

Portfolio 1

Janela Ann Valencia Pangan

8826102

INFO501

Sohrab Farooq

September 29, 2023

Table of Contents

Table of Contents	2
Lab 1 – CA Deployment	4
Description.....	4
Preparation	5
Service Diagram.....	6
Lab Kit Device Make and Model.....	6
Services Management IP Information.....	6
Palo Alto NG-Firewall Services Setup.....	7
Part 1 – Building the Initial Topology	8
Description.....	8
Network Topology	8
Defined Network IP Address and Zone Information.....	9
Palo Alto NG-Firewall Interface IP Information.....	9
Palo Alto NG-Firewall Defined Zones Information	9
Observations.....	10
Task 1: Configure the Palo Alto Firewall Hostname and Banner.....	10
Task 2: Define the Networks and Assign the IP Address to the Devices and Ports.....	10
Task 3: Add and Configure the Virtual Router	11
Task 4: Identify and Set the Network Zones	11
Screenshots.....	12
Task 1: Configure the Palo Alto Firewall Hostname and Banner.....	13
Task 2: Define the Networks and Assign the IP Address to the Devices and Ports.....	14
Task 3: Add and Configure the Virtual Router	15
Task 4: Identify and Set the Network Zones	18
Reflection.....	20
Part 2 – Build the Topology in the vSphere Environment	21
Description.....	21
Observations.....	21
Screenshots.....	21
Part 3 – Deploy your Certificate Authority.....	24
Description.....	24
Observations.....	24
Task 1: Generate a Private Key First to Generate CA Certificate	24
Task 2: Create a Certificate Authority (CA) Certificate using the Generated Private Key.....	25
Screenshots.....	27
Task 1: Generate a Private Key First to Generate CA Certificate	27
Task 2: Create a Certificate Authority (CA) Certificate using the Generated Private Key.....	28
Reflection.....	28
Discuss the Content of the Certificate Authority	28
Discuss the Components of the Certificate Authority	28
Explain the Architecture of Certificate Authority.....	29
Explain the Type of Certificate has been Deployed.....	30
Part 4 – Sign the Certificate Signing Request	30

Description.....	30
Observations.....	30
Task 1: Submit the CSR to the CA Server.....	30
Task 2: Sign the Certificate in the CA Server Prior to Uploading to the Palo Alto Firewall.	31
Task 3: Import the Signed Certificate to the Palo Alto Firewall.	33
Task 4: Validate the Status of the Import CA in the Palo Alto Firewall.....	33
Task 5: Save the Palo Alto Firewall's Running Configuration and Load the Base Configuration.	33
Screenshots.....	34
Task 1: Submit the CSR to the CA Server.....	34
Task 2: Sign the Certificate in the CA Server Prior to Uploading to the Palo Alto Firewall.	34
Task 3: Import the Signed Certificate to the Palo Alto Firewall.	35
Task 4: Validate the Status of the Import CA in the Palo Alto Firewall.....	35
Task 5: Save the Palo Alto Firewall's Running Configuration and Load the Base Configuration.	35
Reflection.....	36
Explain the Importance of the Certificate Signing Request.....	36
Explain the Importance of the Certificate Key	36
Explain the Difference Between the Private Key and Public Keys.....	36
References	40

Lab 1 – CA Deployment

Description

In this Portfolio 1 for laboratory activity in Certificate Authority (CA) deployment, we are mainly expected to deploy a certificate authority to understand the importance of having our digital certificate management centralized. To accomplish this, the Server and Client services using a Virtual Machine (VM) and the Palo Alto NG-Firewall are to be utilized.

Throughout the configuration and accomplishment of this Portfolio 1, we aim to acquire the following understandings and key takeaways:

- Able to install and configure the Palo Alto NG-Firewall with interface assignment, IP address configuration, zone segmentation, and virtual router configuration.
- Learn to install and deploy Certificate Authority Server.
- Know how to enroll the Networking components to the Certificate Authority.
- Able to know and understand the scripts to generate Certificates and Sign it by the Certificate Authority.
- Know how to import the correctly generated certificates to the respective devices, such as to the Palo Alto NG-Firewall.

Given these mentioned goals of Portfolio 1, we are expected to gain practical experience in configuring an additional layer of security to the built and designed network.

Preparation

The physical kits in Room 3G09 were utilized to accomplish all the requirements in this Lab 1 activity. Also, a virtual machine (VM) running in CentOS 7 has been created in the vSphere environment. Note that it is not applicable to connect the devices with a cable as this Lab 1 activity is based with the Server and Client services approach and only the required generated keys have been shared between the devices. Nonetheless, a network topology has been designed to properly understand the installation and configuration of Palo Alto NG-Firewall device, as well as knowing how to import the generated certificates to the respective devices.

As mentioned above, a VM running in CentOS 7 has been created in the vSphere environment which used the Management IP of 10.174.18.20, whilst the Palo Alto NG-Firewall Management IP has not been changed, 10.173.254.55.

Moreover, since I have used my assigned subnet, 10.174.18.0, to create the Server VM in vSphere the other six (6) networks of my built network topology have been configured with this subnet as well. But note that /28 has been used on all networks of my topology. Also, for any password prompt required to access a device, the "Secret55" password has been used.

With that, the only devices used in this Portfolio 1 are the following:

1. Server VM running with CentOS 7 in vSphere environment.
2. One Palo Alto NG-Firewall
3. Personal Laptop

Furthermore, prior proceeding to the configuration of the Palo Alto Firewall, I have run or load the script to reload or change the current configuration to the default settings or base configuration.

Script or command used:

```
load config from Base-Config
```

Service Diagram

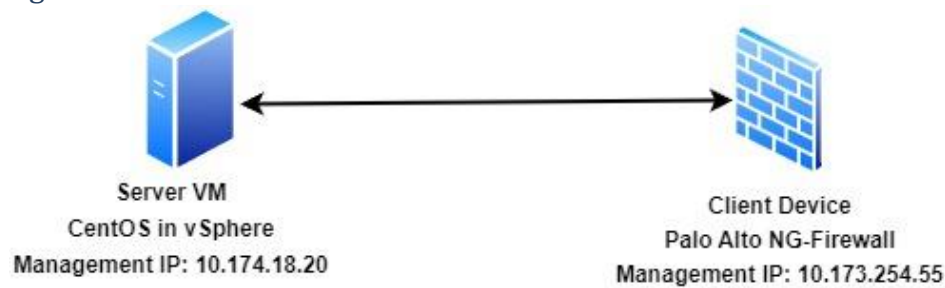


Figure 1-1. This displays that the CentOS VM will serve as the Server VM, while the NG-Firewall Palo Alto will serve as the Client device.

Lab Kit Device Make and Model

Device make and model	
Access Server	Room 3G09 Kit#63
	Kit IP: 10.175.226.162
Switch 1 (Layer 2 Switch)	2960 Cisco Switch
Switch 2 (Layer 3 Switch)	3750 Cisco Switch
Router 1	2811 Cisco Router
Router 2	2811 Cisco Router
Virtual Machines	Kit VM IP Range: 10.175.225.88-92
	Kit VM01: 10.175.224.88
	Kit VM02: 10.175.224.90
	Kit VM03: 10.175.224.91
	Kit VM04: 10.175.224.92

Table 1-1. This displays the Lab Kit information that has been used to configure the requirements in this portfolio.

Services Management IP Information

Management IP	
Palo Alto NG-Firewall	10.173.254.55
CentOS VM in vSphere	10.174.18.20

Table 1-2. This displays the Management IP that has been used to configure both services, Server VM and NG-Firewall Palo Alto.

Palo Alto NG-Firewall Services Setup

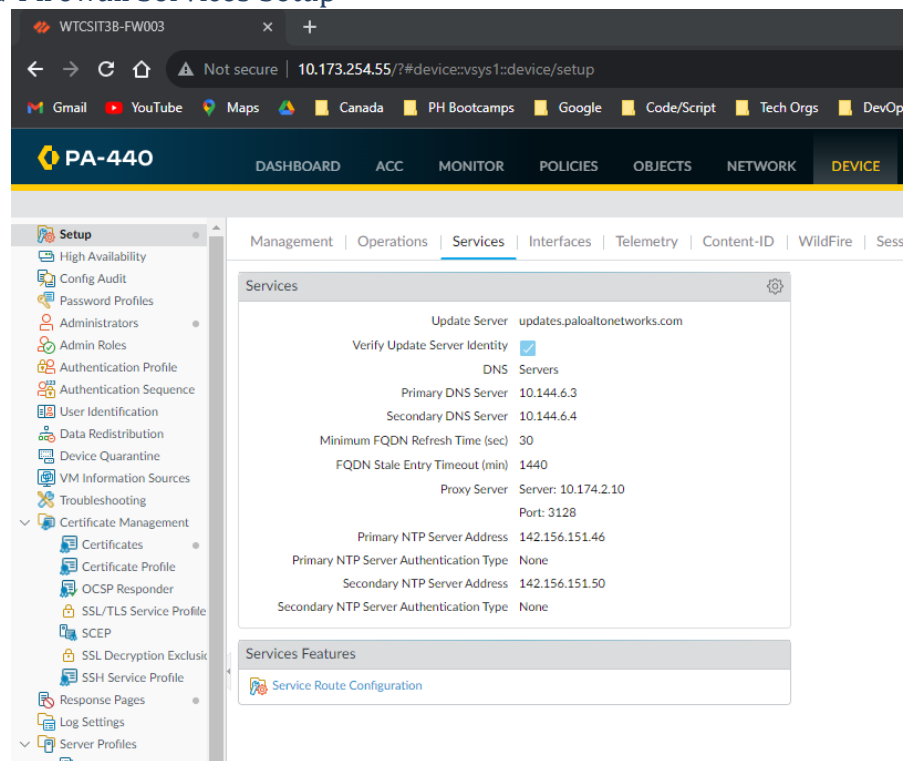


Figure 1-2. This displays that the Palo Alto firewall device is set to the default services setup.

Part 1 – Building the Initial Topology

Description

In this first part of the Lab 1 activity, we are to build and configure our own designed network topology that follows the required lab schematics. The defined IP addresses and naming conventions were observed as instructed with my initials and last four (4) digits of my student IP set as a unique identifier to each networking device.

The Palo Alto NG-Firewall interfaces have been configured with IP address as per the defined networks and segmented zones. Thence, the ethernet1/1 port was used for the Internet network that belongs to the Internet zone. The ethernet1/1 port was used for the Internet network because firewalls usually set or reserved the first port for the ISP connectivity or considered as the LAN port. Thence, followed by ethernet1/2 port for Inside network in the Inside or Trusted zone, ethernet1/3 port for DMZ network of the DMZ or Semi-Trust zone, and ethernet1/4 port for Guest network of the unsecured or Guest zone. On other firewall devices, the interface or port 4 is being used for the secondary ISP or backup wireless connection such as to an LTE device or cellular stick.

Network Topology

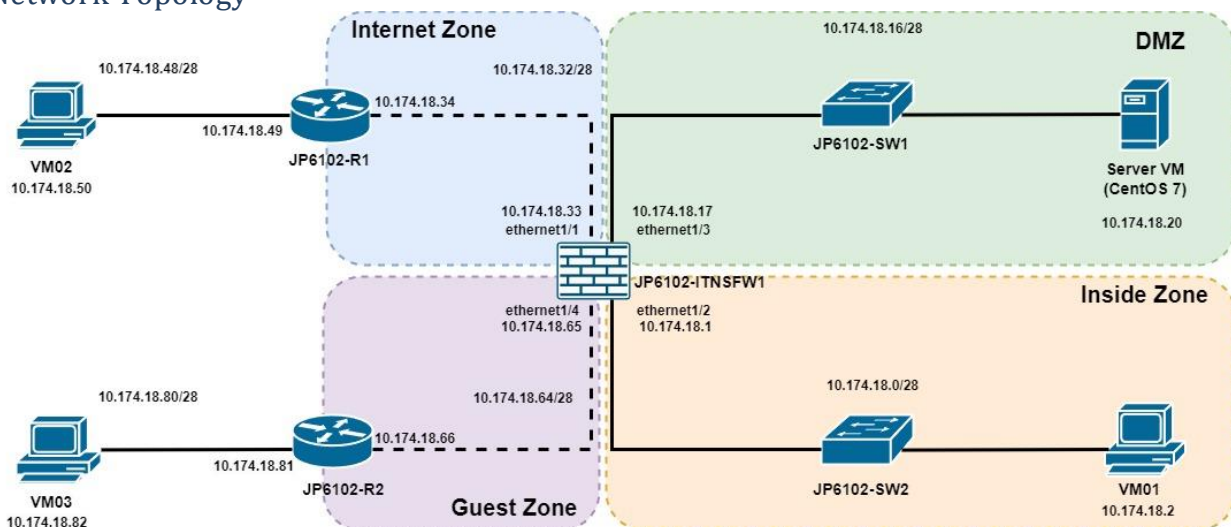


Figure 1-3. This displays the designed network topology used for this Portfolio 1.

Defined Network IP Address and Zone Information

Network #	Network 1	Network 2	Network 3
Zone	Inside Zone	DMZ Zone	Internet Zone
IP Address	10.174.18.0/28	10.174.18.16/28	10.174.18.32/28
Subnet Mask	255.255.255.240	255.255.255.240	255.255.255.240
Usable IP Address Range	10.174.18.2 - 10.174.18.14	10.174.18.18 - 10.174.18.30	10.174.18.34 - 10.174.18.46
Default Gateway	10.174.18.1	10.174.18.17	10.174.18.33

Table 1-3. This displays the Networks 1 to 3 that were used in building the Network Topology shown in Figure 1-3.

Network #	Network 4	Network 5	Network 6
Zone	Internet Zone	Guest Zone	Guest Zone
IP Address	10.174.18.48/28	10.174.18.64/28	10.174.18.80/28
Subnet Mask	255.255.255.240	255.255.255.240	255.255.255.240
Usable IP Address Range	10.174.18.34 - 10.174.18.46	10.174.18.66 - 10.174.18.78	10.174.18.82 - 10.174.18.94
Default Gateway	10.174.18.49	10.174.18.65	10.174.18.81

Table 1-4. This displays the Networks 4 to 6 that were used in building the Network Topology shown in Figure 1-3.

Palo Alto NG-Firewall Interface IP Information

Palo Alto NG-Firewall Interface IP Assignment	
ethernet1/1	10.174.18.33
ethernet1/2	10.174.18.1
ethernet1/3	10.174.18.17
ethernet1/4	10.174.18.65

Table 1-5. This displays the Palo Alto NG-Firewall IP assignment on each interface.

Palo Alto NG-Firewall Defined Zones Information

Palo Alto NG-Firewall Interface Assignment to Each Segmented Zone	
ethernet1/1	Internet Zone
ethernet1/2	Inside Zone
ethernet1/3	DMZ Zone
ethernet1/4	Guest Zone

Table 1-6. This displays the Palo Alto NG-Firewall interface assigned to each segmented zone.

Observations

This section includes the detailed method of procedure that was followed to deliver the expected output or results in this Part 1 of the Portfolio.

Task 1: Configure the Palo Alto Firewall Hostname and Banner

In the web page browser, I have typed in `https://10.173.254.55.com` to access my Palo Alto Firewall.

To change the hostname and Login banner, the following steps are to be followed:

Step 1: Along with the **Device Tab**, proceed to click the **Setup** option, then within it, click the **Management Tab**. In the **General Settings** dashboard, click the **gear** button to change the hostname and Login banner.

Step 2: With that, the Palo Alto Firewall hostname was changed to `JP6102-ITNSFW1`, while the Login banner was changed to the below quote:

Login banner configured:

Welcome to the NG-Firewall of JP6102!

"Cybersecurity is not just about protecting data; it's about protecting people." - Bruce Schneier

Step 3: Click the commit button to save the changes.

Task 2: Define the Networks and Assign the IP Address to the Devices and Ports

With the Network Topology I have designed, a total of six (6) networks were defined with all having a /28 subnet. One of each of the four defined port or interfaces connecting the Palo Alto Firewall that are related to each defined zone, and another one network for the two routers used on the Internet and Guest zone.

The Palo Alto Firewall has eight (8) ports or interfaces to use for configuring the designed network topology, and the Port 1 to 4 were individually defined to their respective assigned network and zone.

To add the IP address to each interface of the Palo Alto Firewall, proceed to follow the steps below:

Step 1: In the **Network Tab**, click the **Interfaces** option.

Step 2: Under the **Ethernet** field, choose the corresponding ethernet interface and set the defined IP address information by clicking the ethernet interface, eg. **ethetner1/1**.

Step 3: With that, the **ethernet1/1**, **ethernet1/2**, **ethernet1/3**, and **ethernet1/4** have all been configured with their assigned IP address based on the designed network topology.

Step 4: Once done, the **commit** button was clicked to save the configuration that has been made.

Task 3: Add and Configure the Virtual Router

A virtual router was added and configured to enable communication or routing on all the identified networks or zones in the designed network topology. However, because the main requirement of this Portfolio 1 had focused on the Certificate Authority, no further configuration was made to allow the traffic to be transferred or exchanged within all the defined networks.

To add a virtual router, the following steps are to be followed:

Step 1: In the **Network Tab**, click on the **Virtual Router** option, then click **Add**. Note that if an existing virtual router named **default** is found, proceed to edit that option with the defined virtual router of this designed network topology.

Step 2: In the Router Settings of the Virtual Router wizard, type in **JP6102-VR** as the name for the virtual router.

Step 3: Then, add all the configured interfaces of the Palo Alto Firewall.

Step 4: In the **Static Routes** option, and along the **IPv4** field, click **Add**.

Step 5: Name the **Static Routes** setting as **Firewall Default Gateway**, enter **0.0.0.0/0** as the destination, and select **ethernet1/1** as the interface.

Step 6: Continue in the **IPv4** setting by determining the **Next Hop** address, which is **10.174.18.34** based on my designed network topology.

Step 7: Review all the configuration made, and once verified to be completed as required, click **OK**, then proceed to **commit** the changes made.

Task 4: Identify and Set the Network Zones

Next and final task is to configure the defined network segmentation based on the required Zones of this portfolio. These four defined network zones are the **Internet**, **Inside**, **DMZ**, and **Guest**.

To configure the network zones, proceed with the following steps:

Step 1: In the **Network Tab**, click the **Zone** option, and click **Add**.

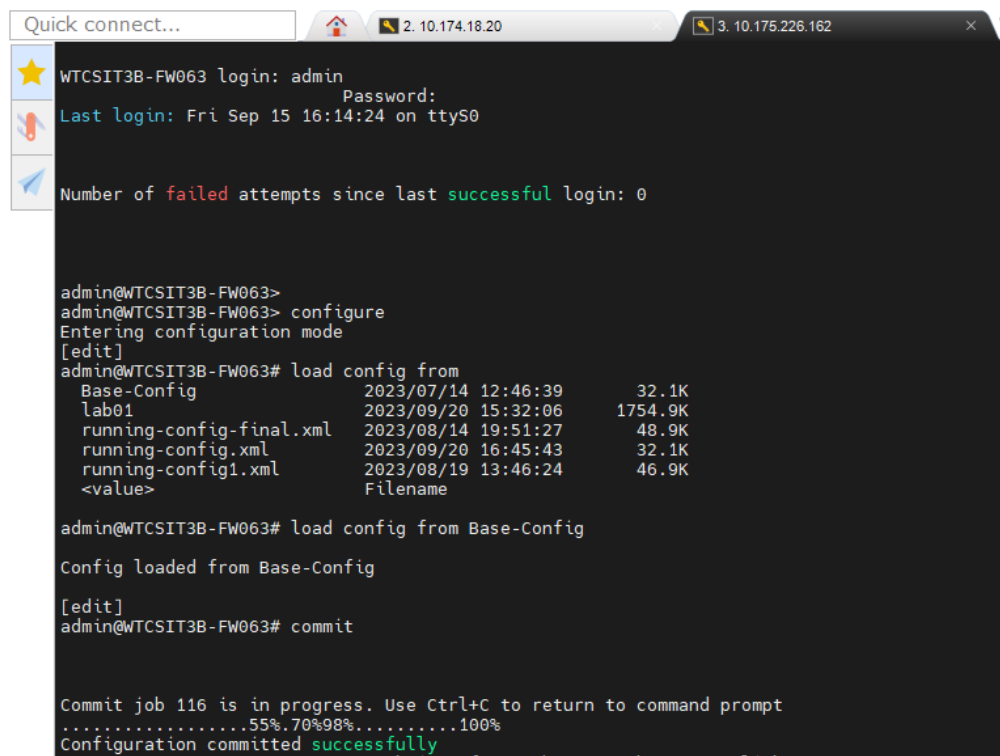
Step 2: In the **Zone wizard window**, type the name of the zone such as **Internet**, select **Layer 3** for the **Type** field.

Step 3: Along the Interfaces section, click **Add**, and select the port or interface assigned to the Internet Zone, eg. **ethernet1/1**.

Step 4: There are still other fields that can be changed, eg. **User Identification ACL**, however, for this Portfolio, those fields are to be left unchanged and in their default settings.

Step 5: Once verified done, click **OK**, then proceed to **commit** the changes made.

Screenshots



```
Quick connect... 2. 10.174.18.20 3. 10.175.226.162
WTCSIT3B-FW063 login: admin
Password:
Last login: Fri Sep 15 16:14:24 on ttyS0
Number of failed attempts since last successful login: 0

admin@WTCSIT3B-FW063>
admin@WTCSIT3B-FW063> configure
Entering configuration mode
[edit]
admin@WTCSIT3B-FW063# load config from
Base-Config          2023/07/14 12:46:39      32.1K
lab01                 2023/09/20 15:32:06     1754.9K
running-config-final.xml 2023/08/14 19:51:27      48.9K
running-config.xml      2023/09/20 16:45:43      32.1K
running-config1.xml     2023/08/19 13:46:24      46.9K
<value>               Filename

admin@WTCSIT3B-FW063# load config from Base-Config
Config loaded from Base-Config

[edit]
admin@WTCSIT3B-FW063# commit

Commit job 116 is in progress. Use Ctrl+C to return to command prompt
.....55%.70%98%.....100%
Configuration committed successfully
```

Figure 1-4. This displays that the command to load the default settings has been run prior to configuring the requirements in this Portfolio.

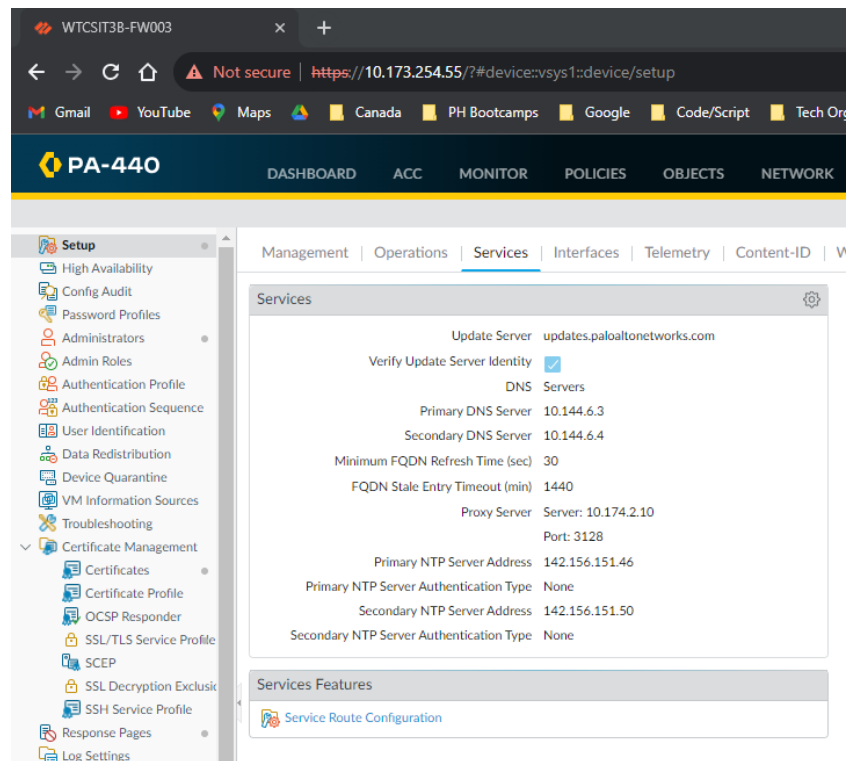


Figure 1-5. This displays that the Palo Alto Firewall is in the default setup services and no changes have been made, including the Management IP.

Task 1: Configure the Palo Alto Firewall Hostname and Banner

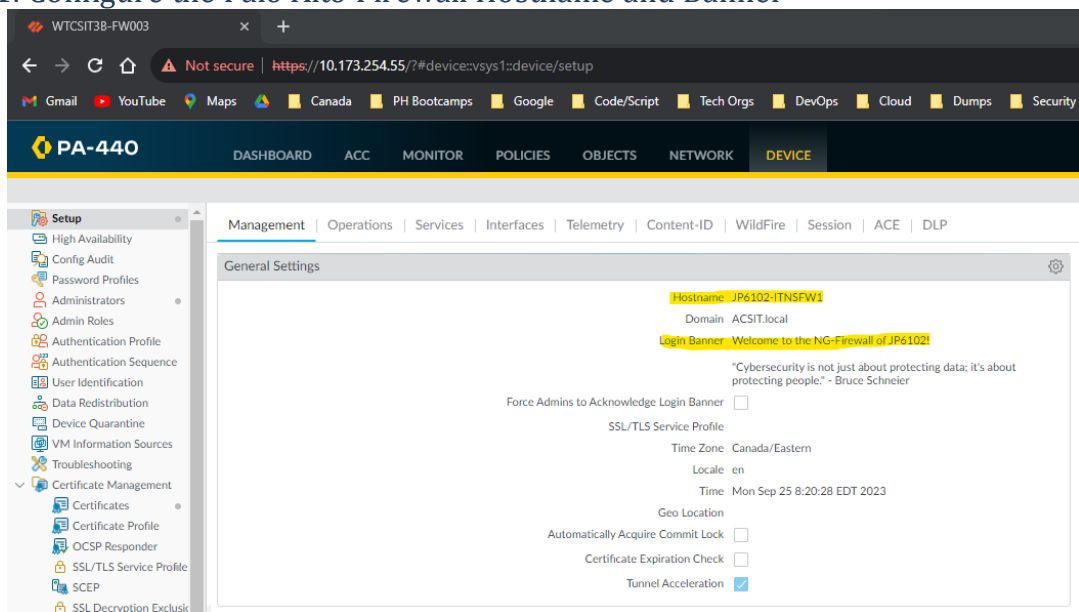


Figure 1-6. This displays that both the Hostname and Login Banner were changed for the Palo Alto Firewall.

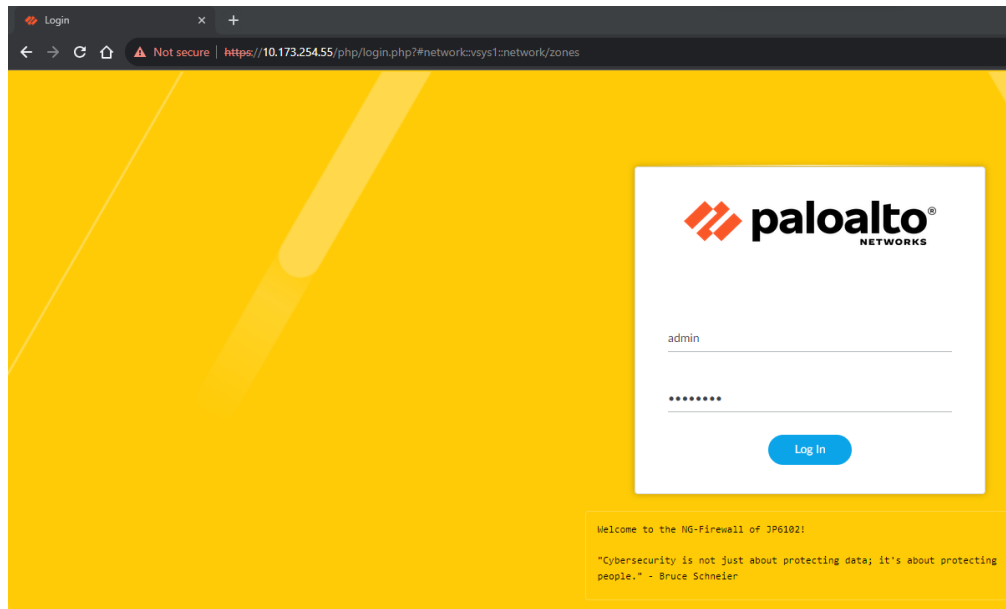


Figure 1-7. This displays that the modified Login Banner was successfully generated to the Palo Alto Firewall.

Task 2: Define the Networks and Assign the IP Address to the Devices and Ports

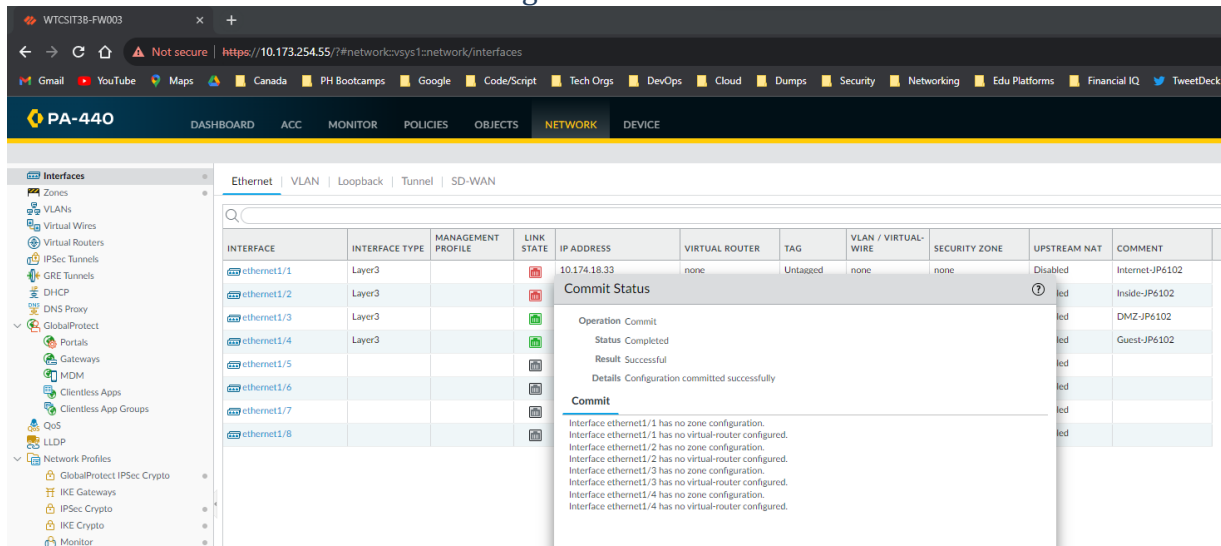


Figure 1-8. This displays that all the defined ports or interfaces, ethernet1/1-4, have been successfully configured based on the defined information of the designed network topology.

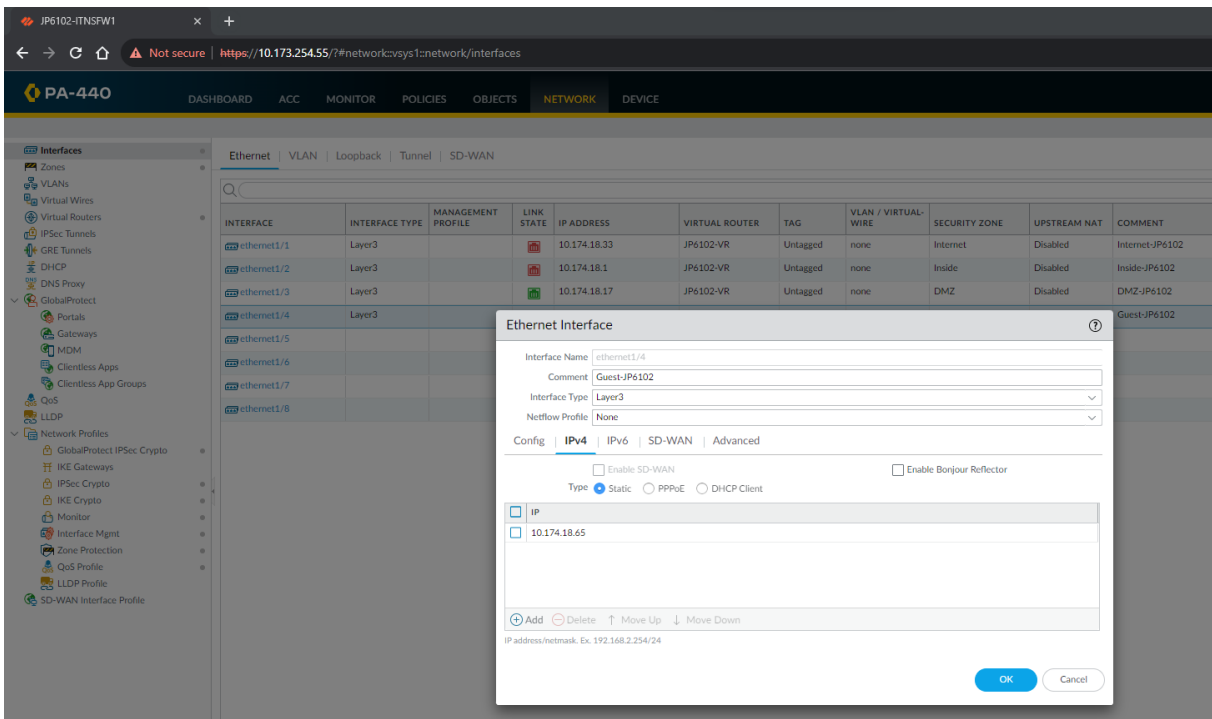


Figure 1-9. This displays the Ethernet Interface wizard window when configuring the ethernet1/4, facing the Guest Zone, of the designed network topology. Also, this shows that the IPv4 Static IP Address has been assigned to this interface.

Task 3: Add and Configure the Virtual Router

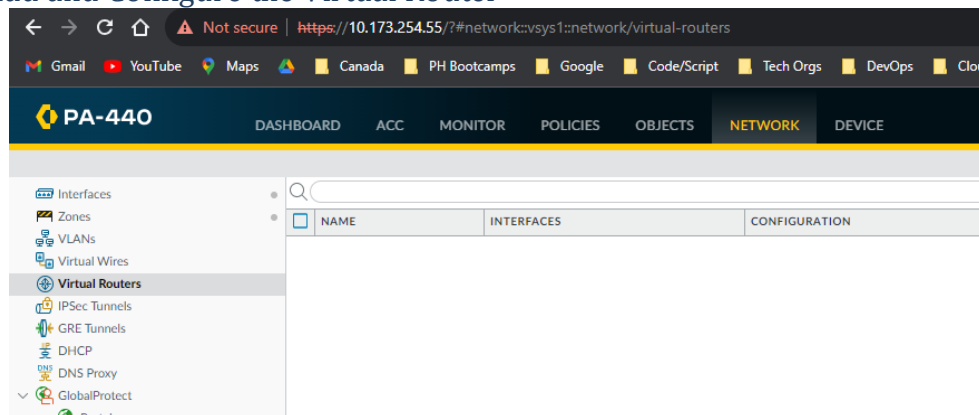


Figure 1-10. This displays that no existing virtual router configuration was initially found.

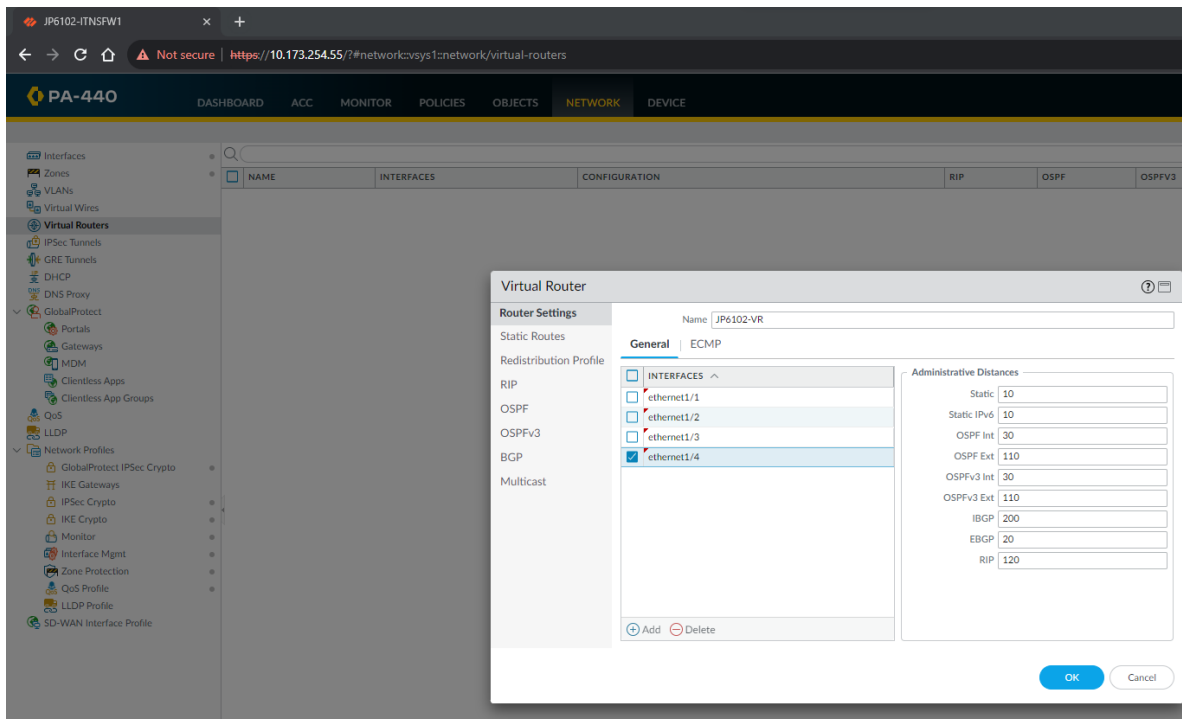


Figure 1-11. This displays the Virtual Router window where the Router Settings have been changed.

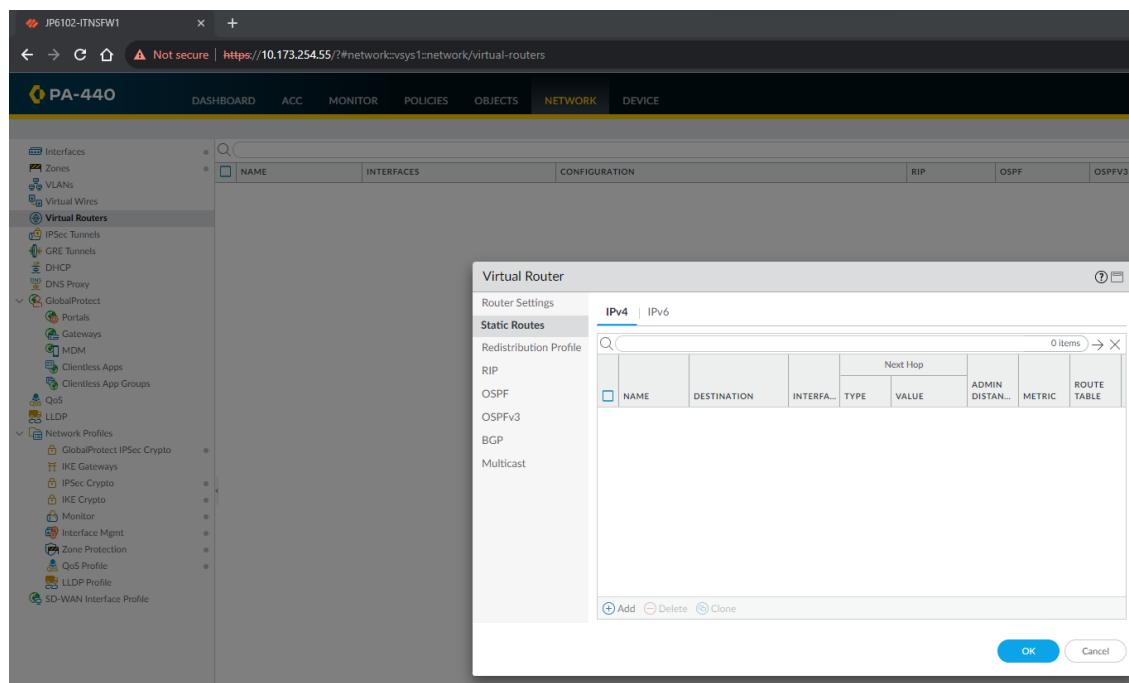


Figure 1-12. This displays the Virtual Router window where no existing Static Routes information was found.

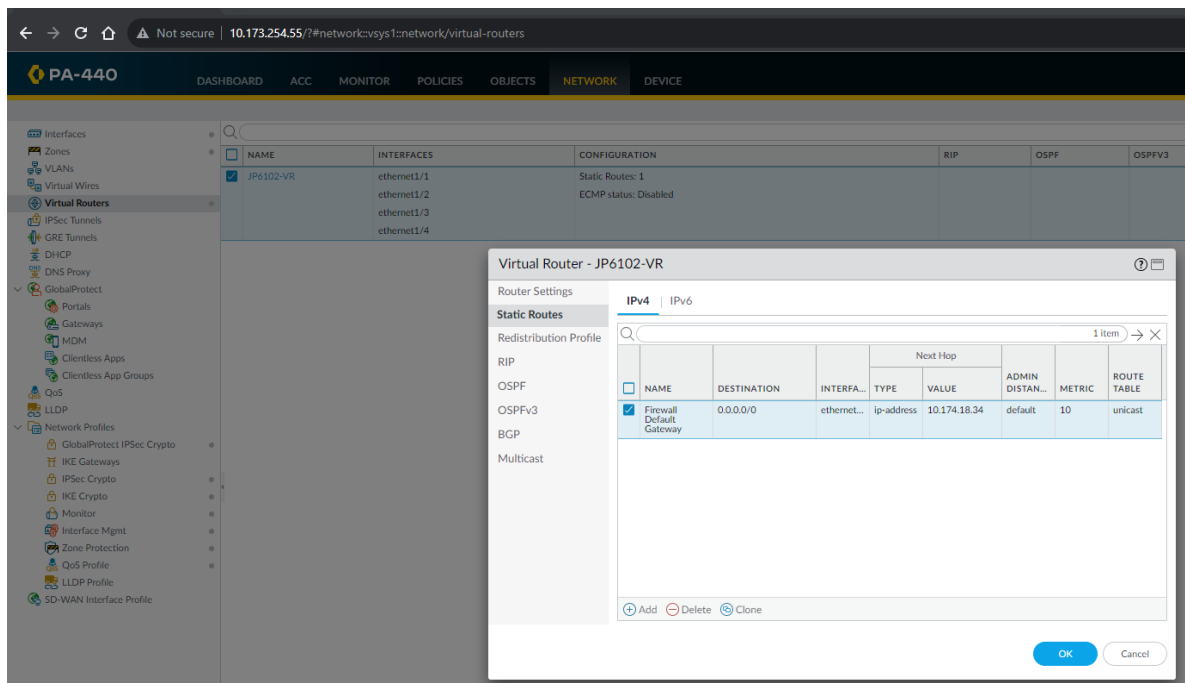


Figure 1-13. This displays the Virtual Router window where the Static Routes information has been changed.

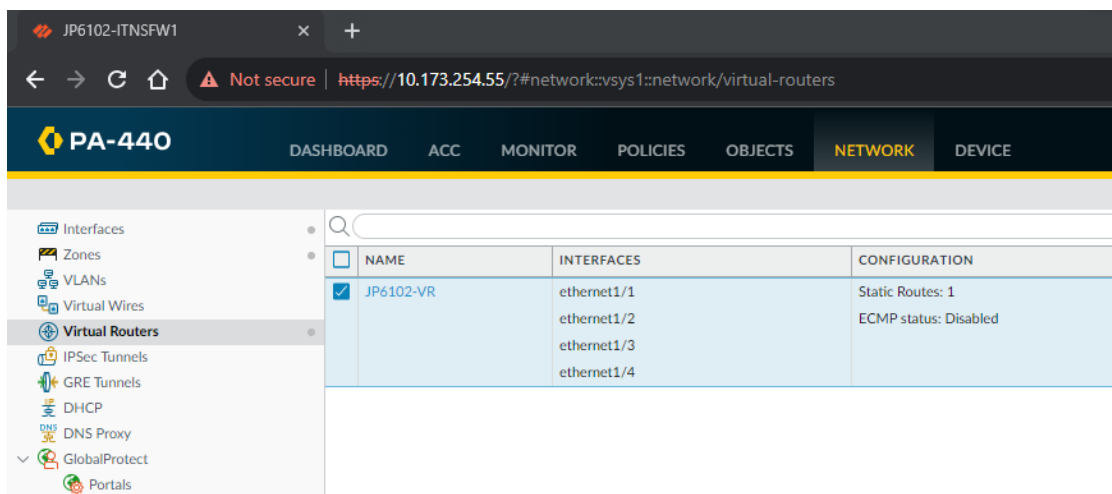


Figure 1-14. This displays the configured Virtual Router settings for the Palo Alto Firewall.

Task 4: Identify and Set the Network Zones

PA-440

DASHBOARD

AGG

MONITOR

POLICIES

OBJECTS

NETWORK

DEVICE

Current

2 Items

→

Interfaces

Zones

VLANs

Virtual Wires

Virtual Routers

IPsec Tunnels

GRE Tunnels

DHCP

DNS Proxy

GlobalProtect

NAME	TYPE	INTERFACES / VIRTUAL SYSTEMS	ZONE PROTECTION PROFILE	PACKET BUFFER PROTECTION	LOG SETTING	User-ID		Device-ID			
						ENABLED	INCLUDED NETWORKS	EXCLUDED NETWORKS	ENABLED	INCLUDED NETWORKS	EXCLUDED NETWORKS
<input type="checkbox"/> Inside	layer3			<input checked="" type="checkbox"/>		<input type="checkbox"/>	any	none	<input type="checkbox"/>	any	none
<input checked="" type="checkbox"/> Internet	layer3			<input checked="" type="checkbox"/>		<input type="checkbox"/>	any	none	<input type="checkbox"/>	any	none

Figure 1-15. This displays that two zones were existing along the Zone option, even though the base configuration was loaded.

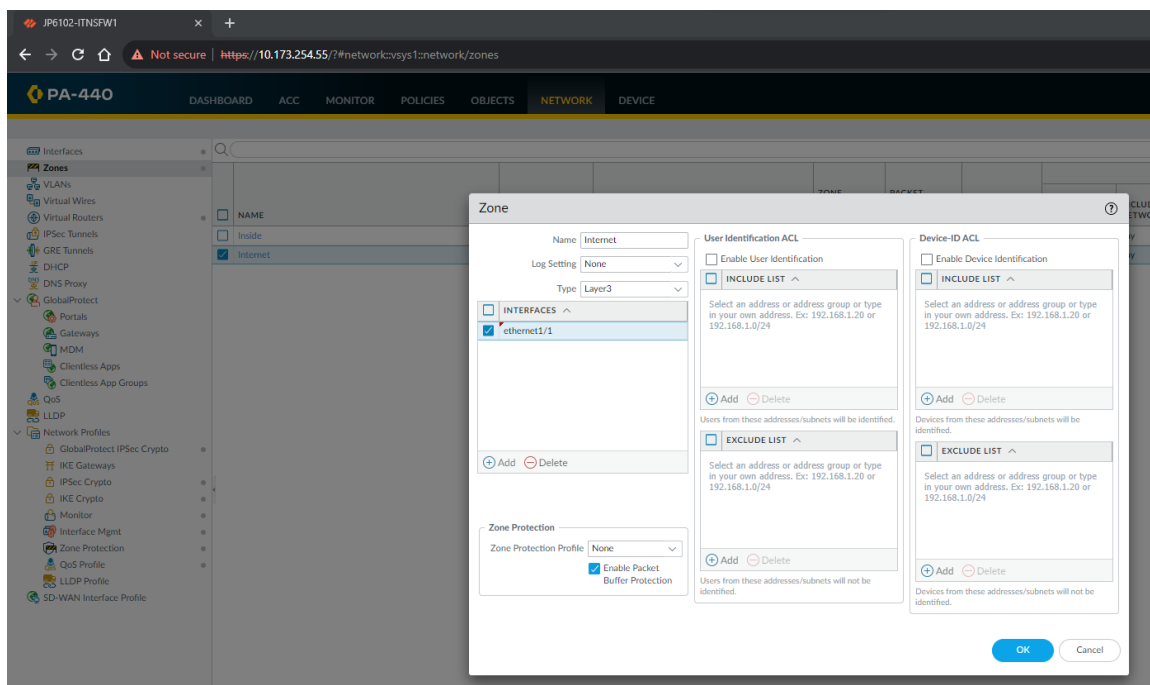
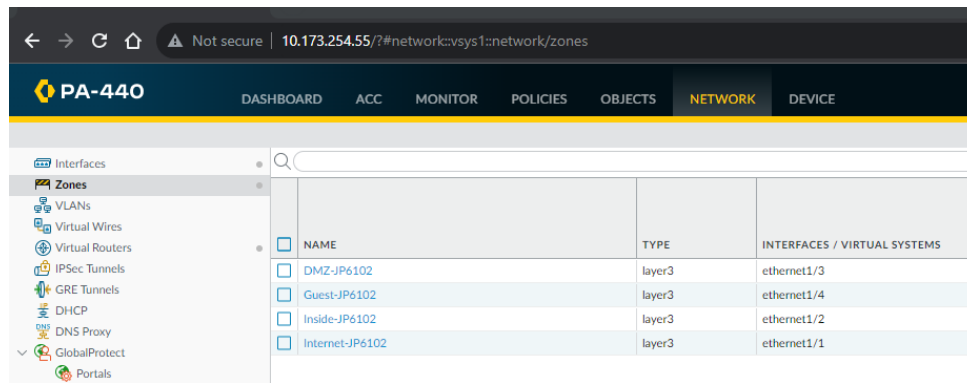


Figure 1-16. This displays the configuration made for the Internet Zone.



The screenshot shows the PA-440 web interface with the 'NETWORK' tab selected. The left sidebar lists various network configuration options, with 'Zones' highlighted. The main content area displays a table of configured zones.

NAME	TYPE	INTERFACES / VIRTUAL SYSTEMS
DMZ-JP6102	layer3	ethernet1/3
Guest-JP6102	layer3	ethernet1/4
Inside-JP6102	layer3	ethernet1/2
Internet-JP6102	layer3	ethernet1/1

Figure 1-17. This displays all four of the configured zones that were defined in the designed network topology.

Reflection

The designed network topology for this Portfolio 1 was based on the Lab 1 – CA Deployment activity, which had only defined three zones. Thus, with the available devices in the physical kit, I have added the Guest Zone to be in ethernet1/4 interface along with connecting the second Cisco Router to that interface. This means that Router 2 in the Guest Zone will serve as the gateway to provide traffic flow between the Guest Zone and other internal network zones.

For each interface of the Palo Alto Firewall, a different and unique network has been assigned to them. This strategy was made to follow the network segmentation method and control the traffic flow between the networks. Moreover, I have known that to contain any possibility of security incidents, it is a best practice to separate the different parts of the network such as the DMZ (or the Demilitarized Zone), Internet Network, Guest Network, Inside Network, etc.

Moreover, for this Portfolio 1, a Server VM running with CentOS in a vSphere environment. With that, I added the Server VM the Switch 1 connected to the ethernet1/3 of the Palo Alto Firewall that was assigned or defined as the DMZ Network. In my previous work experience, the servers are often connected to the DMZ. Given that the DMZ is also referred as a semi-trusted zone, connecting the Server VM that host services, such as generating a Certificate authority, to this zone allows users to access the needed services while ensuring that a stricter access control and policies are being implemented. Moreover, by also adding a switch in the DMZ to be connected to the Server VM, simplified network management and troubleshooting has been designed.

Lastly, aside from configuring the Palo Alto Firewall's interfaces with their defined networks and zones, a Virtual Router was also added. Based on what I have experienced with other firewall vendors, configuring a Virtual Router in a firewall will allow the firewall to execute routing processes such as determining the path for outbound or inbound traffic between different network zones. With an added Virtual Router configured, the firewall serves as a gateway which routes traffic based on the configured protocol and routing table. However, as this Portfolio 1 mainly focused on Certificate Authority, I have only configured the default route and next hop for the static routing process.

Part 2 – Build the Topology in the vSphere Environment

Description

This part of Portfolio 1 is an essential task for the Certificate Authority to be deployed and generated. A virtual machine running with CentOS in vSphere environment is expected to be created and installed. Upon installation of the VM, all its essential packages are to be updated and ensure that the OpenSSL has been installed in it.

Observations

In installing the CentOS VM, I have followed the same process of installing a Windows Server 2019 from our previous term. The only difference is that instead of choosing Windows as one of the options of the installation wizard, I have chosen Linux.

Nonetheless, all the other information such as time zone, IP address, and password were changed based on the requirement of this Portfolio 1.

Additionally, the remote SSH access in the CentOS VM was enabled. Thus, configuring the CentOS VM was done using the MobaXterm terminal emulator and remote desktop software.

Screenshots

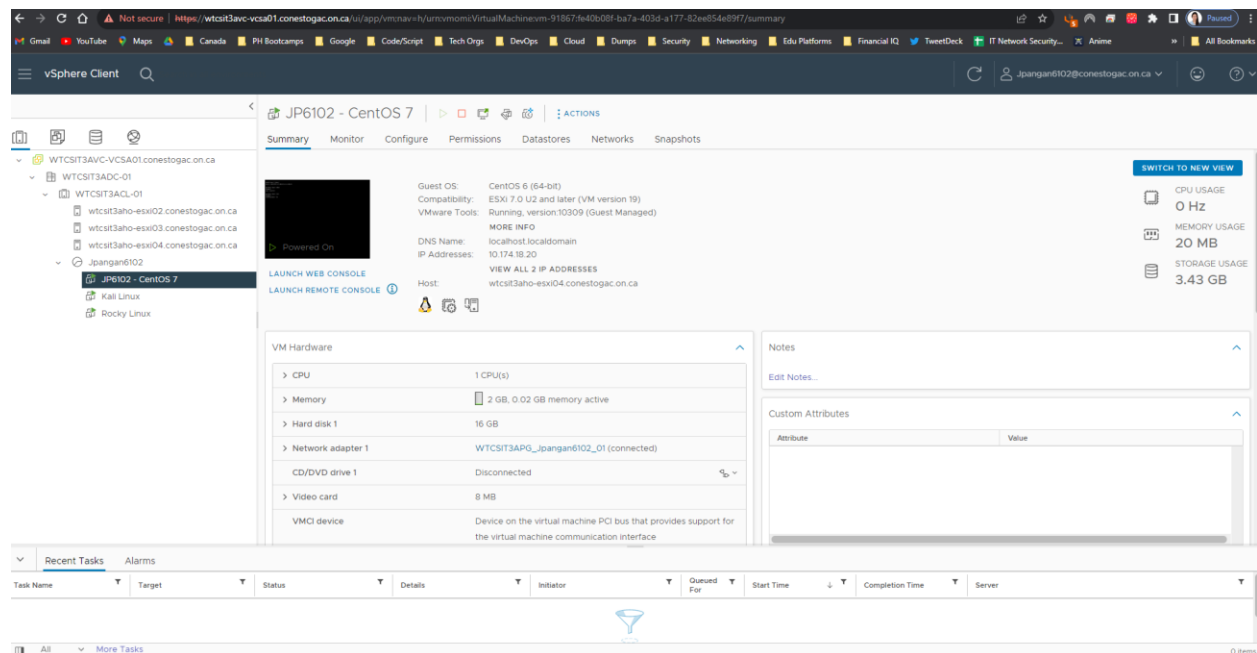
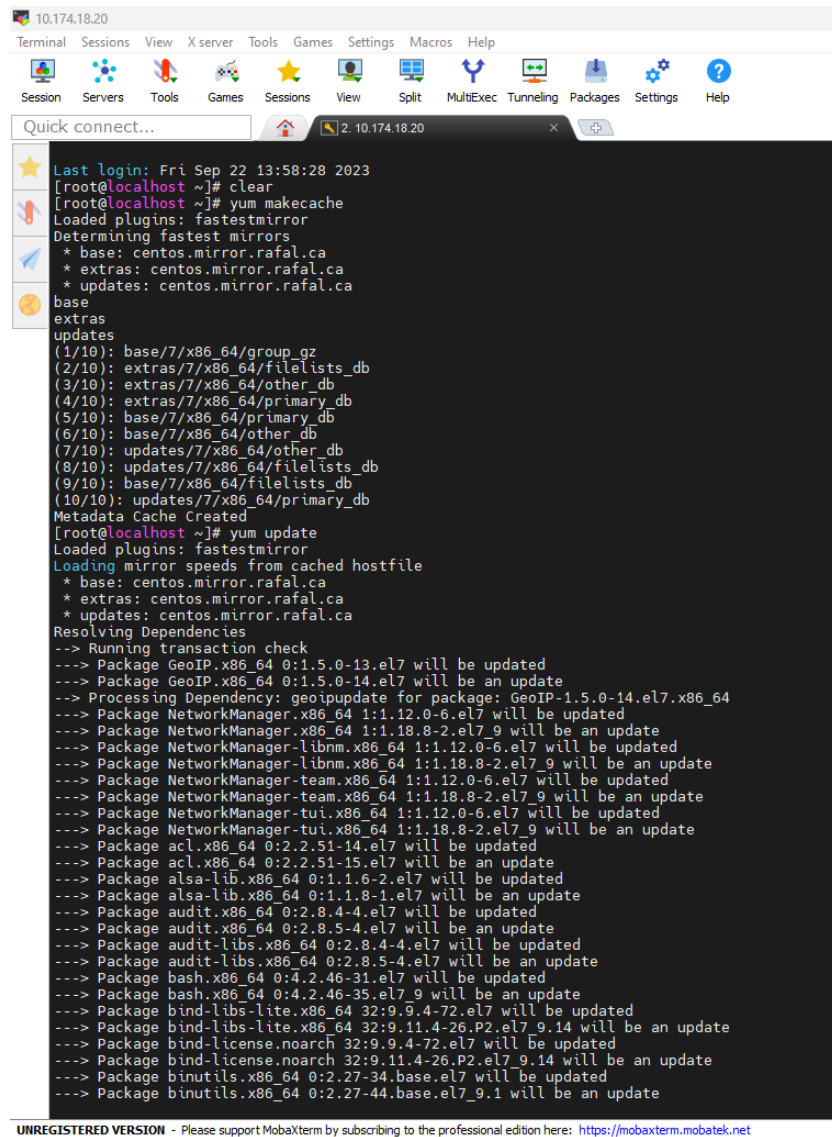


Figure 1-18. This displays that I have installed the virtual machine running with CentOS in the vSphere environment.



```

10.174.18.20
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect... 2. 10.174.18.20
Last login: Fri Sep 22 13:58:28 2023
[root@localhost ~]# clear
[root@localhost ~]# yum makecache
Loaded plugins: fastestmirror
Determining fastest mirrors
 * base: centos.mirror.rafal.ca
 * extras: centos.mirror.rafal.ca
 * updates: centos.mirror.rafal.ca
base
extras
updates
(1/10): base/7/x86_64/group_gz
(2/10): extras/7/x86_64/filelists_db
(3/10): extras/7/x86_64/other_db
(4/10): extras/7/x86_64/primary_db
(5/10): base/7/x86_64/primary_db
(6/10): base/7/x86_64/other_db
(7/10): updates/7/x86_64/other_db
(8/10): updates/7/x86_64/filelists_db
(9/10): base/7/x86_64/filelists_db
(10/10): updates/7/x86_64/primary_db
Metadata Cache Created
[root@localhost ~]# yum update
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos.mirror.rafal.ca
 * extras: centos.mirror.rafal.ca
 * updates: centos.mirror.rafal.ca
Resolving Dependencies
--> Running transaction check
--> Package GeoIP.x86_64 0:1.5.0-13.el7 will be updated
--> Package GeoIP.x86_64 0:1.5.0-14.el7 will be an update
--> Processing Dependency: geoupdate for package: GeoIP-1.5.0-14.el7.x86_64
--> Package NetworkManager.x86_64 1:1.12.0-6.el7 will be updated
--> Package NetworkManager.x86_64 1:1.18.8-2.el7_9 will be an update
--> Package NetworkManager-libnm.x86_64 1:1.12.0-6.el7 will be updated
--> Package NetworkManager-libnm.x86_64 1:1.18.8-2.el7_9 will be an update
--> Package NetworkManager-team.x86_64 1:1.12.0-6.el7 will be updated
--> Package NetworkManager-team.x86_64 1:1.18.8-2.el7_9 will be an update
--> Package NetworkManager-tui.x86_64 1:1.12.0-6.el7 will be updated
--> Package NetworkManager-tui.x86_64 1:1.18.8-2.el7_9 will be an update
--> Package acl.x86_64 0:2.2.51-14.el7 will be updated
--> Package acl.x86_64 0:2.2.51-15.el7 will be an update
--> Package alsa-lib.x86_64 0:1.1.6-2.el7 will be updated
--> Package alsa-lib.x86_64 0:1.1.8-1.el7 will be an update
--> Package audit.x86_64 0:2.8.4-4.el7 will be updated
--> Package audit.x86_64 0:2.8.5-4.el7 will be an update
--> Package audit-libs.x86_64 0:2.8.4-4.el7 will be updated
--> Package audit-libs.x86_64 0:2.8.5-4.el7 will be an update
--> Package bash.x86_64 0:4.2.46-31.el7 will be updated
--> Package bash.x86_64 0:4.2.46-35.el7_9 will be an update
--> Package bind-libs-lite.x86_64 32:9.9.4-72.el7 will be updated
--> Package bind-libs-lite.x86_64 32:9.11.4-26.P2.el7_9.14 will be an update
--> Package bind-license.noarch 32:9.9.4-72.el7 will be updated
--> Package bind-license.noarch 32:9.11.4-26.P2.el7_9.14 will be an update
--> Package binutils.x86_64 0:2.27-34.base.el7 will be updated
--> Package binutils.x86_64 0:2.27-44.base.el7_9.1 will be an update

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Figure 1-19. This displays that I was able to access the virtual machine with SSH connectivity. This also shows that I have updated all the necessary packages of the CentOS VM.

```

10.174.18.20
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages
Quick connect...
libblkid.x86_64 0:2.23.2-65.el7_9.1
libcurl.x86_64 0:7.29.0-59.el7_9.1
libffi.x86_64 0:3.0.13-19.el7
libmount.x86_64 0:2.23.2-65.el7_9.1
libselinux.x86_64 0:2.5-15.el7
libss.x86_64 0:1.42.9-19.el7
libuuid.x86_64 0:2.23.2-65.el7_9.1
logrotate.x86_64 0:3.8.6-19.el7
lz4.x86_64 0:1.8.3-1.el7
nss.x86_64 0:4.34.0-3.1.el7_9
nss-softoken-freebl.x86_64 0:3.79.0-4.el7_9
numactl-libs.x86_64 0:2.0.12-5.el7
openssh-clients.x86_64 0:7.4p1-23.el7_9
pam.x86_64 0:1.1.8-23.el7
plymouth-core-libs.x86_64 0:0.8.9-0.34.20140113.el7.centos
postfix.x86_64 2:2.10.1-9.el7
python-libs.x86_64 0:2.7.5-93.el7_9
readline.x86_64 0:6.2-11.el7
rpm-python.x86_64 0:4.11.3-48.el7_9
selinux-policy-targeted.noarch 0:3.13.1-268.el7_9.2
shadow-utils.x86_64 2:4.6-5.el7
systemd.x86_64 0:219-78.el7_9.7
tuned.noarch 0:2.11.0-12.el7_9
virt-what.x86_64 0:1.18-4.el7_9.1
xz-libs.x86_64 0:5.2.2-2.el7_9

Replaced:
iwl7265-firmware.noarch 0:22.0.7.0-69.el7

Complete!
[root@localhost ~]# rpm -qa | grep open
openssh-server-7.4p1-23.el7_9.x86_64
openssl-1.0.2k-26.el7_9.x86_64
openssl-libs-1.0.2k-26.el7_9.x86_64
xmlsec1-openssl-1.2.20-7.el7_4.x86_64
openldap-2.4.44-25.el7_9.x86_64
openssh-7.4p1-23.el7_9.x86_64
openssh-clients-7.4p1-23.el7_9.x86_64
open-vm-tools-11.0.5-3.el7_9.6.x86_64
[root@localhost ~]# openssl version
OpenSSL 1.0.2k-fips 26 Jan 2017
[root@localhost ~]# sudo yum update
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
* base: centos.mirror.rafa.ca
* extras: centos.mirror.rafa.ca
* updates: centos.mirror.rafa.ca
No packages marked for update
[root@localhost ~]# sudo yum update openssl
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
* base: centos.mirror.rafa.ca
* extras: centos.mirror.rafa.ca
* updates: centos.mirror.rafa.ca
No packages marked for update
[root@localhost ~]# openssl version
OpenSSL 1.0.2k-fips 26 Jan 2017
[root@localhost ~]#

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://>

Figure 1-20. This displays that I have verified that the OpenSSL has already been installed in the CentOS VM.

Part 3 – Deploy your Certificate Authority

Description

Next part of Portfolio 1 is to deploy a Certificate Authority. The OpenSSL installed in the CentOS VM will be used to serve as the CA. In this section, we are to use Linux scripting that was learned from the previous term to deliver the required results in this part of the portfolio. Also, we are expected to generate a private key, then create a Certificate Authority (CA). To acquire the desired results, we need to follow the detailed instructions from Lab 1 – CA Deployment lab activity.

Observations

The following is the detailed method of procedure that was followed to configure and deploy the Certificate Authority. Thus, delivering the expected results.

Task 1: Generate a Private Key First to Generate CA Certificate

Step 1: Go to the default directory where the Certificate Authority-related keys are located when using OpenSSL.

Script to run:

```
cd /etc/pki/CA/private/
```

Step 2: Run the following script in the CentOS VM with an installed OpenSSL. This script is to generate the private key first before generating the Certificate Authority.

Script to run:

```
openssl genrsa -aes128 -out JP6102CA.key 2048
```

General command for openssl to generate a private key:

```
openssl genrsa -out <private_key_file> <key_length>
```

Note: If a password is requested, type in “Secret55” for the script to be processed successfully.

Below is the explanation on each component of the script:

openssl the command-line to call or run the OpenSSL for SSL/TLS and cryptographic operations.

genrsa the command to generate an RSA private key.

-aes128 the command which specifies that AES-128 symmetric encryption algorithm will be used for the private key to be generated.

-out JP6102CA.key this identifies that the generated private key is to be named with a file name of JP6102.key

2048 this sets the key size or number of bits to 2048 for the generated private key.

Step 3: When a pass phrase is requested for the generated private key, use "Secret55" as the default password for it.

Task 2: Create a Certificate Authority (CA) Certificate using the Generated Private Key

Step 1: Again, go to the default directory where the Certificate Authority-related keys are located when using OpenSSL.

Script to run:

```
cd /etc/pki/CA/private/
```

Step 2: Run the following script in the CentOS VM with an installed OpenSSL. This script is to create a Certificate Authority (CA) certificate using my uniquely defined naming conventions.

Script to run:

```
openssl req -new -x509 -days 1825  
-key /etc/pki/CA/private/JP6102CA.key  
-out /etc/pki/CA/certs/JP6102CA.crt
```

General command for OpenSSL to create a Certificate Authority:

```
openssl req -new -key <private_key_file>
-out <csr_file>
```

Note: If a password is requested, type in “Secret55” for the script to be processed successfully.

Below is the explanation on each component of the script:

openssl the command-line to call or run the OpenSSL for SSL/TLS and cryptographic operations.

req the command to create and process certificates, including Certificate Signing Requests (CSRs).

-new means that a new certificate request or self-signed certificate is to be created.

-x509 indicates that a self-signed certificate is to be generated or created.

-days 1825 this sets the certificate’s validity period being set in days. 1825 days means a 5-years validity period.

-key /etc/pki/CA/private/JP6102CA.key this defines the path of the previously generated private key file which will be used to generate the certificate.

-out /etc/pki/CA/certs/JP6102CA.crt this defines the path of the generated certificate. This path is set as the output location where the certificate is expected to be saved.

Step 3: Once the script is run, a few information lines will be displayed and needed to be filled out. This list of required information is called a Distinguished Name or a DN and will be incorporated into the certificate request.

With that, the below details are needed to be filled out to the requested field:

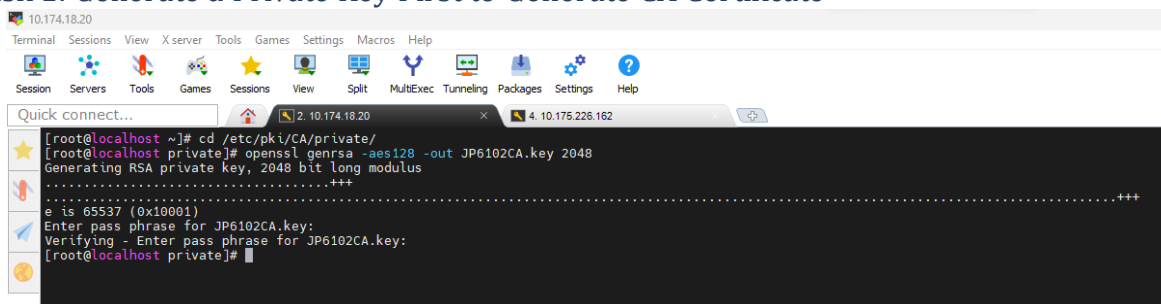
Country Name	CA
State or Province	Ontario
Locality Name	Waterloo
Organizational Name	Conestoga
Organizational Unit Name	ITNS
Common Name	ITNS6102.lab.ca
Email Address	jpangan6102@conestoga.on.ca

Table 1-7. This displays all the needed input information for the required fields of the Distinguished Name data into the certificate request.

Once both scripts from Step 2 and Step 3 have been successfully run, the Certificate Authority server has now been successfully configured and all set to sign client certificates.

Screenshots

Task 1: Generate a Private Key First to Generate CA Certificate

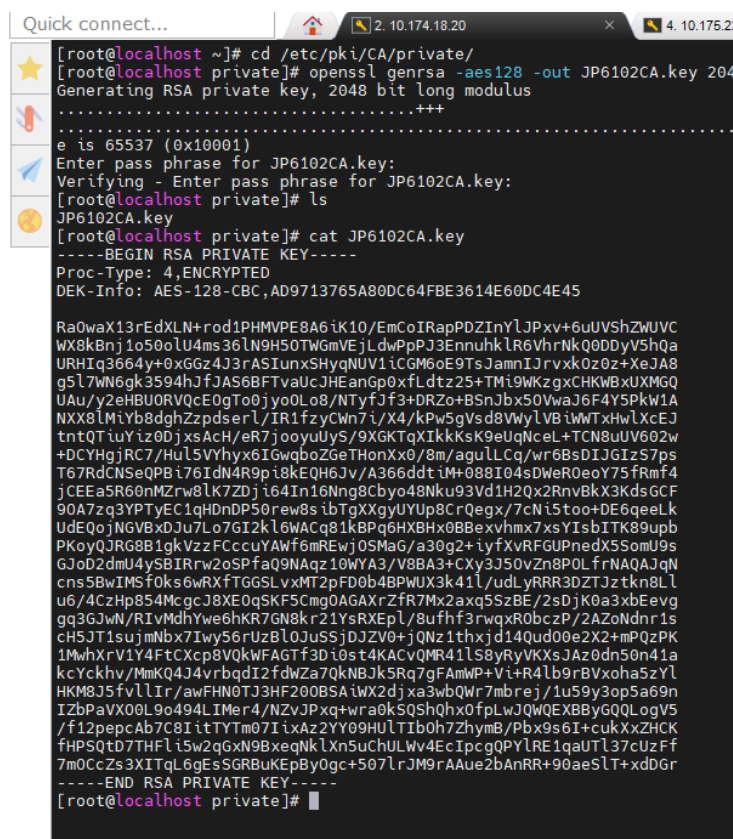


```

10.174.18.20
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
[2. 10.174.18.20] [4. 10.175.228.162]
[root@localhost ~]# cd /etc/pki/CA/private/
[root@localhost private]# openssl genrsa -aes128 -out JP6102CA.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
e is 65537 (0x10001)
Enter pass phrase for JP6102CA.key:
Verifying - Enter pass phrase for JP6102CA.key:
[root@localhost private]#

```

Figure 1-21. This displays that I have successfully generated a private key using the script mentioned in the Observation Section of Part 3 of this Portfolio 1.



```

Quick connect...
[2. 10.174.18.20] [4. 10.175.228.162]
[root@localhost ~]# cd /etc/pki/CA/private/
[root@localhost private]# openssl genrsa -aes128 -out JP6102CA.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
e is 65537 (0x10001)
Enter pass phrase for JP6102CA.key:
Verifying - Enter pass phrase for JP6102CA.key:
[root@localhost private]# ls
JP6102CA.key
[root@localhost private]# cat JP6102CA.key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,AD9713765A80DC64FBE3614E60DC4E45

Ra0waX13rEdXLN+rod1PHMVPE8A6iK10/EmCoIRapPDZInYlJPxv+6uUVShZWUVC
WX8kBnj1o50oLU4ms36lN9H50TWGmVEjLdwPpPJ3EnnuhklR6VhrNkQ0DDyVShQa
URHIq3664y+0xGGz4J3rASiunxSHyqNUV1iCGM6oE9TsJamnIJrvxk0z0z+XeJA8
g5l7WN6gk3594hJfJAS6BFTvaUcJHEangp0xfldtz25+TMi9WKzgXCHKWBxUXMGQ
UAu/y2eHBUORVQcE0gTo0jyoLO8/NTyfJf3+DRZo+BSnJbx50VwaJ6F4Y5PkW1A
NXX8lMiYb8dghZzpdserl/IR1fzyCwn7i/X4/kPw5gVsd8VwylVBiWWTxHwLxcEJ
tntQTiuYiz0DjxsAcH/eR7jooyUyS/9XGKTqXIkkKs9eUqNceL+TCN8uUV602w
+DCYHgJRC7/HuL5VYhyx6IGwqboZGeThonXx0/8m/agulLCq/wr6BsDIjGiZs7ps
T67RdCNSeQPB176IdN4R9p18kEQH6Jv/A366ddtIM+088I04sDwER0eoY75fRmf4
jCEEa5R60nMzrw8lK7ZDji64In16Nng8Cbyo48Nku93Vd1H2Qx2RnvBkX3KdsGCF
90A7zq3YPTyEC1qHDnDP50rew8s1bTgXXgyUYUp8CrQegx/7cN15too+DE6qeLk
UdEQoJNGVBxDJ7L07GI2kl6WACq81kBPq6HXBHx0BBexvhmX7xsYIsbITK89upb
PKoyQJR68B1gkVzzFcCuYAWf6mREwj05MaG/a30g2+iyfXvRFgUPnedX5SomU9s
GJoD2dmU4Y5IRrw2oSPfaQ9NAqz10WYA3/V8BA3+CXy3J50vZn8POLfrNAQAjQn
cns5BwIMSf0ks6wRXftGGSLvxMT2pFD0b4BPWUX3k41l/udLyRRR3DZTjztKn8Ll
u6/4CzHp854McgcJ8XE0qSKF5Cmg0AGAXrZfR7Mx2axq5S2BE/2sDjK0a3xbEevg
gq3GJwN/RivMdhYwe6hKR7GN8kr21YsRXEpl/8ufhf3rwqxR0bczP/2AZoNdnr1s
ch5JT1sujmNbx7Iwy56rUzBl0JuSSjDjZV0+jQnz1thxjd14Qud00e2X2+mPqZPK
1MwhXrV1Y4FtCXcp8VQkWFAGTf3Di0sT4KACvQMR41lS8yRyVKXsJaz0dn50n41a
kcYckhv/MmKQ4j4vrbqdI2fdwZa70kNBjK5Rq7gFAMwP+Vi+R4lB9rBVxoha5zYl
HKM8J5fvlLir/awFHN0TJ3HF200BSA1wX2djxa3wbQw7mbrej/1u59y3op5a69n
IzBPavX00L9o494LIMer4/NZvJPxq+wra0kS0Sh0hx0fplWJQWQEXBBYgQQLOgV5
/f12pepcAb7C8IttYtm07IixAz2Y09HULIb0h7Zhymb/Pbx9s6I+cukXxZHCK
fHPSQtd7THFL1Sw2qGxN9BxeqNkLXn5uChULWv4EcIpcqQPYlRE1qaUTl37cuzFf
7m0CcZs3XITqL6gEsSGRBuKEpByOgc+507lrJM9rAAue2bAnRR+90aeslT+xdGr
-----END RSA PRIVATE KEY-----
[root@localhost private]#

```

Figure 1-22. This displays the content of the generated private key, JP6102.key.

Task 2: Create a Certificate Authority (CA) Certificate using the Generated Private Key

```

[root@localhost private]# openssl req -new -x509 -days 1825 -key /etc/pki/CA/private/JP6102CA.key -out /etc/pki/CA/certs/JP6102CA.crt
Enter pass phrase for /etc/pki/CA/private/JP6102CA.key:
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) [XX]:CA
State or Province Name (full name) []:Ontario
Locality Name (eg, city) [Default City]:Waterloo
Organization Name (eg, company) [Default Company Ltd]:Conestoga
Organizational Unit Name (eg, section) []:ITNSF23
Common Name (eg, your name or your server's hostname) []:JP6102ITNS.lab.ca
Email Address []:jpangan6102@conestogac.on.ca
[root@localhost private]#

```

Figure 1-23. This displays that I have successfully created a self-signed certificate authority (CA) certificate with the required and correct Distinguished Name or DN.

Reflection

Discuss the Content of the Certificate Authority

One of the major components of the Public Key Infrastructure (PKI) is the Certificate Authority (CA). The Certificate Authority (CA) is known as a trusted third-party organization that distributes and control digital certificates. Also, the Certificate Authority (CA) is responsible for ensuring the integrity and validity of a system in secure communication.

Thus, by knowing or learning the contents and purpose of the CA, we were able configure a Certificate Authority server in this part of the portfolio using a Server VM running in CentOS with an OpenSSL toolkit.

Discuss the Components of the Certificate Authority

The components of a Certificate Authority (CA) may vary based on the defined process of implementation and required data. There are numerous components in a Certificate Authority (CA) and listed below are two of the discussed components while the third one is the component that I was able to understand as I progress through the tasks in this portfolio.

- Registration Authority (RA) – This oversees the enrollment and approval of digital certificates. It is considered as a proxy for the CA to control the issuance of certificates. With that, it is responsible for the verification of the identity and authorization of any certificate applicants.
- Certificate Revocation List (CRL) – This oversees establishing and management of the CA's certificate policies and procedures. Thus, the CA produces a CRL to revoke a certificate after it has been issued, and the serial number of the certificate is then deemed invalid.

- Key Generation and Management – The generation and management of the cryptographic key pairs required for certificate issuing are under the responsibility of the CA. In addition to handling the key pairs linked to issued certificates, this involves generating the CA's own key pair.

These mentioned three key components that I have emphasize have help me understand that three main points namely, who is responsible for the enrollment and approval of digital certificates, certificates are not forever valid and can be revoke, and understanding the cryptographic keys are essential to know their role to the CA.

[Explain the Architecture of Certificate Authority](#)

The architecture of Certificate Authority (CA) follows the Public Key Infrastructure (PKI) hierarchy structure. It is expected to only think of CA as a root CA or an issuing CA, however, the CA hierarchy does include different levels of CAs with each having their specific purpose and defined level of trust.

Listed and explained below are the architecture of Certificate Authority (CA):

- Root Certificate Authority (CA) – This is the “trust anchor” for the whole structure of the CA, thus, it is the highest level in the PKI hierarchy. It is a self-signed CA which uses its own private key to be signed. Also, it is in-charge of issuing and signing the certificates for the Intermediate CAs.
- Intermediate Certificate Authority (CA) – Next to the Root CA is the Intermediate CA, which is the middle level in the PKI structure. The certificates for End Entity CAs or for direct End Entity are expected to be issued and signed by intermediate CAs.
- End Entity Certificate Authority (CA) – This is the lowest level in the CA hierarchy and is also known as the Issuing CA. Both the Issuing and Intermediate CAs might exist simultaneously or separately. This CA issues certificates to End Entities such as servers and devices that need certificates for secure communication.

As only one server, Server VM running with CentOS, has been used to manage all the certificates, the Root CA server has been utilized in this portfolio. The Root CA of the CA architecture is known to deliver its own self-signed certificate, and this self-signed certificate has been defined as the type of CA deployed in this task of the portfolio. With that, all other generated certificates in the structure have been acquired from this point.

Explain the Type of Certificate has been Deployed

There are two types of CA that I have observed to be deployed in this portfolio task, the Private CA and the Public CA. To differentiate them and point which task were these certificate type has been process, are as follows:

- Private PKI or Private CA – This is to enhance the security of the internal network as it can only issue certificates for internal use within an organization. Thus, allowing organizations to have their internal trusted infrastructure. This has been configured in Task 2 where a private key has been generated using a script.
- Public PKI or Public CA – This is commonly issued to automatically trusted third-party organizations such as web browsers and devices operating systems. Thus, this is the certificate being issued to the general public. With that, it is definite that the script that was ran in Task 3 which requested information such as Distinguished Name (DN) has granted us for a Public PKI to be configured.

With the specific line of script that was run to generate both the Private and Public PKI or CA, the “-x509” has given the clue that the result has generated a self-signed certificate rather than a full PKI infrastructure. The X.509 standard used for PKI was used in this portfolio activity because this standard contains specific information about the certificate, such as subject, issues, validity period, and other.

Part 4 – Sign the Certificate Signing Request

Description

The last part of this Portfolio 1 is to sign the Certificate Signing Request (CSR). However, in this Portfolio 1, the CSR file that needs to be in the CA Server and required to be signed has been provided along with the Portfolio 1 instruction document. With that, the file named cert_ITNSFW1.csr has been used in the final configuration and completion of the remaining tasks needed.

Once provided CSR file, cert_ITNSFW1.csr, has been successfully copied to the CA and signed, the certificate needs to be imported to the Palo Alto Firewall device for validation. Moreover, a final commit is needed once the certificate has been successfully added along the Device Certificates tab of the Palo Alto Firewall. Thus, this will be followed by a final saving of the running configuration, then by a load command to the base configuration to clear the changes made.

Observations

Task 1: Submit the CSR to the CA Server.

Step 1: Download the provided CSR file named cert_ITNSFW1.csr to the local device or laptop.

Step 2: Open the CSR file named `cert_ITNSFW1.csr` in Notepad++, copy the content of the file, and move it to the CA Server, which is the CentOS VM.

To move the CSR file content, create a file named `cert_ITNSFW1.csr` within the `/etc/pki/CA/private` directory using the `vi` command in Linux.

Script to run:

```
cd /etc/pki/CA/private
vi cert_ITNSFW1.csr
```

Step 3: Save the manually created `cert_ITNSFW1.csr` file in the CentOS VM, then run the `cat cert_ITNSFW1.csr` command to verify the saved file.

Script to run:

```
cat cert_ITNSFW1.csr
```

Task 2: Sign the Certificate in the CA Server Prior to Uploading to the Palo Alto Firewall.

Step 1: Go to the directory where the manually moved or created CSR file named `cert_ITNSFW1.csr` has been saved.

Note that the directory used is `/etc/pki/CA/private`

Step 2: Run the following script in the CentOS VM with an installed OpenSSL. This script is to digitally sign the Certificate Sign Request (CSR) by the Certificate Authority (CA) Server using my uniquely defined naming conventions.

Script to run:

```
openssl x509 -req -in cert_ITNSFW1.csr
-CA /etc/pki/CA/certs/JP6102CA.crt
-CAkey /etc/pki/CA/private/JP6102CA.key
-CACreateserial -out JP6102CA.crt
-days 365
```

General command for OpenSSL to create a Certificate Authority:

```
openssl x509 -req -in <csr_file>
-CA <ca_cert_file>
-CAkey <ca_private_key_file>
-CACreateserial -out <output_cert_file>
-days <validity_period>
```

Note: If a password is requested, type in "Secret55" for the script to be processed successfully.

Below is the explanation on each component of the script:

openssl x509 the command-line to call or run the OpenSSL that is processing the X.509 certificates.

- **req** command to indicate that the input file is a Certificate Signing Request (CSR) file requesting to be signed by the CA.

- **in cert_ITNSFW1.csr** command to define the specific CSR file, or the input file, which contains the public key and other object requesting the certificate.

- **CA /etc/pki/CA/certs/JP6102CA.crt** specifies the path of the Certificate Authority's X.509 certificate file. This will be used to sign the certificate that has been generated.

- **CAkey /etc/pki/CA/private/JP6102CA.key** specifies the path of the Certificate Authority's generated private key. This will be used to sign the certificate that has been generated.

- **CACreateserial** command to call upon OpenSSL to create a serial number for the CA. The serial number that will be generated will act as a unique identifier assigned to every certificate published by the CA.

- **out JP6102CA.crt** this defines the path of the generated certificate. This path is set as the output location where the certificate is expected to be saved.

- **days 365** this sets the certificate's validity period being set in days. 365 days means a 1-year validity period only.

Once the script has been successfully run, the Certificate Signing Request (CSR) has now been successfully digitally signed by the Certificate Authority (CA) server.

Task 3: Import the Signed Certificate to the Palo Alto Firewall.

Step 1: To transfer the successfully signed Digital Certificate, **login to the Palo Alto Firewall device** to import the signed digital certificate.

Step 2: In the **Devices Tab**, click the **Certificates** option along the left panel of the Palo Alto Firewall webpage.

Step 3: Click **Import** and enter the **Certificate Name**, JP6102CA, that was used to generate the Certificate Signing Request (CSR), then **browse for the save signed digital certificate file** in your local device or laptop.

Step 4: Click **OK** once verified that the correct .CRT file has been selected.

Task 4: Validate the Status of the Import CA in the Palo Alto Firewall.

To validate the status of the imported Digital Certificate in the Palo Alto Firewall device, verify the labeled status of the imported certificate.

If the labeled status is set to `valid`, this means that the signed certificate has been successfully imported and within the validity date.

Task 5: Save the Palo Alto Firewall's Running Configuration and Load the Base Configuration.

To complete the configuration of the Portfolio 1, do a final **commit**, then a final saving of the Palo Alto Firewall configuration with exporting the **running-config.xml** file.

Once verified that the configuration changes have been saved via commit and saving the running-config.xml file to the local device or laptop, proceed to clear the configuration made by running the load to base configuration command.

Script to run:

```
load config from Base-Config
```

Screenshots

Task 1: Submit the CSR to the CA Server.

```
[root@localhost private]# vi cert_ITNSFW1.csr
[root@localhost private]# cat cert_ITNSFW1.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICWjCCAUICAQAwFTETMBEGA1UEAxMKMTcyLjE2LjEuMjCCASiWdQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBALwD/Df4mCnGxKU0zW1ag7M2E9/0zITco53zAIyh
InQJolVSxApVfNqHysnZEPa2BQM+ZFD+PKVlNogCLwQTPQWpkck9L7QgLz1TJfvy
PTSEzBv2HhF8KfgZLkfVlnZ19moxXmU16ycV3pal5fJgqCBxSLYC80e07LsVjJIy
ZsYnz7sGzFtX7K+3h1Lr1mSR8mdMs9BuI+BkDqqJ2uRj+L80uPLKn9n24Sr77R26
9crVkyXvJL3wQZc/KPEc/vrn6SoL7+ZicIWLgJx1Ajh4j8aEjELuUmWRWSf9FRJm
zASHX9PtSBLwF4ky0S2kA0+35VG4Ig1xb4DQpbtLLPwaX6cCAwEAaAAMA0GCSqG
SIb3DQEBCwUA4IBAQCCE18KpgeLCUW4cfJcG41wF6ALAPScf2MVVDizbeIzD6od
t4AanGigLch2oZ28cS9P7svmDZ+S3oX1oH8dbjqXW+zLUaaojZLeeG+qs81GG5J0
3L7FFG00Rjla1drdzniTFJ71JAtWQHY8wSfN1lqok3IUgI77iHu0hPsRwm3khozv
I8PyTKAakFK0WU1y9ogpkNtt7fgSVhLPK1hc2Dy7yi0d8FBcIdZVJrjxB0K3WeLX
rLk52FgiUgPx2K28y+AthV9yX1iWq4c3LXmEkFBpGMyL0d0BGhjw0s5ejmxCZSWM
08yC2SykKXuMFgm3JvTyPEqpTnnacU5fJ4a7tFea
-----END CERTIFICATE REQUEST-----
[root@localhost private]#
```

Figure 1-24. This displays that the provided CSR file has been manually added to the CA server.

Task 2: Sign the Certificate in the CA Server Prior to Uploading to the Palo Alto Firewall.

```
[root@localhost private]# openssl x509 -req -in cert_ITNSFW1.csr -CA /etc/pki/CA/certs/JP6102CA.crt -CAkey /etc/pki/CA/private/JP6102CA.key -CAcreateserial -out JP6102CA.crt -days 365
Signature ok
subject=CN=172.16.1.2
Getting CA Private Key
Enter pass phrase for /etc/pki/CA/private/JP6102CA.key:
[root@localhost private]#
```

Figure 1-25. This displays that the script to run for the CA server to sign the certificate has been successfully processed.

```
-----END CERTIFICATE REQUEST-----
[root@localhost private]# openssl x509 -req -in cert_ITNSFW1.csr -CA /etc/pki/CA/certs/JP6102CA.crt -CAkey /etc/pki/CA/private/JP6102CA.key -CAcreateserial -out JP6102CA.crt -days 365
Signature ok
subject=CN=172.16.1.2
Getting CA Private Key
Enter pass phrase for /etc/pki/CA/private/JP6102CA.key:
[root@localhost private]# ls
cert_ITNSFW1.csr  JP6102CA.crt  JP6102CA.key
[root@localhost private]# cat JP6102CA.crt
-----BEGIN CERTIFICATE-----
MIIDMzCCAhsCCQCfB1kuvwWJWtANBgkqhkiG9w0BAQsFADCB0TELMAkGA1UEBhMC
Q0EwEADA0BgNVBAGMB09udGfFyax8xETAPBgNVBACMFdhdG9yYyBvMHRlWwEAYDQK
DAIb25lc3RvZ2EwEADA0BgNVBAsMB0lUTlNGMjE2LjEuMjCCASiWdQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBALwD/Df4mCnGxKU0zW1ag7M2E9/0zITco53zAIyh
InQJolVSxApVfNqHysnZEPa2BQM+ZFD+PKVlNogCLwQTPQWpkck9L7QgLz1TJfvy
PTSEzBv2HhF8KfgZLkfVlnZ19moxXmU16ycV3pal5fJgqCBxSLYC80e07LsVjJIy
ZsYnz7sGzFtX7K+3h1Lr1mSR8mdMs9BuI+BkDqqJ2uRj+L80uPLKn9n24Sr77R26
9crVkyXvJL3wQZc/KPEc/vrn6SoL7+ZicIWLgJx1Ajh4j8aEjELuUmWRWSf9FRJm
zASHX9PtSBLwF4ky0S2kA0+35VG4Ig1xb4DQpbtLLPwaX6cCAwEAaAAMA0GCSqG
SIb3DQEBCwUA4IBAQCCE18KpgeLCUW4cfJcG41wF6ALAPScf2MVVDizbeIzD6od
t4AanGigLch2oZ28cS9P7svmDZ+S3oX1oH8dbjqXW+zLUaaojZLeeG+qs81GG5J0
3L7FFG00Rjla1drdzniTFJ71JAtWQHY8wSfN1lqok3IUgI77iHu0hPsRwm3khozv
I8PyTKAakFK0WU1y9ogpkNtt7fgSVhLPK1hc2Dy7yi0d8FBcIdZVJrjxB0K3WeLX
rLk52FgiUgPx2K28y+AthV9yX1iWq4c3LXmEkFBpGMyL0d0BGhjw0s5ejmxCZSWM
08yC2SykKXuMFgm3JvTyPEqpTnnacU5fJ4a7tFea
-----END CERTIFICATE-----
[root@localhost private]#
```

Figure 1-25. This displays the content of the successfully signed certificate.

Task 3: Import the Signed Certificate to the Palo Alto Firewall.

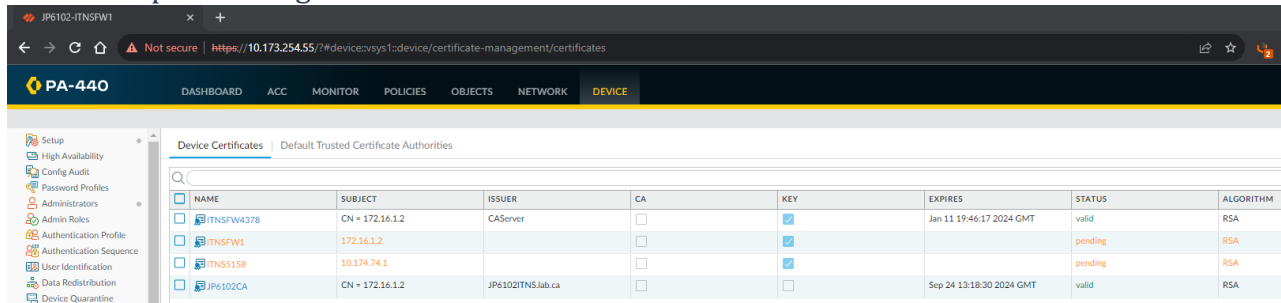


Figure 1-26. This displays that the signed certificate has been successfully imported into the Palo Alto Firewall.

Task 4: Validate the Status of the Import CA in the Palo Alto Firewall.

EXPIRES	STATUS	ALGORITHM	USAGE
Jan 11 19:46:17 2024 GMT	valid	RSA	
	pending	RSA	
	pending	RSA	
Sep 24 13:18:30 2024 GMT	valid	RSA	

Figure 1-27. This displays the zoom-in view of the signed certificate that has been successfully imported. This also shows that the certificate is in the “VALID” status and expiration timestamp is within correct records.

Task 5: Save the Palo Alto Firewall’s Running Configuration and Load the Base Configuration.

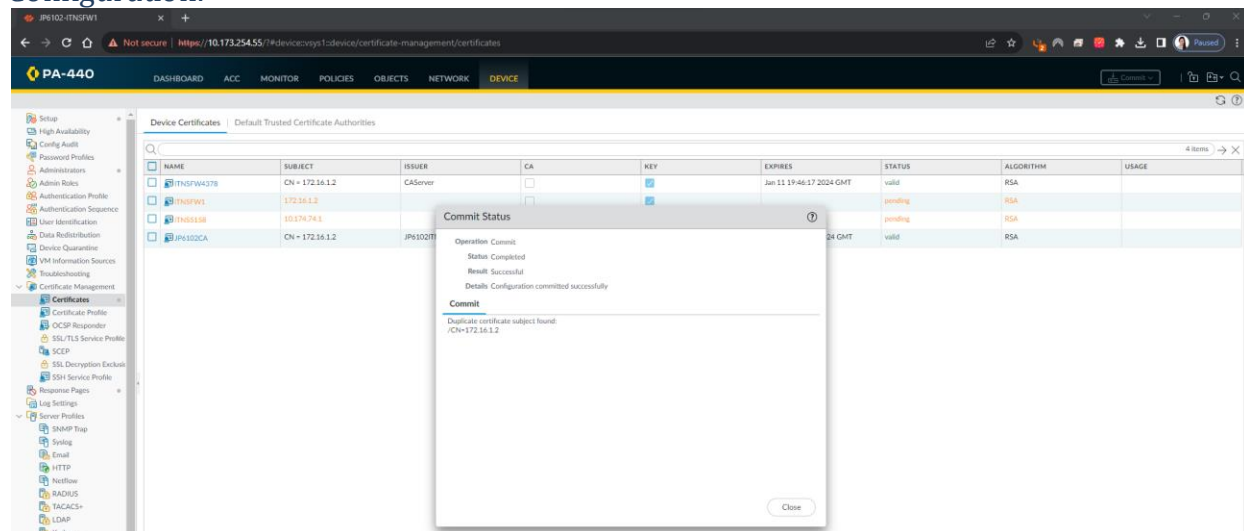


Figure 1-28. This displays the final “commit” that has been made to the Palo Alto Firewall.

Reflection

Explain the Importance of the Certificate Signing Request

As previously explained, the Certificate Authority (CA) is responsible for the issuance and control of digital certificates. And when an entity requests a certificate, a Certificate Signing Request (CSR) is submitted. This CSR is essential because of the following:

- It is used for identifying verification, such as the entity details and public key are included in the process.
- Includes the public key as information to generate the certificate and associate it to the entity's generated private key.
- Contains certificate attributes such as the Distinguished Name (DN) used to define the scope and purpose of the certificate.

Explain the Importance of the Certificate Key

The Certificate Key is mainly the key pairs of private keys and their corresponding public key. As observed with the completed configuration of the final task in this portfolio, which is in Task 4, the importance of Certificate Key are as follows:

- Established a secure communication where the private key is kept secret and used only by the certificate holder. While the public key is being used to the other entity for verification purposes.
- Certify trust and authentication where the private key is being used to prove a trusted source by having digital signed data. Simultaneously, the public key is used to verify the signature of the acclaim trusted source.
- Create a certificate binding where the CA own private key is being used to sign the certificate.

With the combined Certificate Signing Request (CSR) and Certificate Key, secure issuance and handling of certificates is made possible. Both have been used to produce the desired outcome of this portfolio, such as generating the expected results and acquiring the necessary knowledge on every instruction followed.

Explain the Difference Between the Private Key and Public Keys

The difference between the private key and public key has been indirectly explained throughout this portfolio, such that the private key is being kept secret by an entity or organization, while the public key is acquired from the private key and openly given to any entity or organization in need.

Additional difference is that the private key is used for decrypting an encrypted data while public key is used by an entity to encrypt data. Lastly, both keys mainly differ with the security, such that the private key is critical to protect from unauthorized access as it can initiate a security

breach, while the public key does not need to be hidden as it is openly published in digital certificates. Nonetheless, both the private and public keys have been used to form a key pair that ensure secure communication and have delivered the desired output in this portfolio.

Additional verification was made upon my completion of all the requirements of the portfolio to confirm that I was able to produce the expected outcomes of this portfolio.

```
[root@localhost private]# openssl pkey -pubout -in .\JP6102CA.key | openssl sha256
Error opening key .JP6102CA.key
140189372442512:error:02001002:system library:fopen:No such file or directory:bss_file.c:402:fopen('.JP6102CA.key','r')
140189372442512:error:20074002:BIT routines:FILE_CTRL:system lib:bss_file.c:404:
unable to load key
(stdin)= e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
[root@localhost private]# openssl req -pubkey -in .\cert_ITNSFW1.csr -noout | openssl sha256
.csr_ITNSFW1.csr: No such file or directory
139623627265936:error:02001002:system library:fopen:No such file or directory:bss_file.c:402:fopen('.cert_ITNSFW1.csr','r')
139623627265936:error:20074002:BIT routines:FILE_CTRL:system lib:bss_file.c:404:
(stdin)= e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
[root@localhost private]# openssl x509 -pubkey -in .\JP6102CA.crt -noout | openssl sha256
Error opening Certificate .JP6102CA.crt
140270961563536:error:02001002:system library:fopen:No such file or directory:bss_file.c:402:fopen('.JP6102CA.crt','r')
140270961563536:error:20074002:BIT routines:FILE_CTRL:system lib:bss_file.c:404:
unable to load certificate
(stdin)= e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
[root@localhost private]#
```

Figure 1-29. This displays that all three keys generated throughout the completion of this portfolio have delivered the same hash output. Thus, this proves that all the keys match the same public key and hash values.

Script to run:

```
openssl pkey -pubout -in .\JP6102CA.key | openssl sha256

openssl req -pubkey -in .\cert_ITNSFW1.csr -noout | openssl
sha256

openssl x509 -pubkey -in .\JP6102CA.crt -noout | openssl
sha256
```

```

ck connect...
3. 10.174.18.20
Last login: Mon Sep 25 18:47:47 2023 from 10.192.208.167
[root@localhost ~]# openssl x509 -text -noout -in /etc/pki/CA/certs/JP6102CA.crt
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      82:8a:85:ed:cf:9b:30:e9
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=CA, ST=Ontario, L=Waterloo, O=Conestoga, OU=ITNSF23, CN=JP6102ITNS.lab.ca/emailAddress=jpangan6102@conestogac.on.ca
    Validity
      Not Before: Sep 25 13:02:02 2023 GMT
      Not After : Sep 23 13:02:02 2028 GMT
    Subject: C=CA, ST=Ontario, L=Waterloo, O=Conestoga, OU=ITNSF23, CN=JP6102ITNS.lab.ca/emailAddress=jpangan6102@conestogac.on.ca
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
        00:c5:f6:f2:3d:f0:a5:37:5c:92:d8:1b:22:6e:18:
        a0:29:48:2d:06:1c:38:31:2b:50:28:40:d9:0d:5f:
        3b:4c:9d:23:f9:e9:59:37:34:b2:66:5c:62:a3:6d:
        fa:fc:27:79:0c:c8:e0:dd:64:16:3b:6e:7a:74:3d:
        23:7c:e1:17:a2:d5:72:db:a9:b0:b1:ee:34:ee:61:
        ec:b3:d5:65:47:66:5c:f7:f7:f6:46:8d:c0:ea:38:
        3d:c5:53:1d:da:4f:86:49:72:ba:00:50:d5:34:bb:
        6a:cb:22:37:fb:cb:57:f6:f9:3c:09:7c:6c:2b:40:
        de:c2:5d:e8:a2:11:e5:a0:48:8d:29:55:5d:b0:1f:
        18:35:83:32:5c:f0:09:53:27:0a:e6:0f:90:ee:75:
        7c:5a:a2:b9:09:70:9f:58:3f:31:58:87:d1:bc:c5:
        37:4f:fc:72:ae:51:8d:21:a5:ee:5c:af:78:27:a5:
        58:bc:6e:4a:a0:ba:46:a4:79:ca:65:85:aa:31:aa:
        25:a4:08:64:23:96:99:66:f6:47:89:98:c6:f9:0c:
        be:78:53:78:46:e2:c3:ba:50:06:bb:91:92:45:2c:
        76:e3:6e:10:a9:d7:33:ae:2f:2a:75:93:06:2b:76:
        52:fb:a5:87:0e:8d:6f:ef:da:bc:f1:1f:93:16:6d:
        16:41
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        ED:87:10:F4:B6:3D:40:0E:96:45:BF:BB:A8:8C:E8:E6:29:F1:8F:42
      X509v3 Authority Key Identifier:
        keyid:ED:87:10:F4:B6:3D:40:0E:96:45:BF:BB:A8:8C:E8:E6:29:F1:8F:42

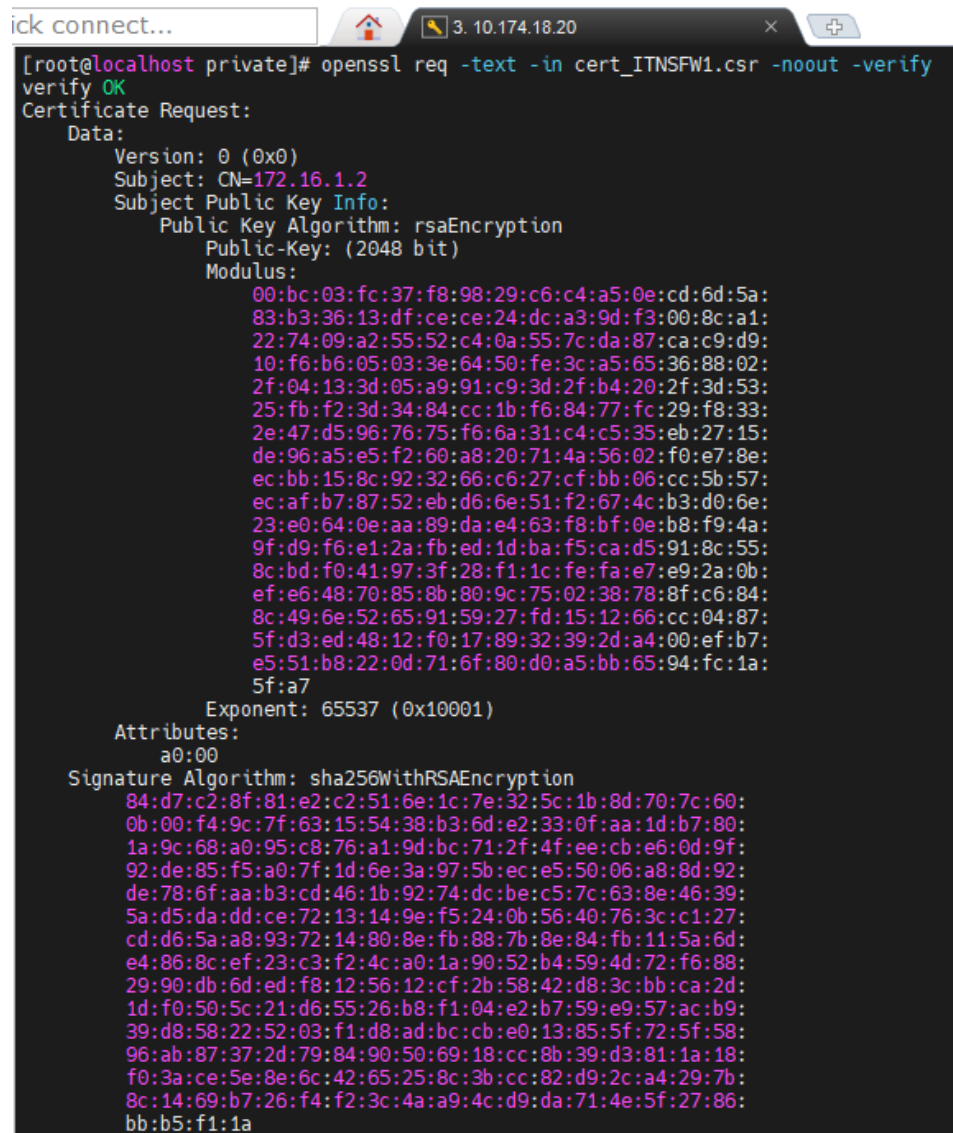
      X509v3 Basic Constraints:
        CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
      9b:18:6d:e9:b4:b7:ee:61:1e:77:49:65:a7:4b:14:0f:d6:51:
      02:58:ce:7f:dc:6a:dd:6f:e3:4a:a5:31:4b:e6:a2:42:14:03:
      1b:c4:c8:e5:21:64:35:22:4e:3d:56:51:c1:6b:2a:6e:7e:68:
      56:e7:8a:f3:3e:ac:87:ea:dc:59:0b:3b:1e:0c:db:eb:ee:f2:
      f4:fc:ea:13:59:52:4d:65:e8:08:25:0b:44:04:ee:e7:67:57:
      3e:60:06:3b:71:90:2e:a3:cc:4f:2b:1e:59:c4:05:bd:81:bf:
      8b:20:b9:8a:c1:61:94:c2:d0:62:62:d4:cd:de:3f:36:5e:3b:
      b5:92:7c:f5:7a:ea:87:cb:93:dc:48:28:d7:65:d5:40:e1:ff:
      10:70:96:e2:c9:e5:ea:4e:88:d3:90:e5:be:7d:2e:36:69:3a:
      07:cf:f9:14:f7:44:11:f6:0e:e8:46:a9:8b:01:f7:2b:ec:38:
      24:f0:8a:be:78:18:2e:35:4e:25:1f:14:33:58:27:37:90:b7:
      5c:e3:60:43:08:58:c4:e0:c0:f5:65:ab:3c:d2:9b:91:fe:ce:
      04:db:b0:01:98:5f:45:e7:27:0e:25:7d:ea:1f:34:95:0d:c6:
      c1:44:2c:5e:64:d5:ac:3a:63:f2:b3:b0:89:5e:28:10:1e:04:
      6a:66:84:f1
[root@localhost ~]#

```

Figure 1-30. This displays that the value or status for the X509 standard was set to TRUE. Thus, this means that the final generated certificate, JP6102.crt, is a self-signed CA.

Script to run:

```
openssl x509 -text -noout -in /etc/pki/CA/certs/JP6102CA.crt
```



```

[root@localhost private]# openssl req -text -in cert_ITNSFW1.csr -noout -verify
verify OK
Certificate Request:
  Data:
    Version: 0 (0x0)
    Subject: CN=172.16.1.2
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
        00:bc:03:fc:37:f8:98:29:c6:c4:a5:0e:cd:6d:5a:
        83:b3:36:13:df:ce:ce:24:dc:a3:9d:f3:00:8c:a1:
        22:74:09:a2:55:52:c4:0a:55:7c:da:87:ca:c9:d9:
        10:f6:b6:05:03:3e:64:50:fe:3c:a5:65:36:88:02:
        2f:04:13:3d:05:a9:91:c9:3d:2f:b4:20:2f:3d:53:
        25:fb:f2:3d:34:84:cc:1b:f6:84:77:fc:29:f8:33:
        2e:47:d5:96:76:75:f6:6a:31:c4:c5:35:eb:27:15:
        de:96:a5:e5:f2:60:a8:20:71:4a:56:02:f0:e7:8e:
        ec:bb:15:8c:92:32:66:c6:27:cf:bb:06:cc:5b:57:
        ec:af:b7:87:52:eb:d6:6e:51:f2:67:4c:b3:d0:6e:
        23:e0:64:0e:aa:89:da:e4:63:f8:bf:0e:b8:f9:4a:
        9f:d9:f6:e1:2a:fb:ed:1d:ba:f5:ca:d5:91:8c:55:
        8c:bd:f0:41:97:3f:28:f1:1c:fe:fa:e7:e9:2a:0b:
        ef:e6:48:70:85:8b:80:9c:75:02:38:78:8f:c6:84:
        8c:49:6e:52:65:91:59:27:fd:15:12:66:cc:04:87:
        5f:d3:ed:48:12:f0:17:89:32:39:2d:a4:00:ef:b7:
        e5:51:b8:22:0d:71:6f:80:d0:a5:bb:65:94:fc:1a:
        5f:a7
      Exponent: 65537 (0x10001)
    Attributes:
      a0:00
  Signature Algorithm: sha256WithRSAEncryption
    84:d7:c2:8f:81:e2:c2:51:6e:1c:7e:32:5c:1b:8d:70:7c:60:
    0b:00:f4:9c:7f:63:15:54:38:b3:6d:e2:33:0f:aa:1d:b7:80:
    1a:9c:68:a0:95:c8:76:a1:9d:bc:71:2f:4f:ee:cb:e6:0d:9f:
    92:de:85:f5:a0:7f:1d:6e:3a:97:5b:ec:e5:50:06:a8:8d:92:
    de:78:6f:aa:b3:cd:46:1b:92:74:dc:be:c5:7c:63:8e:46:39:
    5a:d5:da:dd:ce:72:13:14:9e:f5:24:0b:56:40:76:3c:c1:27:
    cd:d6:5a:a8:93:72:14:80:8e:fb:88:7b:8e:84:fb:11:5a:6d:
    e4:86:8c:ef:23:c3:f2:4c:a0:1a:90:52:b4:59:4d:72:f6:88:
    29:90:db:6d:ed:f8:12:56:12:cf:2b:58:42:d8:3c:bb:ca:2d:
    1d:f0:50:5c:21:d6:55:26:b8:f1:04:e2:b7:59:e9:57:ac:b9:
    39:d8:58:22:52:03:f1:d8:ad:bc:cb:e0:13:85:5f:72:5f:58:
    96:ab:87:37:2d:79:84:90:50:69:18:cc:8b:39:d3:81:1a:18:
    f0:3a:ce:5e:8e:6c:42:65:25:8c:3b:cc:82:d9:2c:a4:29:7b:
    8c:14:69:b7:26:f4:f2:3c:4a:a9:4c:d9:da:71:4e:5f:27:86:
    bb:b5:f1:1a
  
```

Figure 1-31. This displays that the CSR is correct without any modification or sign of being corrupted.

Script to run:

```
openssl req -text -in cert_ITNSFW1.csr -noout -verify
```

References

- Mike Meyers' CompTIA Security+ Certification Guide, Third Edition (Exam SY0-601), 3rd Edition. (2021, May). O'Reilly Online Learning
- CompTIA Security+ SY0-601 Cert Guide, 5th Edition. (2021, August). O'Reilly Online Learning
- Sohrab, F. (2023). Information Governance Security Outcome Cryptography Systems[PowerPoint Slides]. eConestoga
- Sohrab, F. (2023). Lab 1 – Certificate Authority Server Deployment[Lab Document]. eConestoga
- Khalid, S. (2023). Lab01 - Installation and Configuration of NGFW[Lab Document]. eConestoga
- OpenSSL Commands. (n.d.). Pleasant Password Server.
<https://pleasantpasswords.com/info/pleasant-password-server/b-server-configuration/3-installing-a-3rd-party-certificate/openssl-commands>
- OpenSSL Quick Reference Guide. (n.d.). DigiCert.
<https://www.digicert.com/kb/ssl-support/openssl-quick-reference-guide.htm>
- PKI Architecture: Fundamentals of Designing a Private PKI System. (14 August 2023). Hashed Out by The SSL Store™. <https://www.thessslstore.com/blog/pki-architecture-fundamentals-of-designing-a-private-pki-system/>