitors the transfer during a backup procedure to detect and resolve problems such as file access issues or network failures. To provide an audit trail and assist in troubleshooting, logs are generated for every operation.
- 4) Completion and Reporting: The system provides a comprehensive report of the backup operation and a confirmation signal to the source system after the completion of the backup. This includes information about the data that was backed up, any problems encountered, and the location of the backup storage.
- 5) Restorative Signaling: The backup system sends the requested data back to the source system after retrieval from the backup storage. Verification procedures ensure that the data being restored is complete and intact.

Backup signaling ensures the efficiency, reliability, and safety of the backup process. Most modern systems feature real-time notifications and alerts for events to help administrators monitor activities and quickly identify issues. By providing clear-cut communication, backup signaling gives the source and backup systems a reliable framework for data protection and recovery.

Testing screenshots:

```
jeevsbackup@Jeevsbackup:~$ cd /backup/html
jeevsbackup@Jeevsbackup:/backup/html$ ls
index.html
```

```
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset:
   Active: active (running) since Sun 2024-12-01 23:32:53 EST; 9h ago
   Main PID: 4754 (vsftpd)
     Tasks: 1 (limit: 2318)
    CGroup: /system.slice/vsftpd.service
            └─4754 /usr/sbin/vsftpd /etc/vsftpd.conf

Dec 01 23:32:53 Jeevsbackup systemd[1]: Starting vsftpd FTP server...
Dec 01 23:32:53 Jeevsbackup systemd[1]: Started vsftpd FTP server.
lines 1-10/10 (END)
[1]+  Stopped                  sudo systemctl status vsftpd
^[[A^[[A
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:cf:ec:d4 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.10/24 scope global enp0s3
        valid_lft forever preferred_lft forever
jeevsbackup@Jeevsbackup:~$ cd /backup/html
jeevsbackup@Jeevsbackup:/backup/html$ ls
index.html
jeevsbackup@Jeevsbackup:/backup/html$
```

```
jeevsbackup@Jeevsbackup:~$ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset:
   Active: active (running) since Sun 2024-12-01 23:16:48 EST; 4s ago
     Process: 4652 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited,
   Main PID: 4653 (vsftpd)
        Tasks: 1 (limit: 2318)
       CGroup: /system.slice/vsftpd.service
               └─4653 /usr/sbin/vsftpd /etc/vsftpd.conf

Dec 01 23:16:48 Jeevsbackup systemd[1]: Starting vsftpd FTP server...
Dec 01 23:16:48 Jeevsbackup systemd[1]: Started vsftpd FTP server.
lines 1-11/11 (END)
[19]+  Stopped                  sudo systemctl status vsftpd

jeevsbackup@Jeevsbackup:~$
```

```
jeevanandh@jeevanandh-VirtualBox:~$ ftp 10.0.2.10
Connected to 10.0.2.10.
220 (vsFTPd 3.0.3)
Name (10.0.2.10:jeevanandh): jeevsbackup
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd html
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rwxr-xr-x  1 1000  1000  10918 Nov 30 18:14 index.html
226 Directory send OK.
ftp> get index.html
local: index.html remote: index.html
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for index.html (10918 bytes).
226 Transfer complete.
10918 bytes received in 0.00 secs (5.4975 MB/s)
ftp>
```

Brief Introduction about topics:

Linux File System:

The Linux file system can arrange and structure data on storage devices. It uses a hierarchical directory tree, with the top directory called the root directory (/) that further branches down to subdirectories such as /home, /etc, and /var. The system is very organized and effective in handling enormous data since each directory can have a file and other directories inside it. Some of the file system types that are available in Linux include XFS, Btrfs, and ext4-all with different advantages, such as efficient handling of metadata and journaling for data security. Basically, the role of a file system is to guarantee the confidentiality and integrity of data and permission-based access control.

Tar:

Tar, or Tape Archive, is a Linux utility that can combine multiple files or directories into a single archive file. The main idea behind Tar is to gather files together while preserving their metadata, permissions, and directory structure. It is often used to make the process of backup, storage, and transfer of multiple files easier as a single, coherent entity. Although tar does not compress the files by default, it is often combined with a compression tool like gzip or bzip2 to yield a compressed archive. Combining tar with these methods makes it a powerful option for handling large datasets and preparing files for transport.

Gunzip:

Gunzip is a program that allows Linux users to uncompress files compressed using the gzip algorithm. Its primary function is to restore .gz compressed files to their original state. Gunzip works by reversing the compression applied by gzip, thus restoring the uncompressed accessibility of the files. This program is commonly used together with tar when dealing with compressed archive files to assist in extracting and restoring the contents of .tar.gz archives. Gunzip is essential for users who need to decompress files for use after they have been transferred or saved in a compressed format.

Memory:

The memory management of Linux is one of the essential parts of the operating system, which ensures the effective use of virtual and physical memory. Several techniques are employed by the system for maximum efficiency, judicious distribution of resources, and preservation of stability. Linux divides the memory into kernel space, reserved for the essential system operations, and user space, where the applications run. It utilizes virtual memory to allow applications to keep running smoothly during periods of high demand on the memory by allowing processes to utilize more memory than physically available through swapping data out to disk as necessary. Another important feature of Linux is Paging, a form of memory management where it divides the memory into blocks for better dynamic allocation and utilization. The system also utilizes caching to retain commonly requested data within the memory. This reduces visits to slower storage media, hence making operations faster. The operating system also controls memory in discrete zones such as low memory and high memory to maximize performance for the different memory types. This helps the administrator make informed decisions on memory health through monitoring memory utilization and system performance using tools such as free, top, and vmstat.

Processes:

In Linux, a process is an executing form of a program, and it includes the program code, the current activity, and all the called resources. Each process is allocated a Process ID for tracking and management purposes. Under Linux, several processes can run concurrently thanks to its multitasking capability, each of which executes in its separate, dedicated memory space for mutual non-interference. Processes are divided into two kinds: background processes, which execute a task without

user intervention, and foreground processes, which interact with the user directly. The operating system controls the implementation of these activities through a process called scheduling, ensuring that CPU time is divided fairly. Due to their vital information about memory and CPU usage, utilities such as ps, top, and htop are often utilized in process monitoring and control. Linux also allows the user to control the processes by using commands such as kill to terminate a process and good to change its priority. Additionally, Linux allows the creation of complicated process hierarchies and lightweight multitasking by supporting process management features like exec, which changes the current process image, and forking, which creates new processes.

Bonus Part:

1) Man in the Middle attack:

- A "Man-in-the-Middle" attack, in networking terms, is an active eavesdropping attack wherein an attacker intercepts, relays, and possibly alters communication between two unsuspecting parties. This approach uses Scapy, a powerful Python packet manipulation tool, to simulate an MITM attack in a controlled environment. In this case, the principal tactic used is ARP spoofing, whereby the attacker sends fake ARP responses onto the network to which other devices are connected. These responses fundamentally redirect network traffic through the attacker's system by associating the attacker's MAC address with the IP address of a target device.
- The first step in operation is ARP poisoning, whereby the attacker sends the target devices spoof ARP answers to make them believe that the attacker's computer is the genuine gateway or intended communication partner. After this redirection is affected, the attacker is in a position to intercept and change data packets moving between the devices. These captured packets can be sniffed, modified, or forwarded with Scapy, all while maintaining the appearance of normal communication.
- The attacker's system acts like a relay to ensure that the traffic is forwarded smoothly, while allowing legitimate communication to take place but may alter the private data. This is a very clear example of risks involved in open networks and insufficient cryptographic measures. ARP tables are restored to their original state after the simulation so that normal activities can be carried out.

```
c456@c456:~$ arp -a
? (10.0.2.11) at 08:00:27:e2:aa:e2 [ether] on enp0s3
_gateway (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
c456@c456:~$
```

```
c123@c123:~/Documents$ sudo python3 mitm_arp_poison.py 10.0.2.12 10.0.2.1
[sudo] password for c123:
ERROR: Loading module scapy.layers.ipsec
Traceback (most recent call last):
  File "/home/c123/.local/lib/python3.6/site-packages/scapy/main.py", line 156,
in _load
    mod = importlib.import_module(module)
  File "/usr/lib/python3.6/importlib/__init__.py", line 126, in import_module
    return _bootstrap.gcd_import(name[level:], package, level)
  File "<frozen importlib._bootstrap>", line 994, in _gcd_import
  File "<frozen importlib._bootstrap>", line 971, in _find_and_load
  File "<frozen importlib._bootstrap>", line 955, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 665, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 678, in exec_module
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "/home/c123/.local/lib/python3.6/site-packages/scapy/layers/ipsec.py", l
ine 519, in <module>
    from cryptography.utils import CryptographyDeprecationWarning
ImportError: cannot import name 'CryptographyDeprecationWarning'
[*] Victim IP: 10.0.2.12, Victim MAC: 08:00:27:ac:7f:54
[*] Gateway IP: 10.0.2.1, Gateway MAC: 52:54:00:12:35:00
[*] Starting ARP Poisoning...
```

```
jeevanandh@jeevanandh-VirtualBox:/$ arp -a
_gateway (10.0.2.1) at 08:00:27:e2:aa:e2 [ether] on enp0s3
? (10.0.2.7) at 08:00:27:25:b3:01 [ether] on enp0s3
? (10.0.2.11) at 08:00:27:e2:aa:e2 [ether] on enp0s3
jeevanandh@jeevanandh-VirtualBox:/$
```

2) IPsec VPN Tunneling:

With IPsec, one creates an online secure VPN tunnel between two Linux systems over the internet, for both protected flow of data and protection of its access. The two systems now become peers to each other-i.e., they trust each other and agree upon a mode of encryption and decryption for all traffic. In developing this trust, each machine generates its own private key and a digital certificate signed by a shared Certificate Authority. These certificates will help to identify the machines, somewhat like an ID card identifies a person.

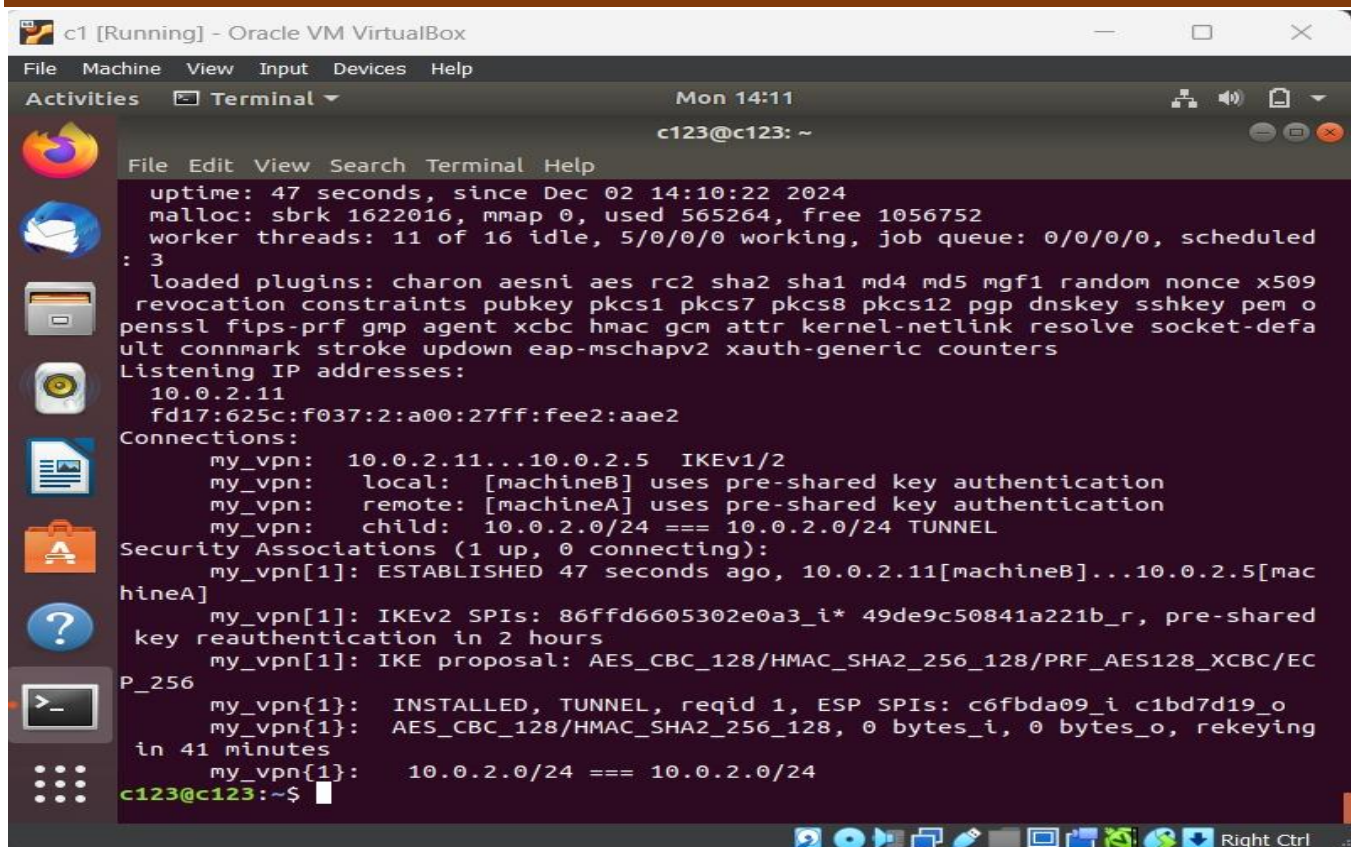
When the VPN is activated, the two machines exchange their certificates and keys in a process called the IKE handshake. During this, they negotiate the encryption methods and mutually agree on how to secure the communication. Once the handshake has been successful, a secure tunnel is established, and the data can pass between both machines in private. Whatever data is sent through the tunnel is encrypted on the sending machine and decrypted only by the receiving machine, keeping it confidential even when intercepted.

In this regard, the IPsec configuration tells it which IPs to use and defines the certificates for authentication. On each machine, we configure a left side-the machine itself-and a right side-the other machine-specifying their respective IP addresses and certificates. When deployed, the VPN ensures

that traffic between these two machines appears as if it is traveling directly between them over a private network, even though it is transmitted over the public internet.

This is a very secure method that fits best for sensitive data that needs to travel between remote systems, such as linking offices or connecting to remote servers. Firewalls and routing settings make sure that only the intended traffic flows through this secure tunnel, and the usage of strong encryption protocols makes it difficult for attackers to breach the connection.

```
jeevanandh@jeevanandh-VirtualBox: ~  
File Edit View Search Terminal Help  
loaded plugins: charon aesni aes rc2 sha2 sha1 md4 md5 mgf1 random nonce x509  
revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey pem ope  
nssl fips-prf gmp agent xcbc hmac gcm attr kernel-netlink resolve socket-default  
connmark stroke updown eap-mschapv2 xauth-generic counters  
Listening IP addresses:  
10.0.2.5  
fd17:625c:f037:2:a00:27ff:fe20:f80f  
Connections:  
my_vpn: 10.0.2.5...10.0.2.11 IKEv1/2  
my_vpn: local: [machineA] uses pre-shared key authentication  
my_vpn: remote: [machineB] uses pre-shared key authentication  
my_vpn: child: 10.0.2.0/24 === 10.0.2.0/24 TUNNEL  
Security Associations (1 up, 0 connecting):  
my_vpn[2]: ESTABLISHED 16 seconds ago, 10.0.2.5[machineA]...10.0.2.11[mach  
ineB]  
my_vpn[2]: IKEv2 SPIs: 86ffd6605302e0a3_i 49de9c50841a221b_r*, pre-shared  
key reauthentication in 2 hours  
my_vpn[2]: IKE proposal: AES_CBC_128/HMAC_SHA2_256_128/PRF_AES128_XCBC/EC  
P_256  
my_vpn{1}: INSTALLED, TUNNEL, reqid 1, ESP SPIs: c1bd7d19_i c6fbda09_o  
my_vpn{1}: AES_CBC_128/HMAC_SHA2_256_128, 0 bytes_i, 0 bytes_o, rekeying  
in 42 minutes  
my_vpn{1}: 10.0.2.0/24 === 10.0.2.0/24  
jeevanandh@jeevanandh-VirtualBox:~$
```


A screenshot of a terminal window titled 'c1 [Running] - Oracle VM VirtualBox'. The terminal shows the output of the 'openvpn --info' command. The output includes system uptime (47 seconds), memory usage (malloc: sbrk 1622016, mmap 0, used 565264, free 1056752), worker threads (11 of 16 idle), loaded plugins (charon, aesni, aes, rc2, sha2, sha1, md4, md5, mgf1, random, nonce, x509, revocation, constraints, pubkey, pkcs1, pkcs7, pkcs8, pkcs12, pgp, dnskey, sshkey, pem, openssl, fips-prf, gmp, agent, xcbc, hmac, gcm, attr, kernel-netlink, resolve, socket-default, connmark, stroke, updown, eap-mschapv2, xauth-generic, counters), listening IP addresses (10.0.2.11), and connections (my_vpn: 10.0.2.11...10.0.2.5 IKEv1/2). It also shows security associations (1 up, 0 connecting) and a rekeying event (my_vpn{1}: AES_CBC_128/HMAC_SHA2_256_128, 0 bytes_i, 0 bytes_o, rekeying in 41 minutes). The terminal prompt is 'c123@c123: ~\$'.

NFS File Sharing:

NFS, or Network File System, is a protocol that allows computers to share files over a network, making it possible for multiple machines to access the same data as if it were stored locally on their disks. Developed by Sun Microsystems, NFS is especially useful in environments where resources need to be centralized and shared among different users or systems, such as in organizations, data centers, or labs. In its essence, NFS functions on a client-server model. The server is the host of the files and directories that one intends to share, while clients access these shared resources across the network. On the server, specific directories are "exported" or, in other words, marked as sharable. These are then "mounted" by the clients, meaning that the clients connect to these directories and make them appear as if they are part of the local file system. This allows users and applications on the client side to interact with the shared files as if they were stored locally.

NFS uses standard communication protocols and operates over TCP/IP to securely and efficiently transfer data. Permissions and access control are managed through the server, ensuring only authorized clients can connect to and interact with the shared directories. In some configurations, NFS can also support multiple users sharing the same files at the same time, which is useful in collaborative tasks or projects.

The important advantage of NFS is that it is simple and compatible. As it is platform-independent, it allows Linux, Unix, and other operating systems to communicate through NFS, making it quite versatile

in terms of file sharing. Besides, it does not require maintaining duplicate copies of files on different systems, which saves storage space as well as administrative overhead.

```
/dev/loop21      350M  350M    0 100% /snap/gnome-3-38-2004/143
/dev/loop22      506M  506M    0 100% /snap/gnome-42-2204/176
10.0.2.5:/srv/nfs_share 25G  11G  13G  47% /mnt/nfs_client_share
c123@c123:~$ cd /mnt/nfs_client_share
c123@c123:/mnt/nfs_client_share$ echo "Testing NFS from client" > textfile.txt
c123@c123:/mnt/nfs_client_share$ ls -l
total 4
-rw-rw-r-- 1 c123 c123 24 Dec  2 14:19 textfile.txt
c123@c123:/mnt/nfs_client_share$ cat testfile.txt
cat: testfile.txt: No such file or directory
c123@c123:/mnt/nfs_client_share$ cat textfile.txt
Testing NFS from client
c123@c123:/mnt/nfs_client_share$
```

```
jeevanandh@jeevanandh-VirtualBox:~$ sudo systemctl restart nfs-kernel-server
jeevanandh@jeevanandh-VirtualBox:~$ cd /srv/nfs_share
jeevanandh@jeevanandh-VirtualBox:/srv/nfs_share$ ls -l
total 4
-rw-rw-r-- 1 jeevanandh jeevanandh 24 Dec  2 14:19 textfile.txt
jeevanandh@jeevanandh-VirtualBox:/srv/nfs_share$ cat textfile.txt
Testing NFS from client
jeevanandh@jeevanandh-VirtualBox:/srv/nfs_share$
```

Future Scope:

- 1) Improved Security Features to further defend against online attacks, implement state-of-the-art security measures such as web application firewalls and intrusion detection systems.
2. Explore the availability of cloud services to enhance data redundancy, resource management, and backups in a scalable manner.
3. Automation and Orchestration: Automate server management, upgrade, and deployment processes by writing automation scripts using tools such as Ansible or Puppet.
- 4) Analytics and Performance Monitoring by using monitoring tools to analyze resource utilization, traffic patterns, and network performance. This will enable proactive management and optimization of the infrastructure.

References:

- 1) For Bind9 configuration: [bin9-doc](#)
- 2) For ISC DHCP Server configuration: [isc-dhcp-doc](#)
- 3) For hosting web page using Apache webserver: [apache-doc](#)

- 4) Firewall implementation: [ufw-doc](#)
- 5) Backup system scripting and automation: [backup-doc](#)