

# LAB 4 – NETWORK SERVICE APPLICATIONS

## Table of Contents

Table of Contents .....	1
Introduction .....	2
Router and Switch Configuration .....	3
Connecting to your table Switch and Router .....	4
Table Switch Configurations .....	5
Switch Running Configurations .....	6
Table Router Configurations .....	7
Router Running Configurations .....	8
Virtual Machine Configurations.....	9
VirtualBox Virtual Machine Cloning .....	10
GNS3 Topology .....	11
Netplan Configurations for each VM.....	12
DNS /etc/hosts Config .....	13
DNS /etc/hostname Config.....	14
DNS Zone File Config .....	15
DNS Forward Lookup Config.....	16
DNS Reverse Lookup Config .....	17
CA Certificate and Key Creation .....	18
HTTPS server(s) and CA Server – Self Signing the CA Certificate.....	19
Configuring nginx on the HTTPS server(s) .....	21
Configuring nginx on the Load Balancer .....	22
Configure nginx on the HTTP Server .....	23
Deliverables .....	24
Alternative Option: Using the CA to sign your Certificate .....	39
Appendix.....	40
References.....	49

## Introduction

This lab focuses on securing HTTP traffic and implementing workload balancing across multiple web servers using the HTTPS protocol. As part of the TELE20049 Network Service Application Lab, the primary objectives are to configure web servers to deliver secure web content, ensure proper network routing, and verify that the load balancer distributes traffic effectively across multiple servers.

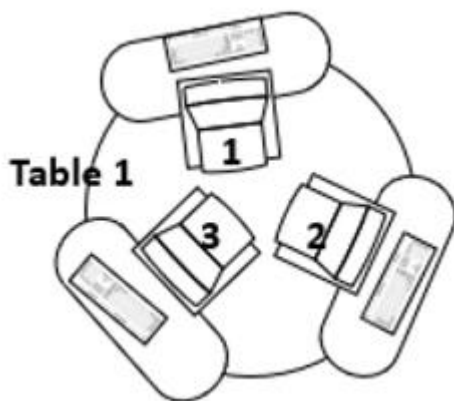
The lab environment, located in the S144 lab-classroom, is centered around a network topology that involves Juniper SRX-240 routers, Cisco and Juniper LAN switches, and a combination of physical and virtualized components managed through GNS3 and VirtualBox. Every participant (student) involved in this project is ultimately responsible for configuring the devices at their assigned table, ensuring that the entire setup is interconnected and responsive to HTTPS requests.

Key tasks include configuring the web servers to handle HTTPS traffic with security certificates signed by a Certificate Authority (CA) server, enabling OSPF on routers for routing advertisement, and setting up the load balancer to evenly distribute HTTPS requests to the web servers. The successful completion of the lab will demonstrate the ability to secure web traffic, manage network configurations, and maintain a balanced load across web servers, which are useful skills for managing networks and security.

## Router and Switch Configuration

To complete this lab, we used the router and switch assigned to our table at the back of S144. In order to configure the routers and switches, we have to:

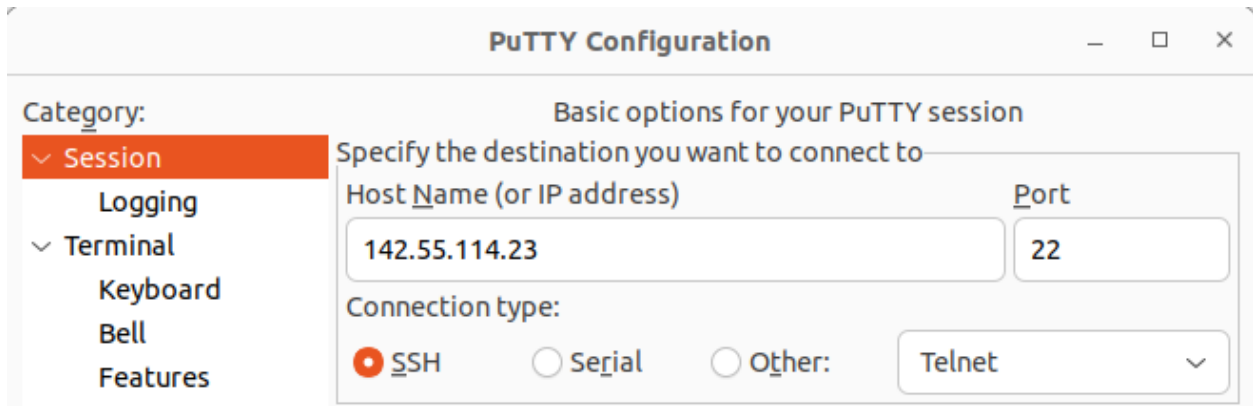
- Connect our computer to the table patch panel at the back using the ethernet ports (either 1A or 1B) located at your assigned computer in the class. Your ports may be labeled differently depending on the position of the computer you are assigned. I am assigned position 1, for table 2 based on the diagram below. Your position may be different.
  - You will want to bring up the interface and assign it the IP address that you were tasked with configuring, in my case I am under my own network of 192.168.8.0/24:  
**sudo ip addr add 192.168.8.2/24 dev enp3s0f0**  
**sudo ip link set enp3s0f0 up**



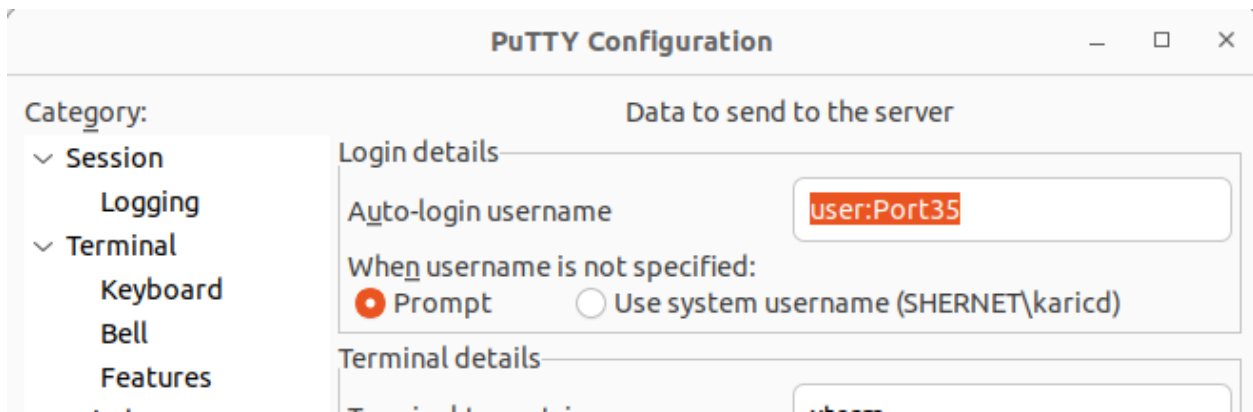
- Have your patch panel connected to your own chosen switch port that you will configure later (I chose port 22 on our tables switch so you will see later how we configure the switch). Each table will have a table router, and just below it will either be a Cisco or Juniper switch. In this lab, we will see how to configure a Cisco switch.
- Have your table switch connected to the table router,
- The table router (Juniper) will be connected to the common switch that will allow everyone's devices to communicate with each other.
- The table switch and router will also be connected to the Cyclades device that will allow us to SSH into the switch and router using PuTTY. In my case, the table router is using Port 5 for our table and the switch is using Port 35 (connected to the Cyclades device).

## Connecting to your table Switch and Router

Connecting to your switch or router depends on the ports that you configured. In PuTTY, you will want to connect to the address 142.55.114.23 over Port 22 under the Session category.



If you select Data under Category, you will have to type in your user:port# like below.



This is how you will connect to your switch and router to configure them.

The default username will be be **root**.

The password is **telecomS144**.

## Table Switch Configurations

Connect to your table router and use the following commands to configure it. It is Juniper, so the syntax will be different than what we are accustomed to so far utilizing GNS3 and Cisco Packet Tracer.

**enable**

**configure terminal** # goes into configuration mode

**hostname S2** # sets the switch name

**vlan 10**

**name VLAN10**

**exit**

**interface gigabitEthernet 1/0/22**

**switchport mode access**

**switchport access vlan 10**

**spanning-tree portfast**

**no shutdown**

**exit**

**interface gigabitEthernet 1/0/1**

**switchport mode trunk**

**switchport trunk allowed vlan 10**

**no shutdown**

**exit**

**wr**

## Switch Running Configurations

You can do the command **show running-config** on your switch to see the configurations you made for your ports as seen below. There will be a lot of information, but you can see the ports we configured in the image below. Since the switches and routers are used throughout the years by other students, you may see pre-existing configurations from before.

```

interface GigabitEthernet0/0
 vrf forwarding Ngmt-vrf
 no ip address
 negotiation auto
!
interface GigabitEthernet1/0/1
 switchport trunk allowed vlan 10
 switchport mode trunk
!
interface GigabitEthernet1/0/2
 switchport mode access
!
interface GigabitEthernet1/0/3
 description "VLAN 100 to J21"
 switchport access vlan 20
 switchport mode access
 spanning-tree portfast
!
interface GigabitEthernet1/0/4
 switchport access vlan 30
 switchport mode access
 spanning-tree portfast
!
interface GigabitEthernet1/0/5
 description "VLAN 200 to J22"
 switchport access vlan 200
 switchport mode access
!
interface GigabitEthernet1/0/6
 description "VLAN 200 to Table 2 Seat 2"
 switchport access vlan 200
 switchport mode access

```

^Port 1/0/1^

```

interface GigabitEthernet1/0/14
!
interface GigabitEthernet1/0/15
!
interface GigabitEthernet1/0/16
!
interface GigabitEthernet1/0/17
!
interface GigabitEthernet1/0/18
!
interface GigabitEthernet1/0/19
!
interface GigabitEthernet1/0/20
!
interface GigabitEthernet1/0/21
 switchport access vlan 30
 switchport mode access
 spanning-tree portfast
!
interface GigabitEthernet1/0/22
 switchport access vlan 10
 switchport mode access
 spanning-tree portfast
!
interface GigabitEthernet1/0/23
!
interface GigabitEthernet1/0/24
 switchport mode access
!
interface GigabitEthernet1/1/1
!
interface GigabitEthernet1/1/2
!
interface GigabitEthernet1/1/3
!
--More--

```

^Port 1/0/22^

## Table Router Configurations

Using PuTTY, you want to SSH into your table router to start making configurations. Follow the commands below in order.

**cli**

**configure**

**set system host-name R2**

**set system login user dino class super-user**

**set system login user dino authentication plain-text-password**

**set interfaces ge-0/0/13 vlan-tagging**

**set interfaces ge-0/0/13 unit 10 vlan-id 10 family inet address 100.64.144.2/24**

**set interfaces ge-0/0/0 unit 0 family inet address 192.168.8.1/24**

**set protocols ospf area 0.0.0.0 interface all**

**commit and-quit**



## Router Running Configurations

You can do any of the commands below to see more information on what you have configured for your router, I will only do the first one and show a sample. You do not want to be in configuration mode for this.

**show configuration**

**show configuration | display set**

**show route**

**show route terse**

```
root@J2> show configuration
## Last commit: 2024-08-07 20:43:06 UTC by root
version 12.3X48-D51;
system {
  host-name J2;
  root-authentication {
    encrypted-password "$1$yquekwrY$P3G1GRsfHUJj3Je1K,6fJ."; ## SECRET-DATA
  }
  login {
    user anas {
      uid 2012;
      class super-user;
      authentication {
        encrypted-password "$1$PzfX7P0c$2lkr0QJ1c7CCt/XtlaBw10"; ## SECRET-DATA
      }
    }
    user dino {
      uid 2010;
      class super-user;
      authentication {
        encrypted-password "$1$bRdrTawq$3Ys.DvgryEQVLmBtoZSIL/"; ## SECRET-DATA
      }
    }
  }
}
```

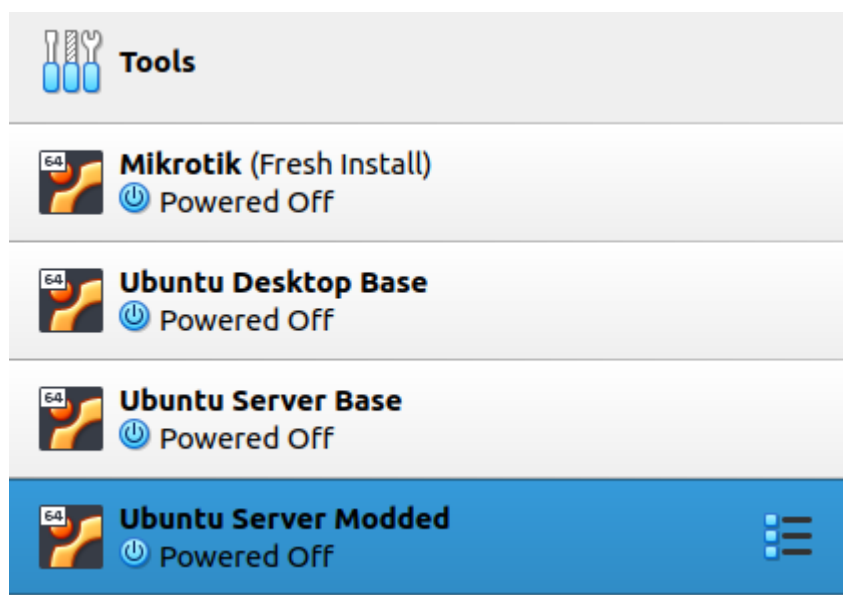
```
interfaces {
  ge-0/0/0 {
    vlan-tagging;
    unit 10 {
      vlan-id 10;
      family inet {
        address 192.168.8.1/24;
      }
    }
  }
}
```

```
protocols {
  ospf {
    area 0.0.0.0 {
      interface all;
    }
  }
}
vpls {
  VLAN10 {
    vlan-id 10;
  }
  VLAN30 {
    vlan-id 30;
  }
  krushnakant {
    vlan-id 200;
  }
  vlan-trust {
    vlan-id 3;
    l3-interface vlan.0;
  }
}
```

## Virtual Machine Configurations

Before we begin diving into each unique configuration for every type of server we have, we want to establish a few things first before we begin cloning all the VMs required and importing them into GNS3.

In VirtualBox, you will want to setup a base Ubuntu Server VM like so. In this case, I am using **Ubuntu Server Modded** for this:



If this is your first time, you will want to go through the whole guided setup of downloading the required packages, naming your server, etc. We will do this because once we import our VMs into GNS3, we will not have internet to install the packages.

Once you are finished, load the VM and open the terminal and install these different packages:

```
sudo apt update
```

```
sudo apt install nginx -y
```

```
sudo apt install bind9 bind9utils dnsutils bind9-doc -y
```

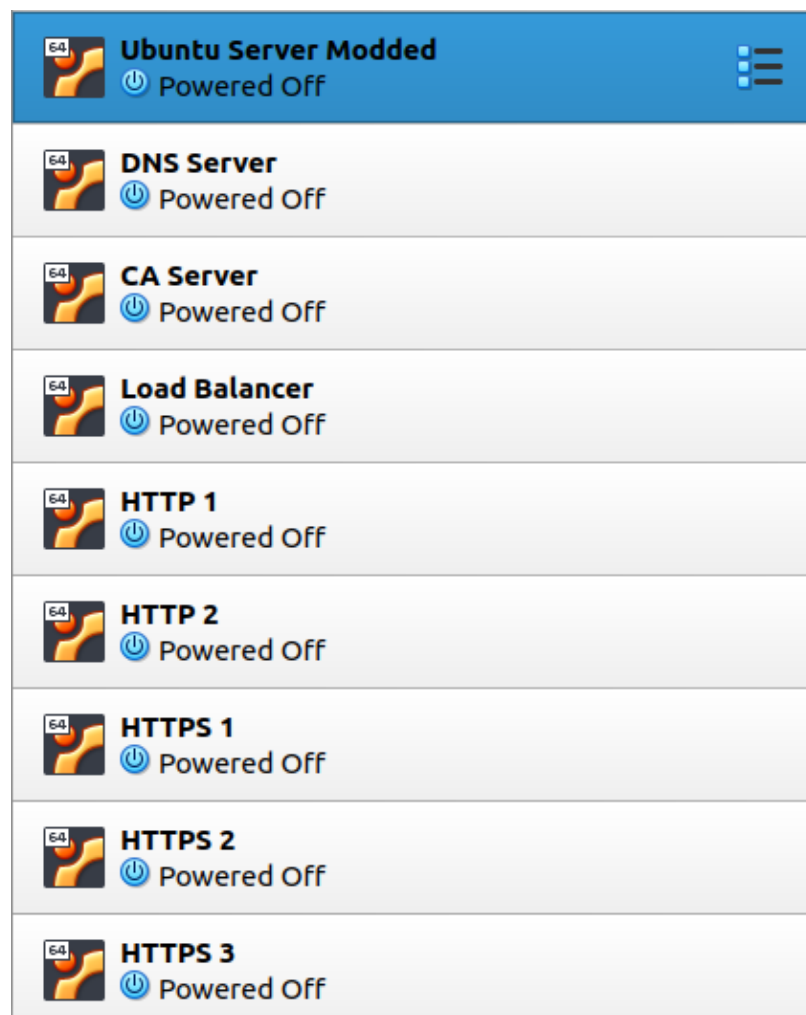
```
sudo apt install openssl (usually comes with ubuntu server but do this for good measures)
```

Shut the VM off when you are done. This will take up more space when cloning your VMs on your computer, but we are not concerned with storage right now because we will have enough to conduct this lab using the S144 computers.

## VirtualBox Virtual Machine Cloning

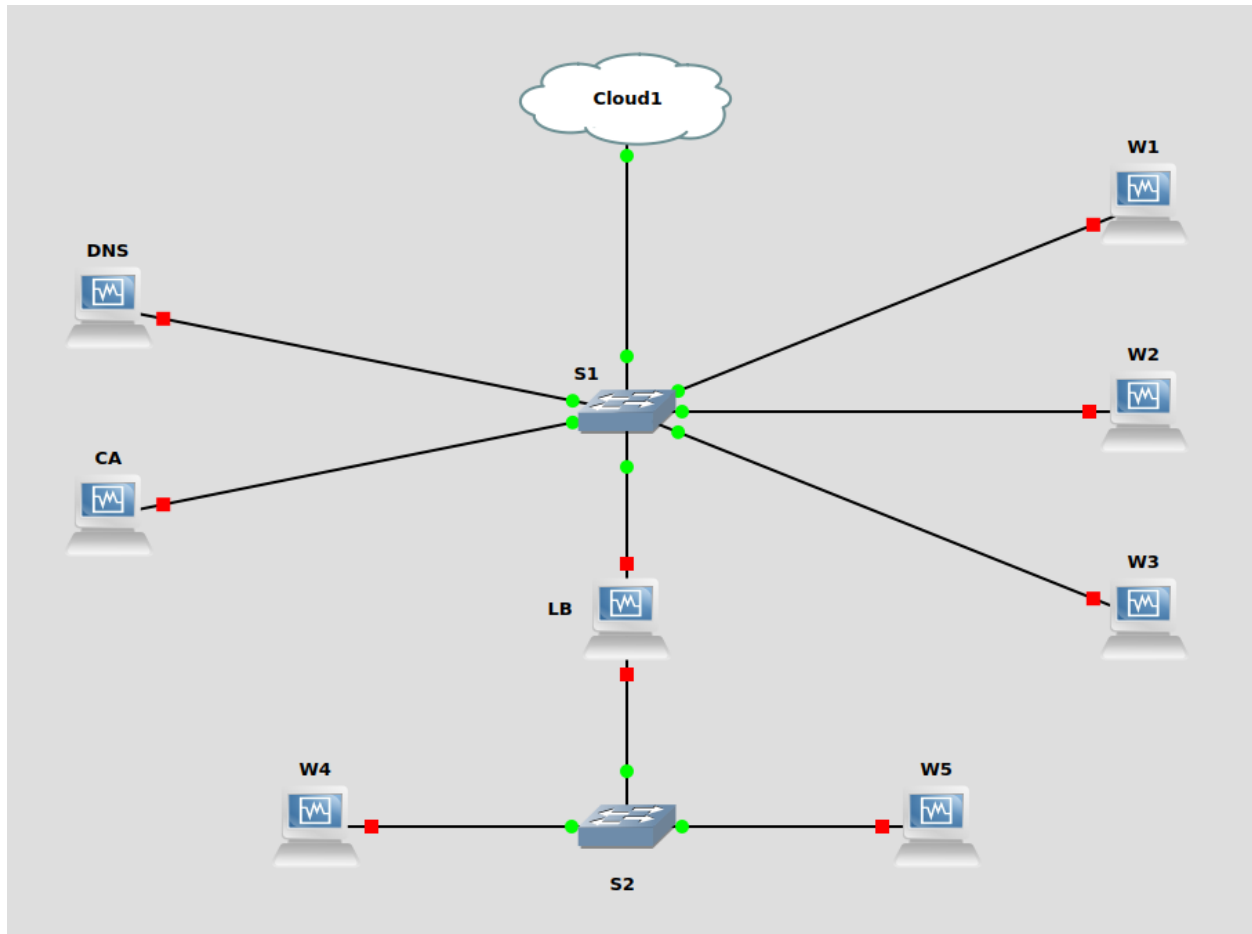
Now we will clone our modded base VM to create 8 types of virtual machines (remember to generate new MAC addresses in VirtualBox):

- DNS Server
- CA Server
- Load Balancer
- HTTP Server 1
- HTTP Server 2
- HTTPS Server 1
- HTTPS Server 2
- HTTPS Server 3



## GNS3 Topology

Import all your VMs into GNS3 and let GNS3 take control of your network adapters. You will want to have 1 network adapter for every server except your load balancer will have 2. Have your topology set up like the photo below.



## Netplan Configurations for each VM

On every VM, you will want to configure your netplan YAML file configuration located in `/etc/netplan` to have a similar pattern (with the spacing to):

network:

version: 2

ethernets:

enp0s3:

addresses:

- 192.168.8.xx/24

routes:

- to: 0.0.0.0/0

via: 192.168.8.1

nameservers:

addresses: [192.168.8.10]

You will want to adjust the highlighted address to insert the IP address depending on the VM you are configuring:

- DNS Server      192.168.8.10
- CA Server      192.168.8.11
- Load Balancer   192.168.8.20 (interface 1) & 172.16.0.1 (interface 2)
- HTTP Server 1   172.16.0.5
- HTTP Server 2   172.16.0.10
- HTTPS Server 1   192.168.8.21
- HTTPS Server 2   192.168.8.22
- HTTPS Server 3   192.168.8.23

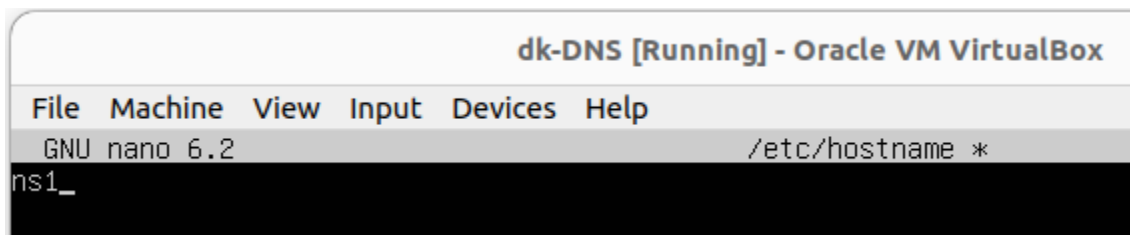
The xx in the highlighted address refers to the last octet of each IP address.

**sudo netplan apply** these changes when you are done.

## DNS /etc/hosts Config

```
dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 6.2 /etc/hosts *
127.0.0.1 localhost
127.0.1.1 s144
192.168.8.10 ns1.dino.org ns1_
# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
```

## DNS /etc/hostname Config

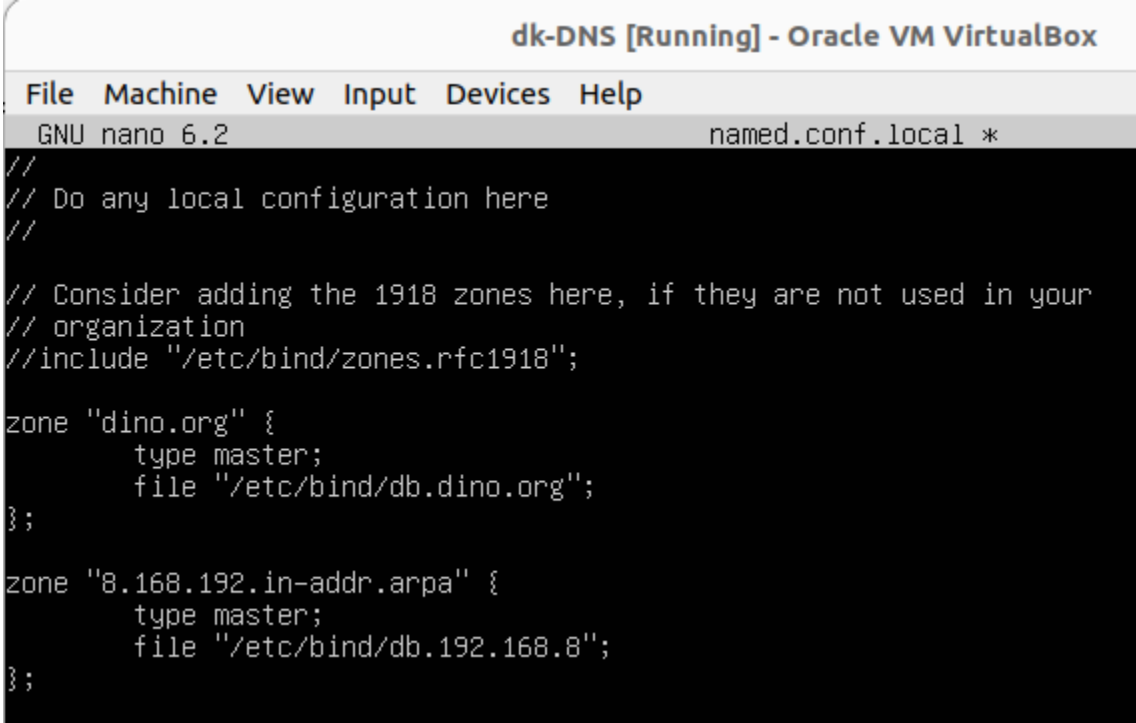


```
dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 6.2 /etc/hostname *
ns1_
```

Do the command **hostname -F /etc/hostname** to apply this change

## DNS Zone File Config

Create a zone up file on your DNS VM and have configurations like that of below. My domain is dino.org.



```
dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 6.2 named.conf.local *
//
// Do any local configuration here
//
// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "dino.org" {
    type master;
    file "/etc/bind/db.dino.org";
};

zone "8.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192.168.8";
};
```



## DNS Forward Lookup Config

```

dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
;
; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA      ns1.dino.org.  admin.dino.org. (
                        2      ; Serial
                        604800 ; Refresh
                        86400  ; Retry
                        2419200 ; Expire
                        604800 ) ; Negative Cache TTL
;
@         IN      NS       ns1.dino.org.
@         IN      A        192.168.8.10
ns1       IN      A        192.168.8.10
w1       IN      A        192.168.8.21
w2       IN      A        192.168.8.22
w3       IN      A        192.168.8.33
w4       IN      A        192.168.8.15
w5       IN      A        192.168.8.25
ca       IN      A        192.168.8.11
lb       IN      A        192.168.8.20

dino.org.      IN      A      192.168.8.20
dino.org.      IN      A      192.168.8.21
dino.org.      IN      A      192.168.8.22
dino.org.      IN      A      192.168.8.23

www.dino.org.  IN      CNAME   dino.org.

```

## DNS Reverse Lookup Config

```

dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:/etc/bind$ cat db.192.168.8
;
; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA      dino.org.      admin.dino.org. (
                        2          ; Serial
                        604800     ; Refresh
                        86400      ; Retry
                        2419200    ; Expire
                        604800 )   ; Negative Cache TTL
;
@         IN      NS       ns1.
@         IN      NS       ns1.dino.org.
10        IN      PTR      ns1.dino.org.
11        IN      PTR      ca.dino.org.
20        IN      PTR      lb.dino.org.
21        IN      PTR      w1.dino.org.
22        IN      PTR      w2.dino.org.
23        IN      PTR      w3.dino.org.
15        IN      PTR      w4.dino.org.
25        IN      PTR      w5.dino.org.
dino@s144:/etc/bind$

```

Status:

```

dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:/etc/bind$ sudo systemctl restart bind9
dino@s144:/etc/bind$ sudo systemctl status bind9
• named.service - BIND Domain Name Server
  Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2024-08-08 21:55:20 UTC; 3s ago
    Docs: man:named(8)
  Process: 1246 ExecStart=/usr/sbin/named $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 1247 (named)
    Tasks: 6 (limit: 4557)
  Memory: 5.6M
    CPU: 40ms
  CGroup: /system.slice/named.service
          └─1247 /usr/sbin/named -u bind

```

## CA Certificate and Key Creation

You can either make a new directory for this, but you want to create a private key and certificate using these commands that we will sign on our LB and HTTPS servers. I will just do it in my ~/ directory on my CA server to simplify things:

**openssl genrsa -out cakey.pem 2048**

**openssl req -new -x509 -key cakey.pem -out cacert.pem -days 365**

```

dk-CA [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:~$ openssl genrsa -out cakey.pem 2048
dino@s144:~$ openssl req -new -x509 -key cakey.pem -out cacert.pem -days 365
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CA
State or Province Name (full name) [Some-State]:Ontario
Locality Name (eg, city) []:Oakville
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Sheridan College
Organizational Unit Name (eg, section) []:S144
Common Name (e.g. server FQDN or YOUR name) []:ca.dino.org
Email Address []:
dino@s144:~$
dino@s144:~$

```

During creation of the cert, it will ask you a few questions like your country, province, company, etc. I left the email part blank (just press enter on that step). You want to secure copy this to a location you will remember on each HTTPS VM and the Load Balancer VM too using the commands:

**scp cakey.pem user@<vm\_ip\_address>:~/path/to/wherever**

**scp cacert.pem user@<vm\_ip\_address>:~/path/to/wherever**

**NOTE:** This method is for self-signing the certificate. If you wish to not do this, follow the commands instead that are in the section **“Alternative Option: Using the CA to sign your Certificate”**

## HTTPS server(s) and CA Server – Self Signing the CA Certificate

Generate a private key for the HTTPS/CA server (I just did this in the ~/ directory again):

**openssl genrsa -out server.key 2048**

Generate a certificate signing request (CSR)

**openssl req -new -key server.key -out server.csr**

Sign the CSR with the CA certificate and key:

**openssl x509 -req -in server.csr -CA cacert.pem -CAkey cakey.pem -CAcreateserial -out server.crt -days 365**

```

dk-W1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:~$ ls
cacert.pem  cakey.pem
dino@s144:~$ openssl genrsa -out server.key 2048
dino@s144:~$ ls
cacert.pem  cakey.pem  server.key
dino@s144:~$ openssl req -new -key server.key -out server.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CA
State or Province Name (full name) [Some-State]:Ontario
Locality Name (eg, city) []:Oakville
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Sheridan College
Organizational Unit Name (eg, section) []:S144
Common Name (e.g. server FQDN or YOUR name) []:w1.dino.org
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
dino@s144:~$ _

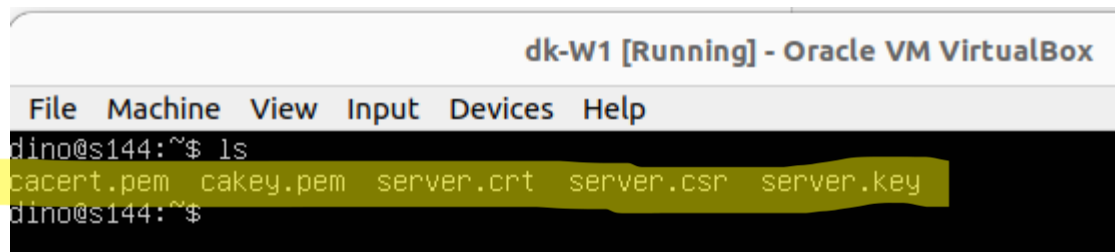
```

```

dk-W1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:~$ ls
cacert.pem  cakey.pem  server.csr  server.key
dino@s144:~$ openssl x509 -req -in server.csr -CA cacert.pem -CAkey cakey.pem -CAcreateserial -out s
erver.crt -days 365
Certificate request self-signature ok
subject=C = CA, ST = Ontario, L = Oakville, O = Sheridan College, OU = S144, CN = w1.dino.org
dino@s144:~$

```

Repeat this process for each HTTPS server and the Load Balancer.

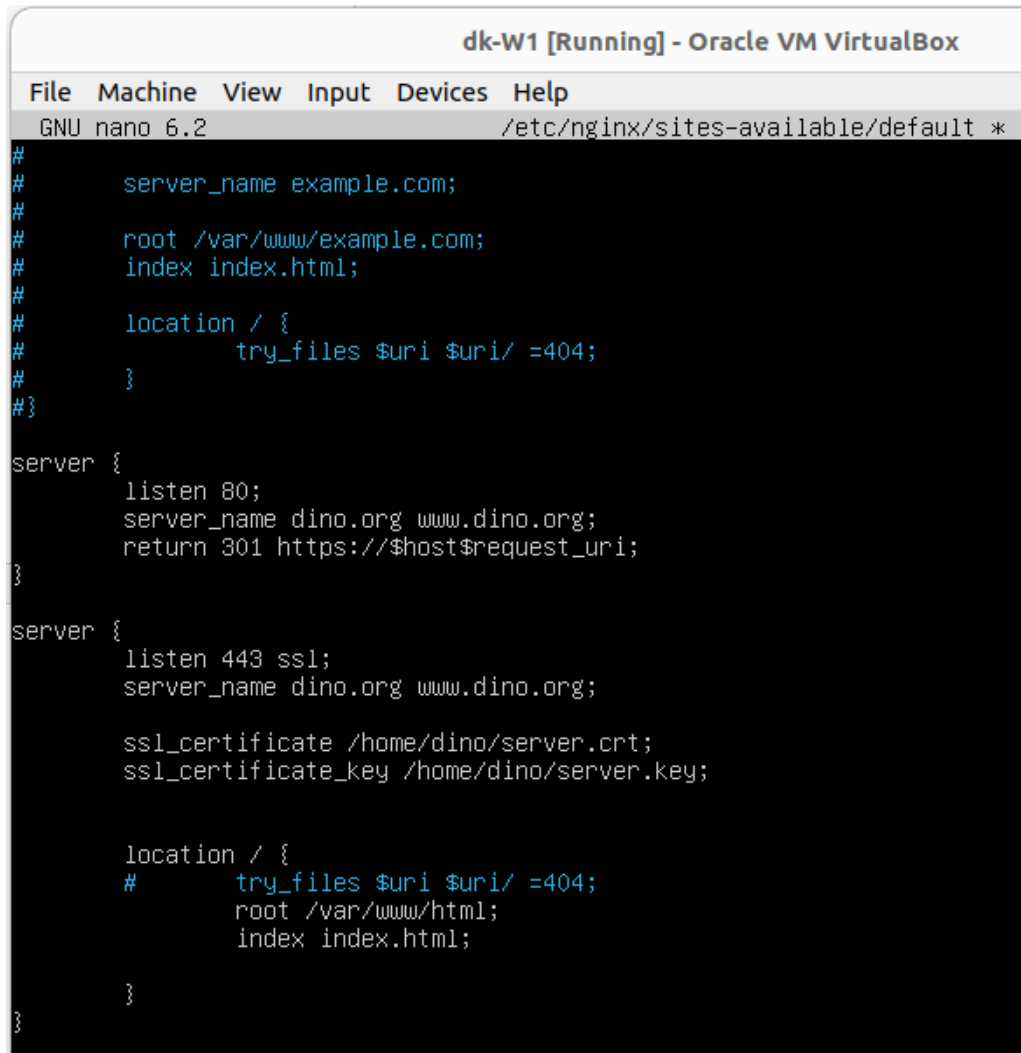


```
dk-W1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:~$ ls
cacert.pem  cakey.pem  server.crt  server.csr  server.key
dino@s144:~$
```

## Configuring nginx on the HTTPS server(s)

Apply these configurations to each of your to HTTP server 1 (w1), server 2 (w2), server 3 (w3)

Edit file in `/etc/nginx/sites-available/default` to have the configurations appended at the bottom like the photo below:



```

dk-W1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 6.2 /etc/nginx/sites-available/default *
#
#       server_name example.com;
#
#       root /var/www/example.com;
#       index index.html;
#
#       location / {
#           try_files $uri $uri/ =404;
#       }
#}

server {
    listen 80;
    server_name dino.org www.dino.org;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name dino.org www.dino.org;

    ssl_certificate /home/dino/server.crt;
    ssl_certificate_key /home/dino/server.key;

    location / {
#       try_files $uri $uri/ =404;
        root /var/www/html;
        index index.html;
    }
}

```

Create symbolic links and restart Nginx:

**`sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/`**

**`sudo systemctl restart nginx`**

Be wary of the name of your index.html file in **`/var/www/html`**

## Configuring nginx on the Load Balancer

Configure the `/etc/nginx/sites-available/default` file

```

dk-LB [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 6.2 default
# try_files $uri $uri/ =404;
# }
#}

upstream dino {
    server 172.16.0.15;
    server 172.16.0.25;
}

server {
    listen 80;
    server_name dino.org lb.dino.org;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name dino.org www.dino.org;

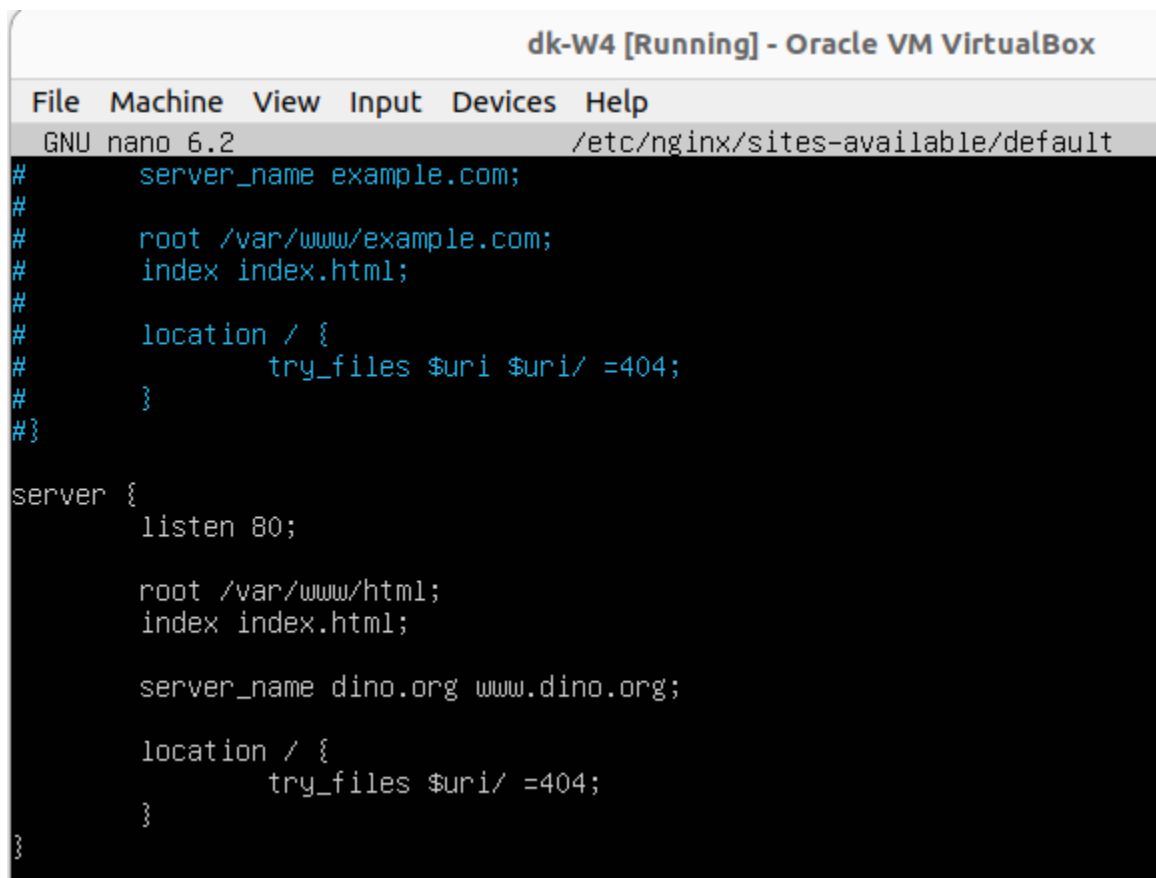
    ssl_certificate /home/dino/server.crt;
    ssl_certificate_key /home/dino/server.key;

    location / {
        proxy_pass http://dino;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

```

## Configure nginx on the HTTP Server

Do these for both HTTP server 1 (w1) and server 2 (w2)



The screenshot shows a terminal window titled "dk-W4 [Running] - Oracle VM VirtualBox". The terminal is running the GNU nano 6.2 text editor, editing the file /etc/nginx/sites-available/default. The configuration file contains the following content:

```
#
#   server_name example.com;
#
#   root /var/www/example.com;
#   index index.html;
#
#   location / {
#       try_files $uri $uri/ =404;
#   }
#}

server {
    listen 80;

    root /var/www/html;
    index index.html;

    server_name dino.org www.dino.org;

    location / {
        try_files $uri/ =404;
    }
}
```



## Deliverables

For this report, I will conduct an nslookup, ping test, status check, and curl to show our configurations in action for the HTTP and HTTPS servers. Then we will go further one step further. Later I will sniff the transactions between web server 1 (HTTPS) and the device I am curling from. I will also sniff the transactions between the Load Balancer and web server 4 (HTTP). The tests will be done from the DNS server.

---

### Load Balancer

Ping test successful to LB:

```

dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:/etc/bind$ ping 192.168.8.20
PING 192.168.8.20 (192.168.8.20) 56(84) bytes of data.
64 bytes from 192.168.8.20: icmp_seq=1 ttl=64 time=1.58 ms
64 bytes from 192.168.8.20: icmp_seq=2 ttl=64 time=0.837 ms
^C
--- 192.168.8.20 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.837/1.206/1.576/0.369 ms
dino@s144:/etc/bind$

```

The forward and reverse DNS queries are successful:

```

dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:/etc/bind$ nslookup lb.dino.org
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
Name:   lb.dino.org
Address: 192.168.8.20

dino@s144:/etc/bind$ nslookup 192.168.8.20
20.8.168.192.in-addr.arpa      name = lb.dino.org.

Authoritative answers can be found from:

dino@s144:/etc/bind$

```

## Status Check Successful:

```

dk-LB [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:~$ sudo systemctl status nginx
• nginx.service - A high performance web server and a reverse proxy server
  Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
  Active: active (running) since Fri 2024-08-09 01:10:08 UTC; 36min ago
    Docs: man:nginx(8)
  Process: 1710 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0)
  Process: 1712 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0)
 Main PID: 1714 (nginx)
   Tasks: 3 (limit: 4557)
  Memory: 4.0M
    CPU: 35ms
  CGroup: /system.slice/nginx.service
          └─1714 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
            └─1715 "nginx: worker process"
              └─1716 "nginx: worker process"

Aug 09 01:10:08 s144 systemd[1]: Starting A high performance web server and a reverse proxy server.
Aug 09 01:10:08 s144 systemd[1]: Started A high performance web server and a reverse proxy server.
lines 1-17/17 (END)

```

## Curling the HTTP Server http://www.dino.org:

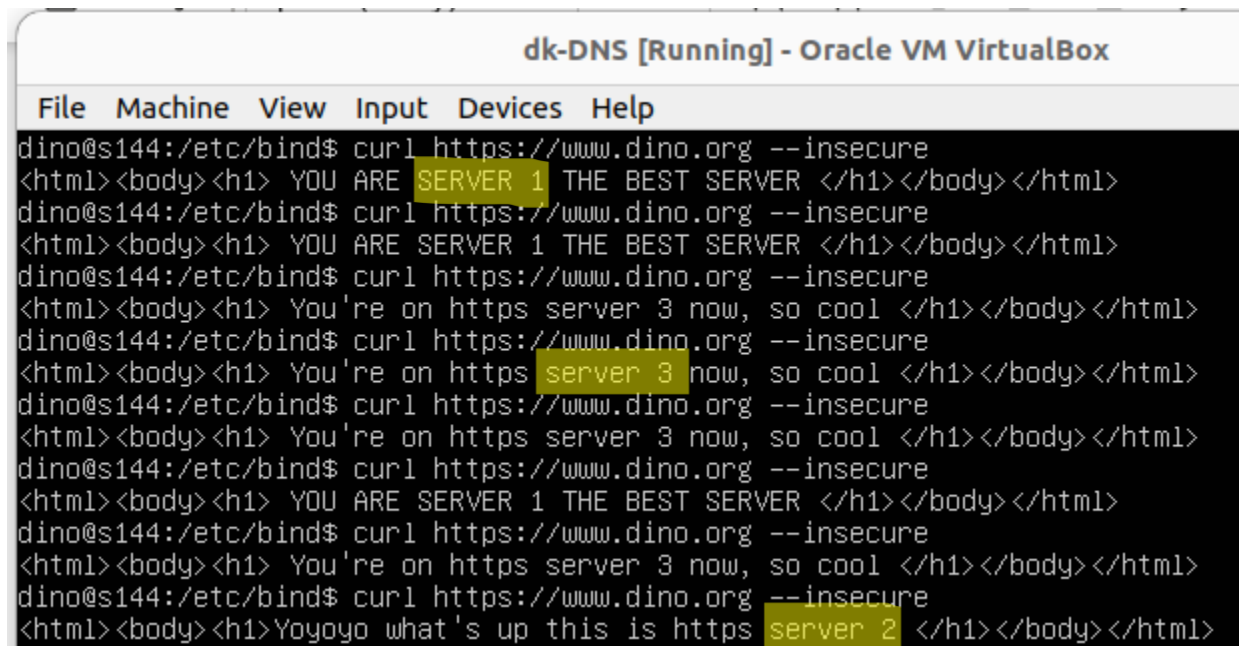
```

dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:/etc/bind$ curl http://www.dino.org
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
dino@s144:/etc/bind$ _

```

Curling the HTTPS Server <https://www.dino.org>

```
dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:/etc/bind$ curl https://www.dino.org --insecure
<html><body><h1> YOU ARE SERVER 1 THE BEST SERVER </h1></body></html>
dino@s144:/etc/bind$ curl https://www.dino.org --insecure
<html><body><h1> YOU ARE SERVER 1 THE BEST SERVER </h1></body></html>
dino@s144:/etc/bind$ curl https://www.dino.org --insecure
<html><body><h1> You're on https server 3 now, so cool </h1></body></html>
dino@s144:/etc/bind$ curl https://www.dino.org --insecure
<html><body><h1> You're on https server 3 now, so cool </h1></body></html>
dino@s144:/etc/bind$ curl https://www.dino.org --insecure
<html><body><h1> You're on https server 3 now, so cool </h1></body></html>
dino@s144:/etc/bind$ curl https://www.dino.org --insecure
<html><body><h1> YOU ARE SERVER 1 THE BEST SERVER </h1></body></html>
dino@s144:/etc/bind$ curl https://www.dino.org --insecure
<html><body><h1> You're on https server 3 now, so cool </h1></body></html>
dino@s144:/etc/bind$ curl https://www.dino.org --insecure
<html><body><h1>Yoyoyo what's up this is https server 2 </h1></body></html>
```

Server 1, 2, and 3 are seen above for our HTTPS servers (w1, w2, and w3)

## W1 (HTTPS Server 1)

Ping test successful to W1:

```

dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:/etc/bind$ ping 192.168.8.21
PING 192.168.8.21 (192.168.8.21) 56(84) bytes of data.
64 bytes from 192.168.8.21: icmp_seq=1 ttl=64 time=1.53 ms
64 bytes from 192.168.8.21: icmp_seq=2 ttl=64 time=1.83 ms
^C
--- 192.168.8.21 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.528/1.679/1.831/0.151 ms
dino@s144:/etc/bind$ _

```

Nslookup successful:

```

dk-DNS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:/etc/bind$ nslookup w1.dino.org
Server:          127.0.0.53
Address:         127.0.0.53#53

Non-authoritative answer:
Name:   w1.dino.org
Address: 192.168.8.21

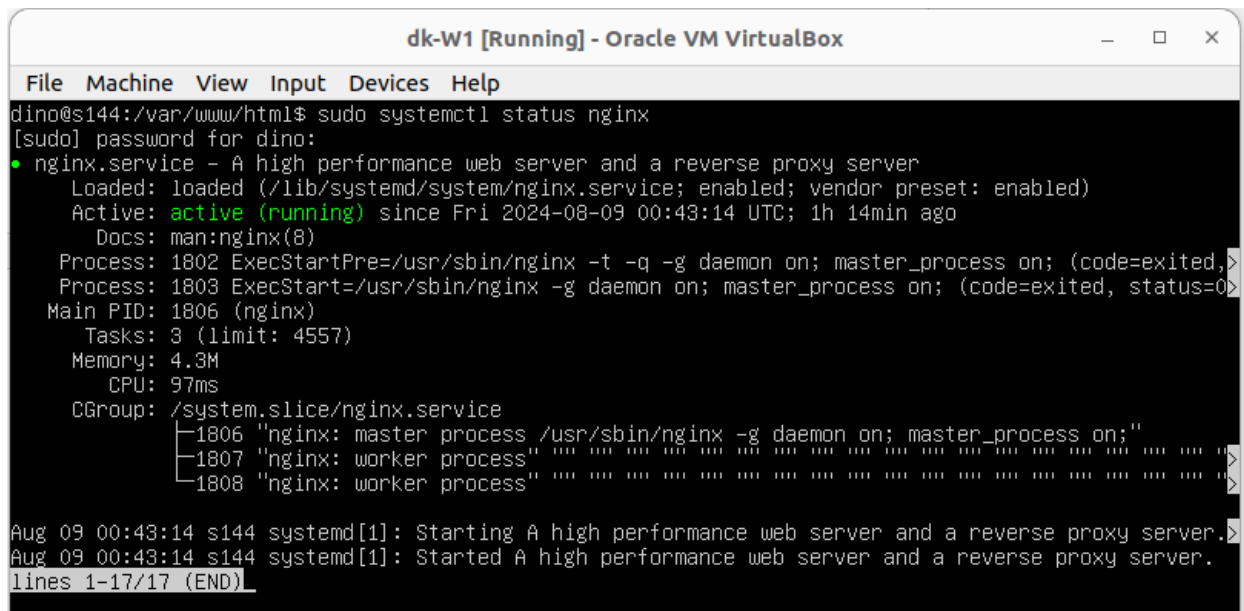
dino@s144:/etc/bind$ nslookup 192.168.8.21
21.8.168.192.in-addr.arpa      name = w1.dino.org.

Authoritative answers can be found from:

dino@s144:/etc/bind$

```

## Status Check Nginx:

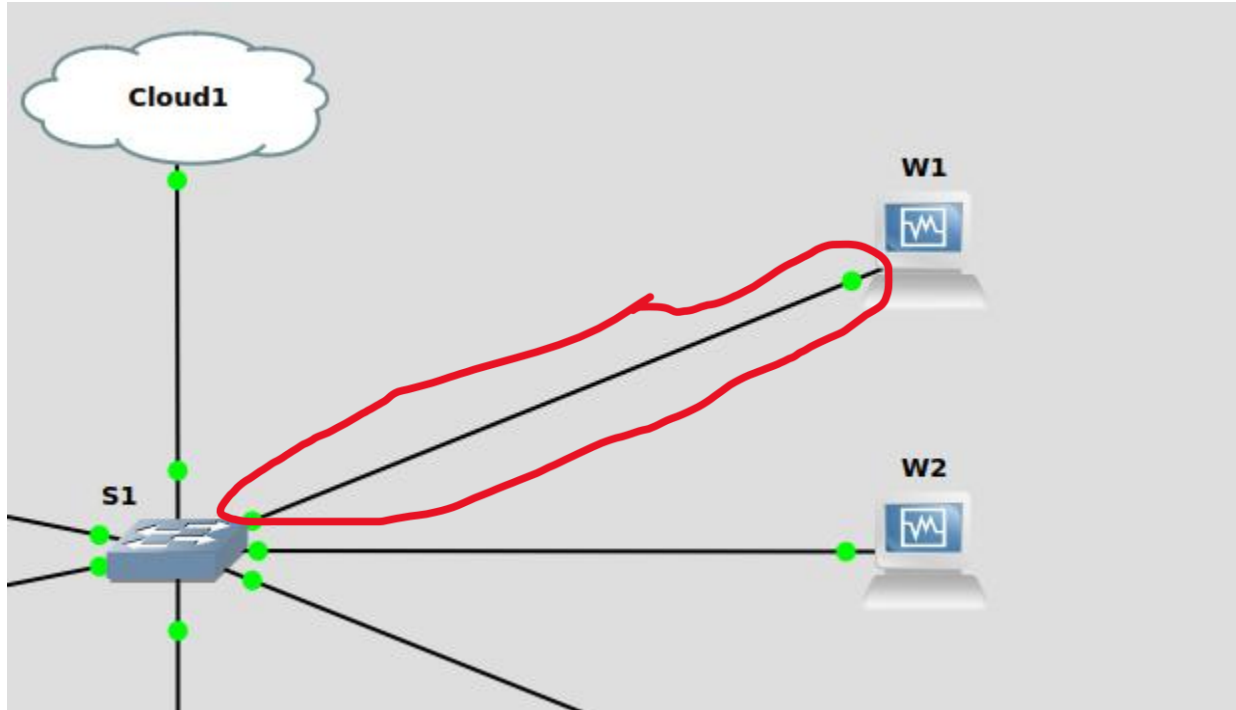


```
dk-W1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
dino@s144:/var/www/html$ sudo systemctl status nginx
[sudo] password for dino:
• nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2024-08-09 00:43:14 UTC; 1h 14min ago
     Docs: man:nginx(8)
   Process: 1802 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0)
   Process: 1803 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0)
  Main PID: 1806 (nginx)
    Tasks: 3 (limit: 4557)
   Memory: 4.3M
      CPU: 97ms
   CGroup: /system.slice/nginx.service
           └─1806 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─1807 "nginx: worker process"
             └─1808 "nginx: worker process"

Aug 09 00:43:14 s144 systemd[1]: Starting A high performance web server and a reverse proxy server.
Aug 09 00:43:14 s144 systemd[1]: Started A high performance web server and a reverse proxy server.
lines 1-17/17 (END)
```

Sniffing and analyzing the transactions of the web access toward webserver W1.

Start a WireShark capture on the highlighted node:



## The WireShark Capture:

216 58.000525	Cisco c6:1c:16	Spanning-tree-(for-...	STP	60 Conf. Root = 32768/10/58:ac:78:c6:1c:00 Cost = 0 Port = 0x8016
217 59.780314	192.168.8.1	224.0.0.5	OSPF	90 Hello Packet
218 59.964979	192.168.8.10	192.168.8.21	TCP	74 54466 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1490713512 TSecr=6
219 59.965901	192.168.8.21	192.168.8.10	TCP	74 443 → 54466 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1098348
220 59.966730	192.168.8.10	192.168.8.21	TCP	66 54466 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1490713513 TSecr=1098348531
221 59.977521	192.168.8.10	192.168.8.21	TLSv1.3	583 Client Hello
222 59.978260	192.168.8.21	192.168.8.10	TCP	66 443 → 54466 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSval=1098348543 TSecr=1490713524
223 59.981759	192.168.8.21	192.168.8.10	TLSv1.3	1514 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data
224 59.981835	192.168.8.21	192.168.8.10	TLSv1.3	72 Application Data
225 59.982500	192.168.8.10	192.168.8.21	TCP	66 54466 → 443 [ACK] Seq=518 Ack=1455 Win=64128 Len=0 TSval=1490713529 TSecr=1098348547
226 59.986834	192.168.8.10	192.168.8.21	TLSv1.3	146 Change Cipher Spec, Application Data
227 59.987072	192.168.8.10	192.168.8.21	TLSv1.3	164 Application Data
228 59.988049	192.168.8.21	192.168.8.10	TLSv1.3	369 Application Data
229 59.988441	192.168.8.21	192.168.8.10	TLSv1.3	369 Application Data
230 59.988767	192.168.8.21	192.168.8.10	TLSv1.3	403 Application Data
231 59.989166	192.168.8.10	192.168.8.21	TCP	66 54466 → 443 [ACK] Seq=696 Ack=2061 Win=64128 Len=0 TSval=1490713536 TSecr=1098348553
232 59.989811	192.168.8.10	192.168.8.21	TLSv1.3	90 Application Data
233 59.989880	192.168.8.10	192.168.8.21	TCP	66 54466 → 443 [FIN, ACK] Seq=720 Ack=2398 Win=64128 Len=0 TSval=1490713537 TSecr=109834
234 59.990663	192.168.8.21	192.168.8.10	TCP	66 443 → 54466 [FIN, ACK] Seq=2398 Ack=721 Win=64768 Len=0 TSval=1098348556 TSecr=149071
235 59.991292	192.168.8.10	192.168.8.21	TCP	66 54466 → 443 [ACK] Seq=721 Ack=2399 Win=64128 Len=0 TSval=1490713538 TSecr=1098348556

Close up:

STP	60 Conf. Root = 32768/10/58:ac:78:c6:1c:00
OSPF	90 Hello Packet
TCP	74 54466 → 443 [SYN] Seq=0 Win=64240 Len=
TCP	74 443 → 54466 [SYN, ACK] Seq=0 Ack=1 Win
TCP	66 54466 → 443 [ACK] Seq=1 Ack=1 Win=6425
TLSv1.3	583 Client Hello
TCP	66 443 → 54466 [ACK] Seq=1 Ack=518 Win=64
TLSv1.3	1514 Server Hello, Change Cipher Spec, Appl
TLSv1.3	72 Application Data
TCP	66 54466 → 443 [ACK] Seq=518 Ack=1455 Win
TLSv1.3	146 Change Cipher Spec, Application Data
TLSv1.3	164 Application Data
TLSv1.3	369 Application Data
TLSv1.3	369 Application Data
TLSv1.3	403 Application Data
TCP	66 54466 → 443 [ACK] Seq=696 Ack=2061 Win
TLSv1.3	90 Application Data
TCP	66 54466 → 443 [FIN, ACK] Seq=720 Ack=239
TCP	66 443 → 54466 [FIN, ACK] Seq=2398 Ack=72
TCP	66 54466 → 443 [ACK] Seq=721 Ack=2399 Win

## Wireshark Capture W1 Explanation

### 1. TCP 3-Way Handshake

The TCP 3-way handshake is the initial process by which the client and server establish a TCP connection before any TLS communication begins. The steps are:

**SYN (Synchronize):** The client initiates a connection by sending a SYN packet to the server.

- **Frame 218:** The client (192.168.8.10) sends a SYN packet to the server (192.168.8.21) on port 443, requesting to establish a connection.

**SYN-ACK (Synchronize-Acknowledge):** The server responds to the client with a SYN-ACK, acknowledging the connection request.

- **Frame 219:** The server (192.168.8.21) responds with a SYN-ACK, indicating that it has received the client's request and is ready to establish the connection.

**ACK (Acknowledge):** The client sends an ACK packet to the server, completing the 3-way handshake.

- **Frame 220:** The client (192.168.8.10) sends an ACK packet, completing the handshake and establishing a TCP connection.



## 2. TLS Handshake and Delivery of the Security Certificate

Once the TCP connection is established, the TLS handshake begins. This process is then used to establish a secure communication channel between the client and server.

**Client Hello:** The client initiates the TLS handshake by sending a Client Hello message, proposing cryptographic parameters to the server.

- **Frame 221:** The client sends a "Client Hello" message, which includes supported cipher suites, TLS versions, and other parameters necessary to establish a secure session.

**Server Hello and Certificate Delivery:** The server responds with a "Server Hello" message, selecting the appropriate cipher suite and TLS version. The server also sends its digital certificate to the client for authentication.

- **Frame 224:** The server responds with a "Server Hello" message, agreeing on the cipher suite and other parameters. The server's certificate is also delivered in this message to authenticate the server to the client.

### 3. Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange (or Elliptic-Curve Diffie-Hellman, ECDHE) is a method used in this case that would be used to securely exchange cryptographic keys over a public channel. This process ensures that both the client and server can generate a shared secret key used to encrypt the communication.

**Server Key Exchange:** If Diffie-Hellman is being used, the server will send its Diffie-Hellman parameters (such as the public key) in the Server Key Exchange message.

- **Frame 224:** The server sends the key exchange parameters (such as the Diffie-Hellman public value) as part of the TLS handshake.

**Client Key Exchange:** The client generates its own Diffie-Hellman public key, sends it to the server, and both parties calculate the shared secret.

- This step is not shown in the capture snippet but would typically follow the Server Hello.

#### 4. TLS Messages

After the key exchange, the client and server use the shared secret key to encrypt the session. This phase includes the transmission of application data (e.g., HTTP requests and responses and such) securely.

**Change Cipher Spec:** The client and server inform each other that subsequent messages will be encrypted.

- **Frame 225:** The "Change Cipher Spec" message is sent, indicating that the client will now use the newly negotiated cipher suite for encryption.

**Application Data:** Encrypted messages are exchanged between the client and server.

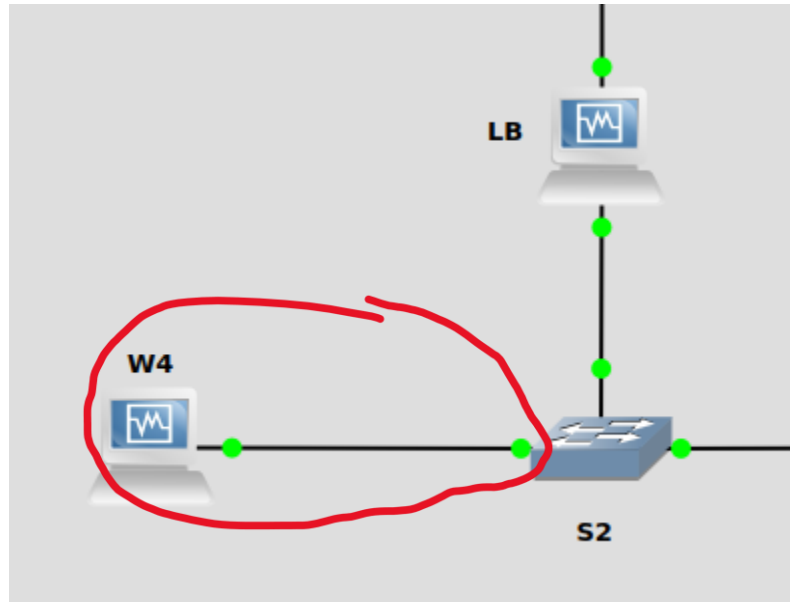
- **Frames 226-234:** These frames show encrypted application data being exchanged. This data represents secure communication between the client and server, including any HTTP requests and responses.

**TLS Close Notify (Optional):** After communication is complete, either party can send a "Close Notify" alert to gracefully close the TLS session.

- **Frame 234:** A FIN-ACK segment indicates the termination of the TCP session, which could be part of the TLS session closure.

## Sniffing and analyzing between the Load Balancer and HTTP Web Server (W4)

Start a WireShark capture on the highlighted node:



## WireShark Captures:

	Time	Source	Destination	Protocol	Length
61	38.945560	172.16.0.5	172.16.0.1	HTTP	3
62	38.945603	172.16.0.5	172.16.0.1	TCP	
63	38.945829	172.16.0.1	172.16.0.5	TCP	
64	38.946099	172.16.0.1	172.16.0.5	TCP	
65	38.946463	172.16.0.5	172.16.0.1	TCP	
66	39.114788	PcsCompu_d9:b3:51	Broadcast	ARP	
67	40.181481	PcsCompu_d9:b3:51	Broadcast	ARP	
68	41.097947	172.16.0.5	192.168.8.10	DNS	
69	41.098054	172.16.0.5	192.168.8.10	DNS	
70	41.273450	PcsCompu_d9:b3:51	Broadcast	ARP	
71	42.348006	PcsCompu_d9:b3:51	Broadcast	ARP	
72	43.361208	PcsCompu_d9:b3:51	Broadcast	ARP	
73	44.052306	PcsCompu_d6:ad:f8	PcsCompu_b7:c1:48	ARP	
74	44.053099	PcsCompu_b7:c1:48	PcsCompu_d6:ad:f8	ARP	
75	44.383103	PcsCompu_d9:b3:51	Broadcast	ARP	
76	44.550072	172.16.0.1	172.16.0.10	TCP	
77	44.550937	172.16.0.10	172.16.0.1	TCP	
78	44.551579	172.16.0.1	172.16.0.10	TCP	
79	44.551858	172.16.0.1	172.16.0.10	HTTP	2
80	44.552595	172.16.0.10	172.16.0.1	TCP	

Length	Info
387	HTTP/1.1 404 Not Found (text/html)
66	80 → 39412 [FIN, ACK] Seq=322 Ack=178 Win=65024 Len=0 TSval=3420385287 TSecr=440060644
66	39412 → 80 [ACK] Seq=178 Ack=322 Win=64128 Len=0 TSval=440060644 TSecr=3420385287
66	39412 → 80 [FIN, ACK] Seq=178 Ack=323 Win=64128 Len=0 TSval=440060644 TSecr=3420385287
66	80 → 39412 [ACK] Seq=323 Ack=179 Win=65024 Len=0 TSval=3420385288 TSecr=440060644
60	Who has 192.168.8.10? Tell 172.16.0.10
60	Who has 192.168.8.10? Tell 172.16.0.10
74	Standard query 0xb8f4 A ntp.ubuntu.com
74	Standard query 0x0951 AAAA ntp.ubuntu.com
60	Who has 192.168.8.10? Tell 172.16.0.10
60	Who has 192.168.8.10? Tell 172.16.0.10
60	Who has 192.168.8.10? Tell 172.16.0.10
60	Who has 172.16.0.5? Tell 172.16.0.1
60	172.16.0.5 is at 08:00:27:b7:c1:48
60	Who has 192.168.8.10? Tell 172.16.0.10
74	48050 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=962649628 TSecr=0
74	80 → 48050 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=528730851 TSecr=962649630
66	48050 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=962649630 TSecr=528730849
243	GET / HTTP/1.0
66	80 → 48050 [ACK] Seq=1 Ack=178 Win=65024 Len=0 TSval=528730851 TSecr=962649630
387	HTTP/1.1 404 Not Found (text/html)
66	80 → 48050 [FIN, ACK] Seq=322 Ack=178 Win=65024 Len=0 TSval=528730851 TSecr=962649630
66	48050 → 80 [ACK] Seq=178 Ack=322 Win=64128 Len=0 TSval=962649631 TSecr=528730851
66	48050 → 80 [FIN, ACK] Seq=178 Ack=323 Win=64128 Len=0 TSval=962649632 TSecr=528730851
66	80 → 48050 [ACK] Seq=323 Ack=179 Win=65024 Len=0 TSval=528730853 TSecr=962649632

## WireShark Capture W4 and LB Explanation

### 1. HTTP Request and Response

**Frames 61 to 66:** These frames involve an HTTP request made from the source (172.16.0.5) to the destination (172.16.0.1).

- **Frame 61:** The client (172.16.0.5) sends an HTTP request for a resource, but it receives a 404 Not Found error. This indicates that the requested resource does not exist on the server.
- **Frames 62-66:** These are part of the TCP connection closure process, where the FIN-ACK packets are exchanged to properly close the connection between the client and the server.

### 2. ARP Requests

**Frames 66 to 72:** These are ARP requests where devices are trying to discover the MAC address associated with the IP addresses.

- For example, "Who has 192.168.8.10?" is an ARP request asking for the MAC address of 192.168.8.10.
- This process allows devices on the network to map IP addresses to physical hardware addresses (MAC addresses).

### 3. DNS Queries

**Frames 68 and 69:** These show DNS queries being made by the source (172.16.0.5) to resolve hostnames to IP addresses.

- **Frame 68:** A standard DNS query is sent out to resolve a hostname (possibly related to NTP or another service).
- **Frame 69:** Another DNS query is sent out, likely related to resolving another service or domain.

#### 4. TCP 3-Way Handshake and HTTP Transaction

**Frames 73 to 84:** This segment involves another TCP 3-way handshake followed by an HTTP transaction.

- **Frames 73-75:** The TCP 3-way handshake occurs between the source (172.16.0.5) and the destination (172.16.0.10):
  - **Frame 73:** SYN packet is sent from the client to the server to initiate the connection.
  - **Frame 74:** The server responds with a SYN-ACK packet, acknowledging the connection request.
  - **Frame 75:** The client sends an ACK packet, completing the 3-way handshake and establishing the TCP connection.
- **Frame 77:** The client sends an HTTP GET request for a resource from the server at 172.16.0.10.
- **Frame 85:** The server responds with an HTTP 404 error, indicating that the requested resource was not found.

#### 5. Connection Closure

**Frames 86 to 91:** These frames show the termination of the TCP connection between the client and the server.

- **Frame 86:** The client initiates the termination with a FIN packet.
- **Frame 87:** The server acknowledges the FIN packet.
- **Frames 89-91:** The connection is fully closed after a few more ACK packets are exchanged.

## Alternative Option: Using the CA to sign your Certificate

### On the HTTPS server:

Generate a private key: **openssl genrsa -out server.key 2048**

Generate a CSR (follow the prompts): **openssl req -new -key server.key -out server.csr**

Transfer the CSR to the CA Server (use the right directories and IP address here):

**scp server.csr user@ca\_server\_ip:/home/user/csrs/**

### On the CA server:

Sign the certificate (put in the right paths indicated in the commands below):

**cd /home/user/csrs/**

**openssl ca -in server.csr -out server.crt -days 365 -cert /path/to/cacert.pem -keyfile /path/to/cakey.pem**

Transfer the signed certificate back to the HTTPS server:

**scp /home/user/csrs/server.crt user@https\_server\_ip:/home/user/**

### Back on the HTTPS Server

From here, you want to copy the signed certificate to a directory in which you will specify in your nginx configurations (refer to the HTTPS server nginx configuration guide section).



## Appendix

Walking through the different sections of the lab report, explaining each command that is used and the significance of the screenshots included.

### Switch Configuration

- **Commands:**
  - `sudo ip addr add 192.168.8.2/24 dev enp3s0f0`: Here, I assigned the IP address 192.168.8.2 to my network interface enp3s0f0. This was necessary to ensure my computer was correctly configured to communicate within the 192.168.8.0/24 network.
  - `sudo ip link set enp3s0f0 up`: After assigning the IP address, I used this command to bring up the network interface, making it active and ready for communication.
- **Screenshot:** The screenshot I included here shows the terminal output after executing these commands. It verifies that the IP address was successfully assigned, and the interface was activated.

## Connecting to My Table Switch and Router

- **Commands:**
  - **PuTTY Configuration:** I needed to connect to my table's switch and router via SSH, so I configured PuTTY with the necessary connection details. Specifically, I entered the IP address 142.55.114.23 and set the port to 22.
- **Screenshot:** This screenshot captures the PuTTY session setup screen, displaying the connection details. It serves as a confirmation that I correctly configured the session to connect to the switch and router at my assigned table.

## Table Switch Configurations

- **Commands:**
  - enable: To begin configuring the switch, I first entered privileged mode using the enable command.
  - configure terminal: I then accessed global configuration mode to start setting up the switch.
  - hostname S2: I set the switch's hostname to S2 to make it easily identifiable in the network.
  - vlan 10: I created VLAN 10 to segment the network traffic.
  - name VLAN10: I named this VLAN VLAN10 to keep the configurations organized and easily understandable.
  - interface gigabitEthernet 1/0/22: I selected port 22 for configuration.
  - switchport mode access: I configured port 22 to operate in access mode, associating it with a single VLAN.
  - switchport access vlan 10: I assigned port 22 to VLAN 10, ensuring that traffic on this port is tagged with VLAN 10.
  - spanning-tree portfast: I enabled PortFast on port 22 to reduce the time it takes for the port to begin forwarding packets.
  - no shutdown: Finally, I made sure the port was active by issuing the no shutdown command.
- **Screenshot:** The screenshot accompanying this section shows the switch's configuration as I did these commands. The highlighted lines correspond to the specific configurations I applied, such as the setup of VLAN 10 and the configuration of port 22.

## Switch Running Configurations

- **Commands:**
  - **show running-config:** I used this command to display the current configuration of the switch, which includes all the settings I applied in the previous steps.
- **Screenshot:** The screenshot here captures the output of the show running-config command. It specifically highlights the configurations related to VLAN 10 and the ports I configured. This screenshot serves as a record of the successful application of my configurations.

## Table Router Configurations

- **Commands:**

- cli: I started by entering the Juniper CLI to begin configuring the router.
- configure: I then entered configuration mode to make the necessary changes.
- set system host-name R2: I set the router's hostname to R2 to distinguish it from other devices.
- set system login user dino class super-user: I created a super-user account named dino for administrative access.
- set system login user dino authentication plain-text-password: I set a plain-text password for the dino user.
- set interfaces ge-0/0/13 vlan-tagging: I enabled VLAN tagging on the interface ge-0/0/13.
- set interfaces ge-0/0/13 unit 10 vlan-id 10 family inet address 100.64.144.2/24: I assigned the IP address 100.64.144.2/24 to VLAN 10 on the router.
- set interfaces ge-0/0/0 unit 0 family inet address 192.168.8.1/24: I configured the router's interface ge-0/0/0 with the IP address 192.168.8.1/24.
- set protocols ospf area 0.0.0.0 interface all: Finally, I enabled OSPF routing on all interfaces in area 0.0.0.0.
- commit and-quit: I saved and applied all configurations with this command.

- **Screenshot:** The screenshot associated with this section displays the router's configuration, showing the commands I executed and the results. This visual representation confirms that the router was configured correctly, with OSPF routing enabled and the proper IP addresses assigned.

## Router Running Configurations

- **Commands:**
  - show configuration: This command displays the router's current configuration in a detailed format.
  - show configuration | display set: I used this command to show the configuration in a set format, making it easier to understand the applied settings.
  - show route: This command lists the routing table, showing the routes known by the router.
  - show route terse: I used this command for a more concise view of the routing table.
- **Screenshot:** The screenshot included here captures the output of the show configuration command. It provides a clear overview of the router's configurations, including the IP addresses, VLANs, and OSPF settings. This verifies that the router was configured correctly.

## DNS Configurations

- **/etc/hosts:**
  - Contains mappings between IP addresses and hostnames, allows my system to resolve hostnames to IP addresses locally without querying an external DNS server.
- **/etc/hostname:**
  - Specifies the hostname of my machine, which is used to identify the system on the network.
- **Forward Lookup Zone File (/etc/bind/db.dino.org):**
  - Maps domain names to IP addresses, enabling the DNS server to resolve domain names to the correct IP addresses.
- **Reverse Lookup Zone File (/etc/bind/db.192.168.8):**
  - Maps IP addresses back to domain names, allowing reverse DNS lookups to identify the domain associated with an IP address.

## HTTP and HTTPS Nginx Configurations

- **HTTP Configuration File (/etc/nginx/sites-available/default):**
  - Configures the Nginx server to handle incoming HTTP requests, forwarding them to a backend server using the proxy\_pass directive.
- **HTTPS Configuration File (/etc/nginx/sites-available/default):**
  - Configures the Nginx server to handle HTTPS requests, using SSL/TLS certificates for encryption. It forwards secure requests to a backend server while ensuring the communication is encrypted.

## CA Certificate and Key Creation

### 1. Generate a Private Key:

- Command: `openssl genrsa -out cakey.pem 2048`
- **Explanation:** This command generates a 2048-bit RSA private key and saves it in a file called `cakey.pem`. This key is essential for signing certificates and is kept secure on the CA server.

### 2. Create a Self-Signed CA Certificate:

- Command: `openssl req -new -x509 -key cakey.pem -out cacert.pem -days 365`
- **Explanation:** This command generates a new X.509 certificate (`cacert.pem`) that is self-signed by the CA using the private key `cakey.pem`. The certificate is valid for 365 days. During this process, I was prompted to enter information such as the country, state, and organization name, which are included in the certificate's Distinguished Name (DN).

### 3. Filling Out Certificate Details:

- During the creation of the certificate, I filled out information such as:
  - **Country Name (2-letter code):** CA
  - **State or Province Name:** Ontario
  - **Locality Name:** Oakville
  - **Organization Name:** Sheridan College
  - **Common Name:** `ca.dino.org`
- **Explanation:** These details help with identifying my certificate and CA, ensuring that it is unique and traceable to the organization (Sheridan College).



## **Load Balancer Nginx Configuration**

- **Load Balancer Configuration File (/etc/nginx/sites-available/default):**
  - Configures Nginx as a load balancer, directing incoming traffic to a pool of backend servers. In this setup, I added SSL settings to keep the communication secure, and I made sure that the client's information is properly forwarded to the backend servers.

## References

Carapaica, Felix. *HTTP and HTTPS protocols*. July 2024. Accessed through Sheridan College SLATE Portal.

Carapaica, Felix. *Principles of Public Private Key Cryptography* Brief PowerPoint Presentation. Aug 2024. Accessed through Sheridan College SLATE Portal.

Carapaica, Felix. *RSA PKI Asymmetric Cryptography* Brief PowerPoint Presentation. Aug 2024. Accessed through Sheridan College SLATE Portal.

Carapaica, Felix. *Web Installation NGINX* Brief PowerPoint Presentation. July 2024. Accessed through Sheridan College SLATE Portal.

Nedelcu, C. (2010). *Nginx http server*. Packt Publishing Ltd..

networklessons.com. (Sep 26, 2017). *OpenSSL Certification Authority (CA) on Ubuntu Server*. <https://www.youtube.com/watch?v=oCl0gzLPPMI>

Viega, J., Messier, M., & Chandra, P. (2002). *Network security with openssl: cryptography for secure communications*. " O'Reilly Media, Inc."