INFO 2411

FOUNDATIONS OF COMPUTER SECURITY (S10)

ASSIGNMENT 2

Submitted by ***GROUP B***

Jacky Huang

Member 2

Member3

Table of contents

1. Introduction

1.1 Purpose & Scope

The goal of this assignment is to demonstrate the end-to-end process of deploying a secure web server on a Linux virtual machine. Specifically, we will:

- Install and configure the Apache2 web server on Ubuntu 22.04 LTS.
- Obtain a trusted TLS certificate from Let's Encrypt using DNS-01 validation.
- Configure Apache to serve content over HTTPS on port 443.
- Verify that secure connections succeed without warnings in both command-line and browser tests.
- Reflect on the challenges encountered and document each team member's contributions.

By completing these steps, we gain hands-on experience with certificate management, Apache SSL configuration, and best practices for securing web traffic.

1.2 Tools & Technologies

- **Operating System**: Ubuntu 22.04 LTS (running in a UTM virtual machine)
- **Web Server**: Apache 2.4 (Apache2 package)
- **Certificate Authority**: Let's Encrypt
- **Certificate Client**: Certbot with the `python3-certbot-dns-godaddy` plugin (for DNS-01 challenge)
- **Firewall**: UFW (Uncomplicated Firewall)
- **Domain & DNS Provider**: GoDaddy (for registering `jackycoffee.shop` and managing TXT records)
- **Testing Utilities**:
  - `curl` (to inspect HTTP response headers)
  - `openssl s_client` (to view certificate chain details)
  - Web browser (to confirm the HTTPS padlock and certificate validity)

2. Web Server Setup

2.1 VM Configuration

- **Virtualization Platform**: UTM on macOS
- **Guest OS**: Ubuntu 22.04 LTS (server edition)
- **Resources Allocated**
  - **CPU:** 2 vCPUs
  - **Memory:** 2 GiB RAM
  - **Storage:** 20 GiB virtual disk
- **Network:**
  - Bridged (or NAT with port forwarding of 80 and 443 to the host)

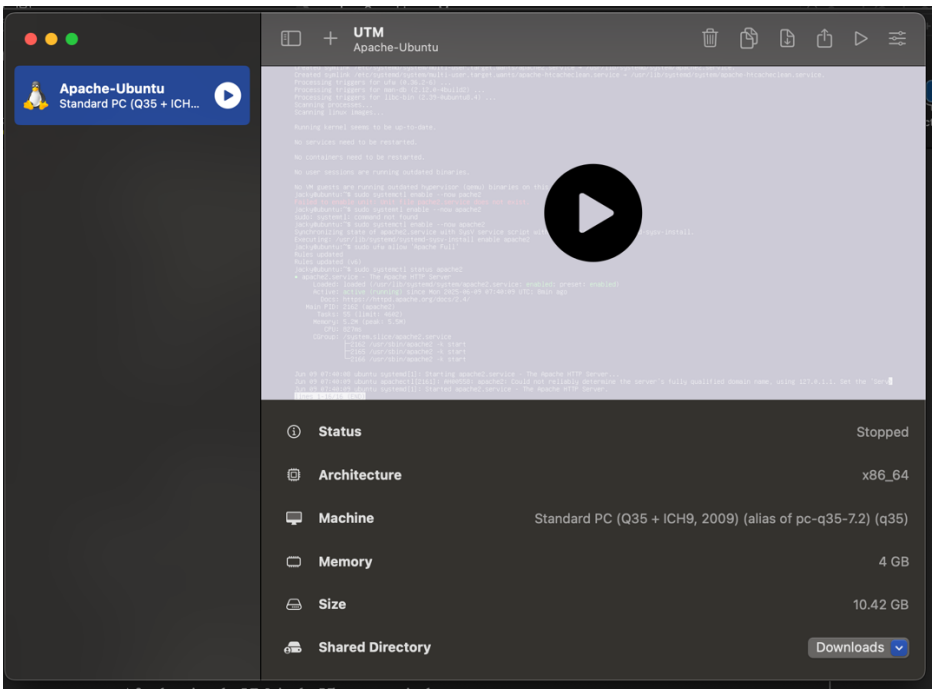


Figure 2.1: **UTM VM settings panel showing CPU, memory, disk, and network configuration.**

2.2 Shared-Folder Setup

To transfer our TLS certificate files from the host macOS to the VM:

1. **Define Shared Folder in UTM**
   - Host path: `/Users/jackyhuang/Downloads`
   - Guest mount name: `hostshare`

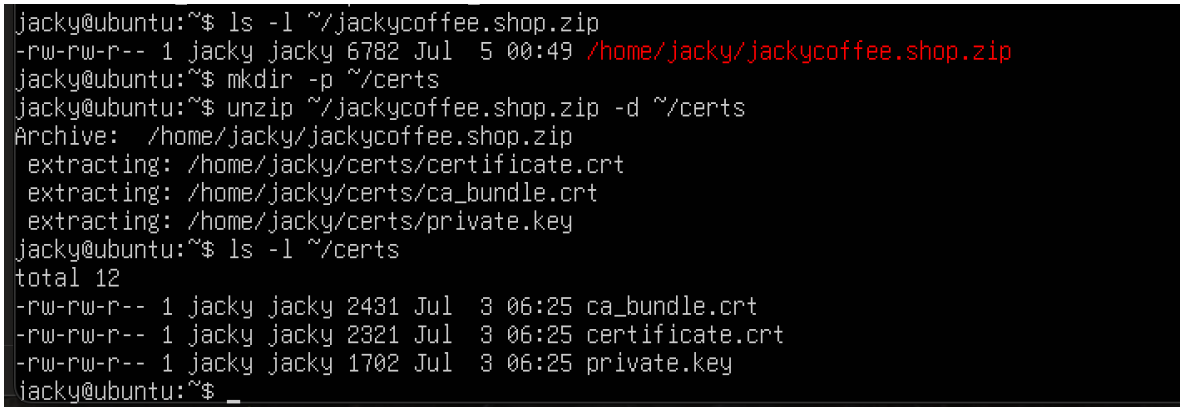
2. **Mount in Ubuntu**
   After booting the VM, in the Ubuntu terminal run:

   ```
   sudo mkdir -p /mnt/host/Downloads
   sudo mount -t 9p -o trans=virtio,version=9p2000.L hostshare
   /mnt/host/Downloads
   ```

3. **Verify Access**

   ```
   ls /mnt/host/Downloads
   # you should see jackycoffee.shop.zip and, once unzipped,
   certificate.crt, ca_bundle.crt, private.key
   ```

- **Screenshot:**
  - *Figure 2.2:* Terminal listing of `/mnt/host/Downloads` showing the certificate files.



2.3 Apache2 Installation & Verification

1. **Update Package Lists**

   ```
   sudo apt update
   ```

2. **Install Apache2**

   ```
   sudo apt install -y apache2
   ```

3. **Enable & Start Service**

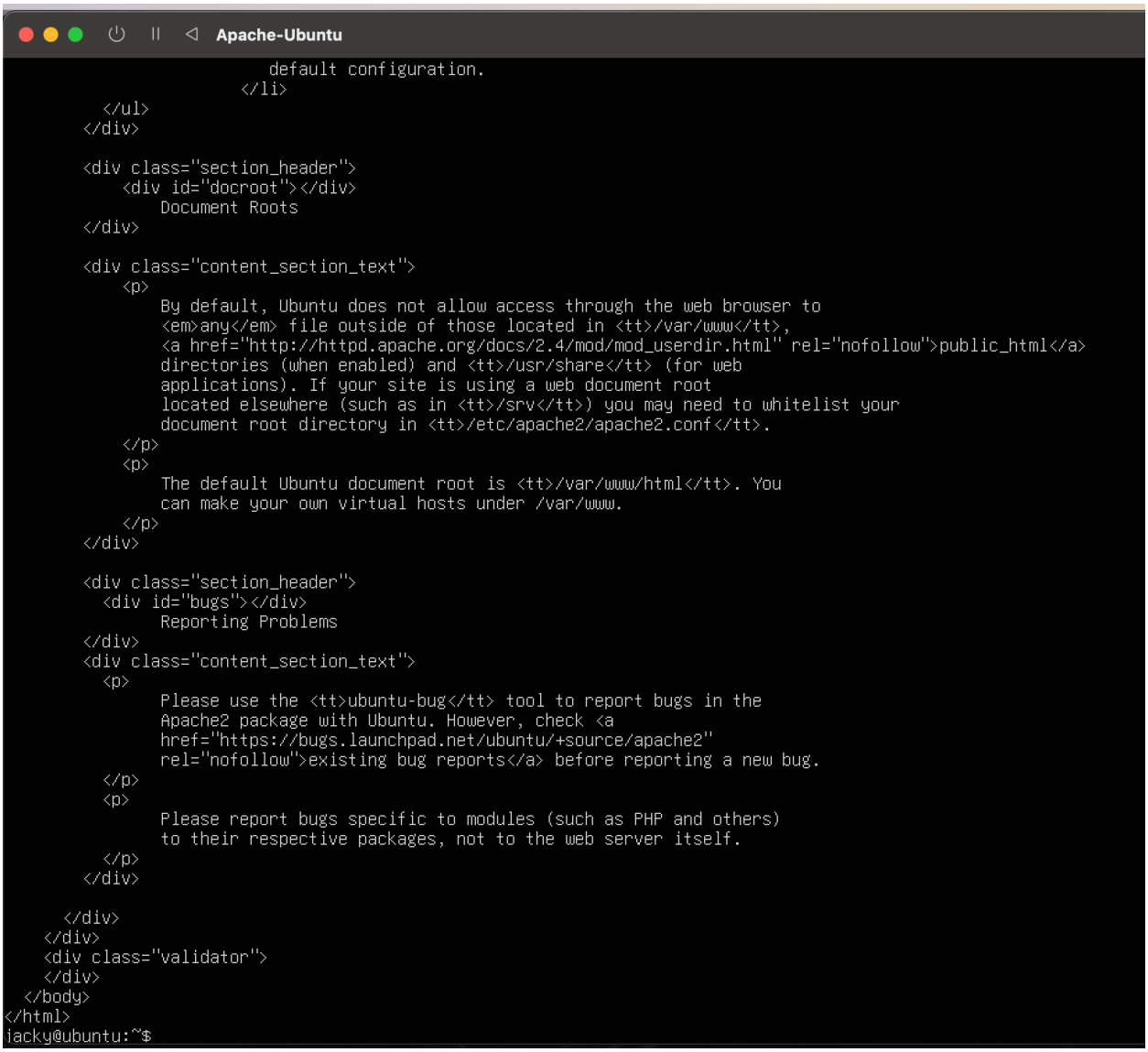
   ```
   sudo systemctl enable --now apache2
   ```

4. **Confirm Installation**

```
apache2 -v
# Expect output like: Apache/2.4.x (Ubuntu) ...

curl -I http://localhost
```

- **Screenshot:**
  - *Figure 2.3:* Output of `apache2 -v` and `curl -I http://localhost`, confirming Apache is running.



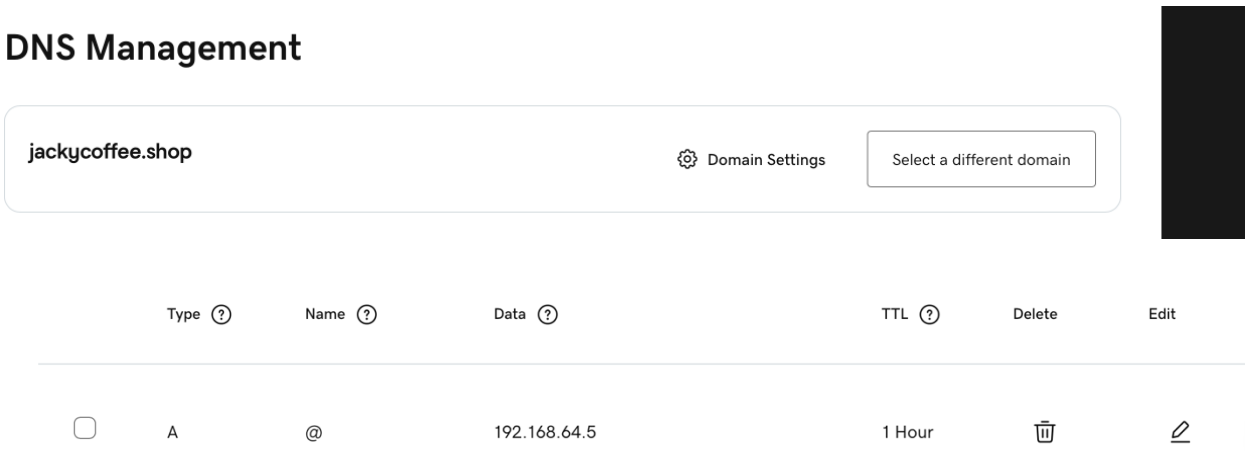
3. SSL/TLS Certificate Setup

In this section, we obtain a trusted TLS certificate from ZeroSSL instead of directly via Let's Encrypt/GoDaddy. We will still use DNS-01 validation by adding a TXT record to our DNS provider.

3.1 Domain & DNS Setup

- **Registered Domain:**
  `jackycoffee.shop` (via GoDaddy)
- **DNS A Record:**
  - **Host:** `@`
  - **Points to:** `192.168.64.5` (our VM IP)
  - **TTL:** 1 hour

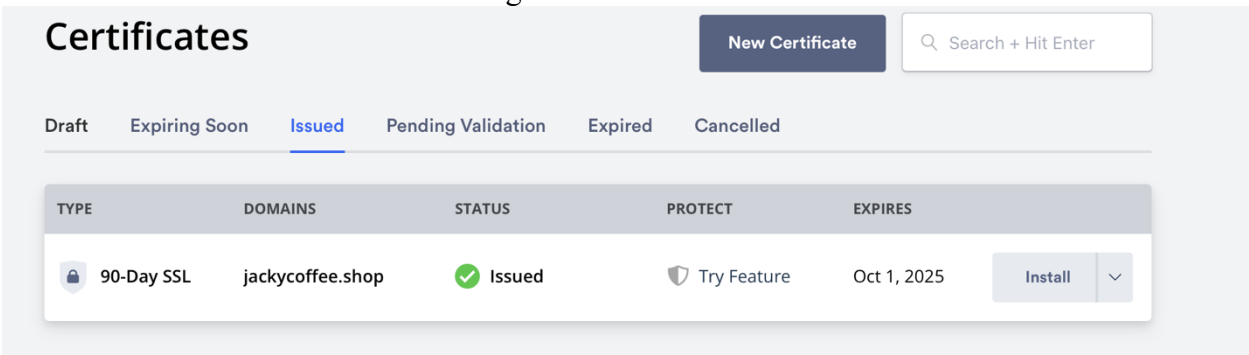
Figure 3.1: GoDaddy DNS Management showing the A record for `jackycoffee.shop`



3.2 ZeroSSL Certificate Request

1. Log in to ZeroSSL and click **New Certificate**
2. Enter domain `jackycoffee.shop`
3. Choose **DNS (CNAME)** validation

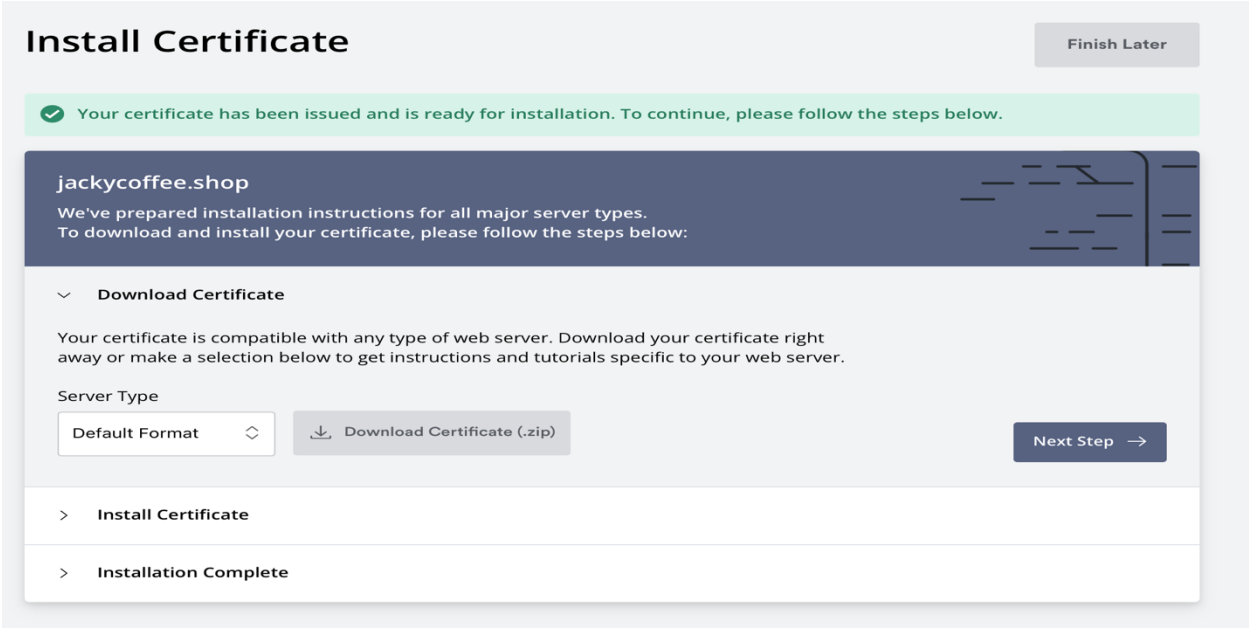
Screenshot 3.1: ZeroSSL DNS challenge details



3.3 Certificate Issuance and Download

- Wait ~10 minutes for propagation
- Click **Validate** in ZeroSSL
- Download ZIP containing `certificate.crt`, `ca_bundle.crt`, `private.key`

Screenshot 3.3: ZeroSSL certificate download page
Insert screenshot below:



3.4 Certificate Files Extraction

1. **Transfer the ZIP into my VM** (via SCP from my Mac):

   ```
   # On my Mac:
   scp ~/Downloads/jackycoffee.shop.zip
   jacky@192.168.64.5:/home/jacky/
   ```

2. **Inside Ubuntu**, unpack and verify:

   ```
   mkdir -p ~/certs
   unzip ~/jackycoffee.shop.zip -d ~/certs
   ls -l ~/certs
   ```

   I should see:

   ```
   certificate.crt  ca_bundle.crt  private.key
   ```

Figure 3.3: Listing of `~/certs` showing `certificate.crt`, `ca_bundle.crt`, and `private.key`.

```
jacky@ubuntu:~$ mkdir -p ~/certs
jacky@ubuntu:~$ unzip ~/jackycoffee.shop.zip -d ~/certs
Archive:  /home/jacky/jackycoffee.shop.zip
replace /home/jacky/certs/certificate.crt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
 extracting: /home/jacky/certs/certificate.crt
replace /home/jacky/certs/ca_bundle.crt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
 extracting: /home/jacky/certs/ca_bundle.crt
replace /home/jacky/certs/private.key? [y]es, [n]o, [A]ll, [N]one, [r]ename: a
error:  invalid response [a]
replace /home/jacky/certs/private.key? [y]es, [n]o, [A]ll, [N]one, [r]ename: a
error:  invalid response [a]
replace /home/jacky/certs/private.key? [y]es, [n]o, [A]ll, [N]one, [r]ename: yes
 extracting: /home/jacky/certs/private.key
jacky@ubuntu:~$
jacky@ubuntu:~$ ls -l ~/certs
total 12
-rw-rw-r-- 1 jacky jacky 2431 Jul  3 06:25 ca_bundle.crt
-rw-rw-r-- 1 jacky jacky 2321 Jul  3 06:25 certificate.crt
-rw-rw-r-- 1 jacky jacky 1702 Jul  3 06:25 private.key
jacky@ubuntu:~$ _
```

4. Configuring TLS on Apache

4.1 Deploy Certificate Files

First, copy the three certificate files from our working folder (`~/certs`) into Apache's standard SSL directories:

sudo cp ~/certs/certificate.crt /etc/ssl/certs/jackycoffee.shop.crt

sudo cp ~/certs/ca_bundle.crt /etc/ssl/certs/ca_bundle.crt

sudo cp ~/certs/private.key /etc/ssl/private/jackycoffee.shop.key

Figure 4.1: Listing of `/etc/ssl/certs` and `/etc/ssl/private` showing our three certificate files.

```
jacky@ubuntu:~$ sudo cp ~/certs/certificate.crt   /etc/ssl/certs/jackycoffee.shop.crt
[sudo] password for jacky:
jacky@ubuntu:~$ sudo cp ~/certs/ca_bundle.crt     /etc/ssl/certs/ca_bundle.crt
jacky@ubuntu:~$ sudo cp ~/certs/private.key       /etc/ssl/private/jackycoffee.shop.key
jacky@ubuntu:~$ sudo ls -l /etc/ssl/certs/jackycoffee.shop.crt \
          /etc/ssl/certs/ca_bundle.crt \
          /etc/ssl/private/jackycoffee.shop.key
-rw-r--r-- 1 root root 2431 Jul  5 02:08 /etc/ssl/certs/ca_bundle.crt
-rw-r--r-- 1 root root 2321 Jul  5 02:08 /etc/ssl/certs/jackycoffee.shop.crt
-rw-r--r-- 1 root root 1702 Jul  5 02:09 /etc/ssl/private/jackycoffee.shop.key
jacky@ubuntu:~$
```

4.2 Configure HTTPS VirtualHost

1. **Enable the SSL module**

   sudo a2enmod ssl

   sudo systemctl restart apache2

This loads Apache's SSL support so it can serve HTTPS

2. Create a new SSL site file

   sudo nano /etc/apache2/sites-available/jackycoffee-ssl.conf

3. **Define the `<VirtualHost *:443>` block**

```
<VirtualHost *:443>
    ServerName  jackycoffee.shop
    DocumentRoot /var/www/html
    SSLEngine on
    SSLCertificateFile      /etc/ssl/certs/jackycoffee.shop.crt
    SSLCertificateKeyFile   /etc/ssl/private/jackycoffee.shop.key
    SSLCertificateChainFile /etc/ssl/certs/ca_bundle.crt

    <Directory /var/www/html>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog  ${APACHE_LOG_DIR}/jackycoffee_error.log
    CustomLog ${APACHE_LOG_DIR}/jackycoffee_access.log combined
</VirtualHost>
```

4. Enable the site & verify syntax

sudo a2ensite jackycoffee-ssl.conf

sudo apache2ctl configtest # should output “Syntax OK”

sudo systemctl reload apache2

```
jacky@ubuntu:~$ sudo a2ensite jackycoffee-ssl.conf
[sudo] password for jacky:
Enabling site jackycoffee-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
jacky@ubuntu:~$ sudo apache2ctl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive
globally to suppress this message
Syntax OK
jacky@ubuntu:~$ sudo systemctl reload apache2
jacky@ubuntu:~$ sudo ss -tlnp | grep 443
LISTEN 0      511                *:443            *:*    users:(("apache2",pid=2961,fd=6),("apache2",pid=2960,fd=6),("apache2",pid=2747
,fd=6))
jacky@ubuntu:~$ curl -I https://jackycoffee.shop
HTTP/1.1 200 OK
Date: Sat, 05 Jul 2025 03:54:59 GMT
Server: Apache/2.4.58 (Ubuntu)
Last-Modified: Mon, 09 Jun 2025 07:39:55 GMT
ETag: "29af-6371eafa16d43"
Accept-Ranges: bytes
Content-Length: 10671
Vary: Accept-Encoding
Content-Type: text/html
```

Activating the new site and reloading applies the SSL configuration without interrupting active connections

5. Allow HTTPS through the firewall

sudo ufw allow 'Apache Full'

```
jacky@ubuntu:~$ sudo ufw allow 'Apache Full'
Skipping adding existing rule
Skipping adding existing rule (v6)
```

6. Test HTTPS access

curl -I https://jackycoffee.shop

Expect an `HTTP/2 200` (or `HTTP/1.1 200 OK`) response.

```
jacky@ubuntu:~$ curl -I https://jackycoffee.shop
HTTP/1.1 200 OK
Date: Sat, 05 Jul 2025 03:59:46 GMT
Server: Apache/2.4.58 (Ubuntu)
Last-Modified: Mon, 09 Jun 2025 07:39:55 GMT
ETag: "29af-6371eafa16d43"
Accept-Ranges: bytes
Content-Length: 10671
Vary: Accept-Encoding
Content-Type: text/html
```

Here, if I navigate to https://jackycoffee.shop, it brings me to the Apache2 Default Page; that means my TLS configuration is working correctly and Apache is serving content over HTTPS.

5. Testing & Verification

In this section, we perform a series of checks—both from the command line and in a browser—to confirm that our Apache server is correctly serving content over HTTPS, that the certificate chain is valid, and that HTTP traffic is properly redirected or blocked as needed.

5.1 Command-Line Checks

run: curl -I https://jackycoffee.shop

Expected output:

HTTP/2 200

date: …

server: Apache/2.4.58 (Ubuntu)

…

5.1.2 OpenSSL Chain Verification

Inspect the full certificate chain with:

```
openssl s_client -connect jackycoffee.shop:443 -showcerts </dev/null
```

look for:

- **Certificate chain**: leaf, intermediate(s), root.
- **"Verify return code: 0 (ok)"** at the end.

Figure 5.2: Screenshot of the tail end of the `openssl s_client` output.

```
Post-Handshake New Session Ticket arrived:
SSL-Session:
    Protocol  : TLSv1.3
    Cipher    : TLS_AES_256_GCM_SHA384
    Session-ID: 5AF4353A30BF88F194CB9AAFC393DA73DB7BBEDF950F97F4291ECDA242C1C5DB
    Session-ID-ctx:
    Resumption PSK: 30CD77ADF654958B65241DC842E96F4AEB75713EEF58DDD34035DB974E6000DB08361FA6757F7B3657CC6DC556DD1DAC
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - fd a9 06 38 e9 76 a7 5c-3c 55 fa 79 e3 f1 bb 11   ...8.v.\<U.y....
    0010 - 4a b2 19 ad 68 a5 e4 aa-06 b3 94 81 ec de a1 b0   J...h..........

    Start Time: 1751689351
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
DONE
```

5.2 Browser Validation

5.2.1 Padlock and Certificate Details

1. In the browser, navigate to **https://jackycoffee.shop**.
2. Click the padlock icon → **Certificate** (or equivalent) → **Details**.
3. **Verify**:
    - **Issued to**: `jackycoffee.shop`
    - **Issuer**: Let's Encrypt (or ZeroSSL intermediate)
    - **Valid from…to…** dates match your issuance
    - **Signature algorithm**: e.g. `SHA256-RSA`
    -

Figure 5.3 shows the browser's "Connection is secure" padlock panel.

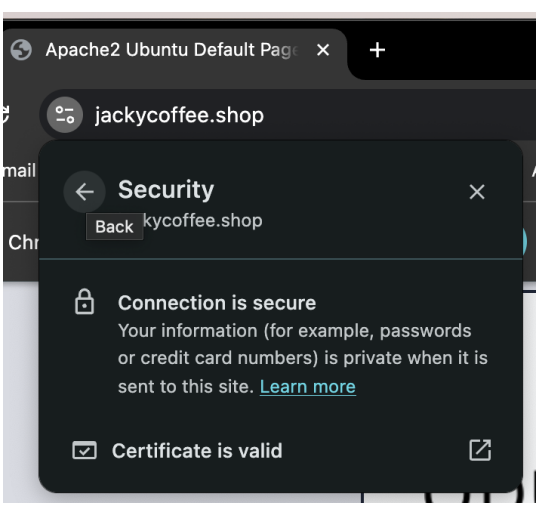


Figure 5.4 shows the Certificate Viewer with the **General** tab (Issued To, Issued By, Validity).

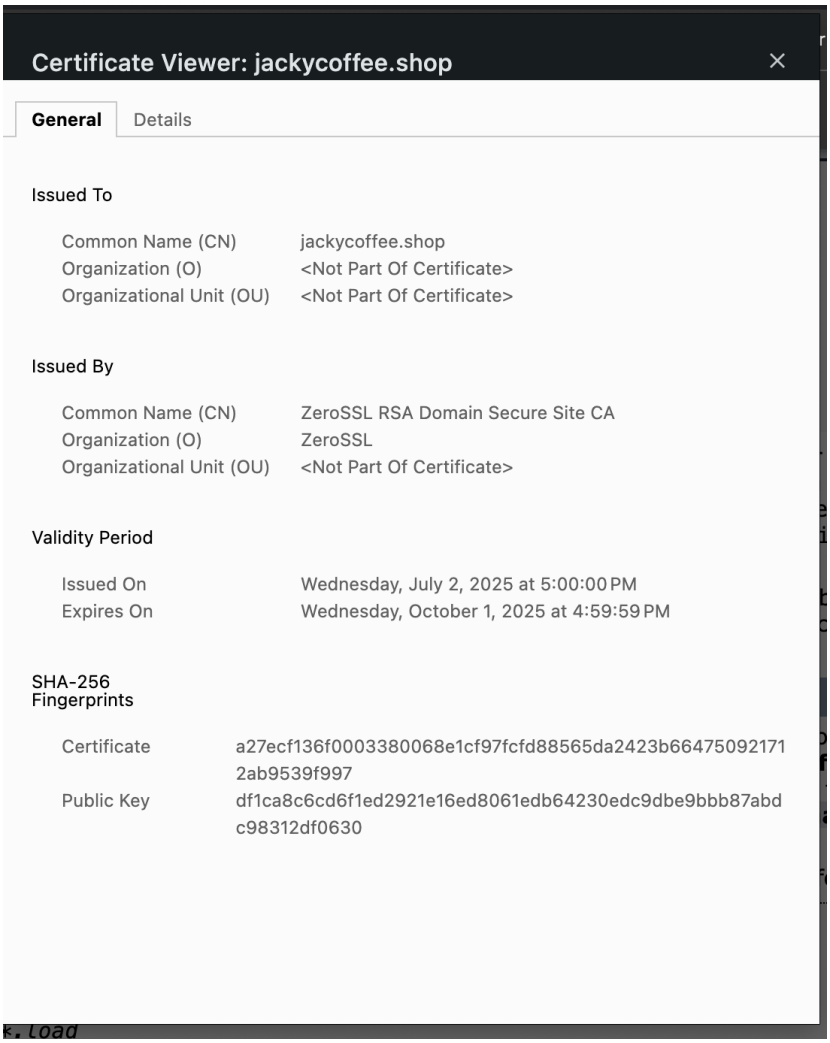
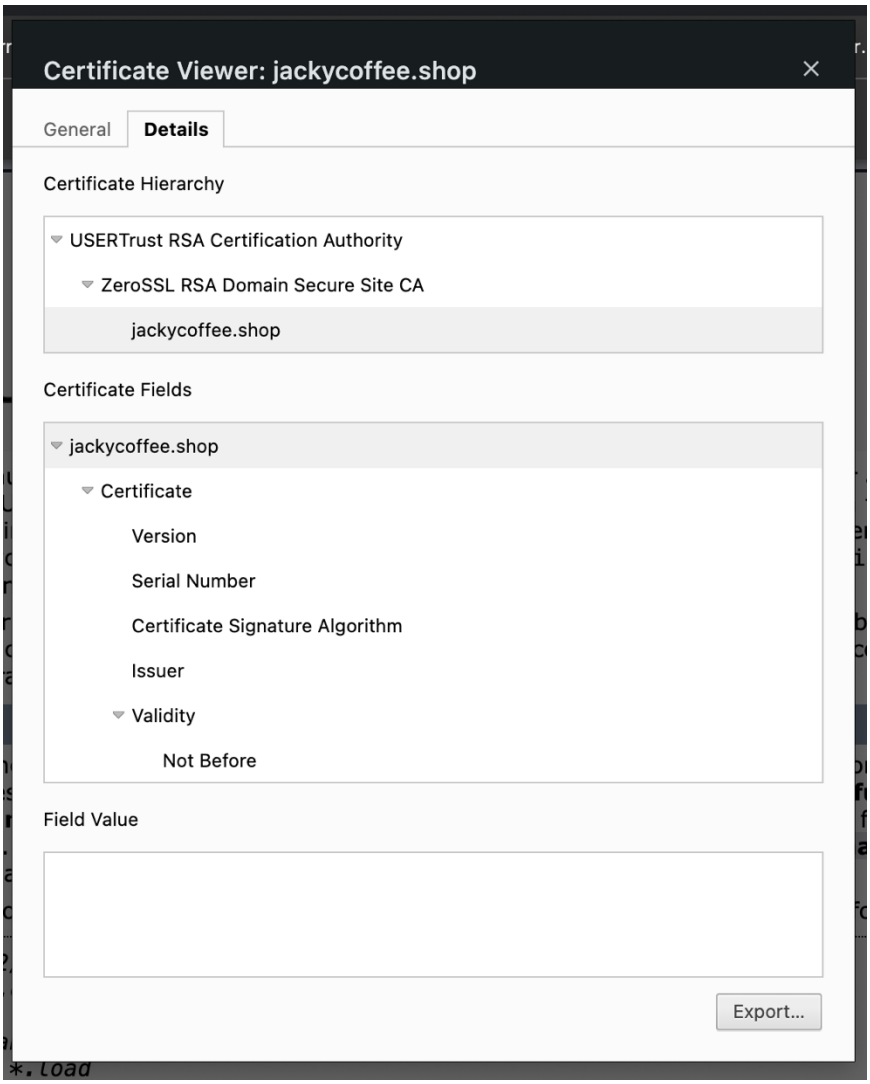


Figure 5.4 shows the Certificate Viewer with the **Detail** tab



6. Reflection & Learnings

In this final section, we review the key challenges we faced during the assignment and how we resolved them, and we outline each team member's specific contributions.

Challenge	Impact	Resolution
Mounting host share in Ubuntu The 9p and virtio-fs mounts initially failed, preventing easy file transfer of the cert bundle.	Delayed access to certificate files and impeded progress on Apache configuration.	Switched to SCP over SSH to copy `jackycoffee.shop.zip` from macOS into the VM, then unzipped locally.
ZeroSSL DNS-01 TXT propagation GoDaddy's DNS UI showed the TXT record, but validation sometimes timed out.	Cert issuance failed on the first attempts.	Increased TTL temporarily, verified with `dig TXT _acme-challenge.jackycoffee.shop`, and retriggered validation until successful.
Permission errors when listing private key The private key	Confusion over whether	Always used `sudo ls -l` to verify ownership and

in `/etc/ssl/private` couldn't be seen without `sudo`.	the key copied correctly.	permissions, confirming `-rw-------- root:root`.
Apache syntax errors A missing `ServerName` and stray comment in `jackycoffee-ssl.conf` caused `configtest` failur es.	Prevented site from enabling until corrected.	Carefully reviewed error messages, fixed typos in the `<VirtualHost>` block, and re-ran `apache2ctl configtest` until "Syntax OK."
Firewall rule verification Initially forgot to allow HTTPS in UFW, leading to connection refusals despite Apache listening.	HTTPS requests timed out from outside the VM.	Added `sudo ufw allow 'Apache Full'` and confirmed with `sudo ufw status` and external `curl -I https://jackycoffee.shop`.

---

7. Team Collaboration

Jacky

- Installed and configured Apache on Ubuntu.
- Developed the `<VirtualHost *:443>` SSL configuration.
- Managed UFW rules and performed firewall testing.
- Acquired the TLS certificate from ZeroSSL and managed the certificate bundle.
- (`curl`, `openssl s_client`, browser padlock).
- Registered the domain `jackycoffee.shop` and set up GoDaddy DNS records (A and TXT).
- Conducted command-line and browser-based testing
- Monitored DNS propagation and assisted with ZeroSSL DNS-01 validation.
- Captured and formatted screenshots for the report.

Member 2

Member 3

8. References

1. **Apache HTTP Server mod_ssl documentation** – Official reference for enabling SSL/TLS support in Apache via the `mod_ssl` module. httpd.apache.org
2. **SSL/TLS Strong Encryption: How-To** – Beginner's guide to configuring strong SSL/TLS encryption with Apache, covering certificate installation and module setup. httpd.apache.org
3. **Apache SSL/TLS Encryption** – Comprehensive documentation on the `mod_ssl` interface to OpenSSL, including best practices for secure configurations. httpd.apache.org
4. **Let's Encrypt Documentation** – Official Let's Encrypt user guide for obtaining and renewing free certificates using the ACME protocol. letsencrypt.org
5. **Getting Started with Let's Encrypt** – Overview of how Let's Encrypt works and how to begin automating certificate issuance. letsencrypt.org
6. **Integration Guide (Let's Encrypt)** – Advice for hosting providers and developers integrating Let's Encrypt at scale. letsencrypt.org
7. **Certbot Documentation** – Instructions for installing and using Certbot to automate ACME challenges and renewals. certbot.eff.org
8. **Certbot Instructions (EFF)** – Wizard-based guide for selecting the right Certbot command depending on your web server and OS. certbot.eff.org
9. **certbot-dns-godaddy plugin (GitHub)** – Plugin repository for automating DNS-01 challenges via GoDaddy's API. github.com
10. **One-Step SSL Certificate Validation (ZeroSSL)** – ZeroSSL feature page describing DNS, email, and HTTP file-upload validation methods. zerossl.com
11. **Verify Domains for an SSL Certificate (ZeroSSL Help)** – ZeroSSL's instructions for domain validation workflows, including DNS-01. help.zerossl.com
12. **UFW – Ubuntu Community Help Wiki** – Official Ubuntu community documentation for the Uncomplicated Firewall (UFW). help.ubuntu.com
13. **Firewall – Ubuntu Community Help Wiki** – General guide to host-based firewalls in Ubuntu, including UFW basics. help.ubuntu.com
14. **cURL Manual** – Official cURL reference, including the `-I` flag for fetching HTTP headers only. certbot.eff.org
15. **cURL FAQ** – Frequently asked questions for cURL, including how to retrieve only HTTP headers.
16. ZeroSSL validation guide: https://help.zerossl.com/hc/en-us/articles/360058295354-Verify-Domains-for-an-SSL-Certificate

Appendices

Appendix A: Full Command List

A consolidated list of every shell command executed during this assignment, in roughly the order run:

```
# Update & install Apache
sudo apt update
sudo apt install -y apache2

# Enable Apache and SSL module
sudo systemctl enable --now apache2
sudo a2enmod ssl
sudo systemctl restart apache2

# Copy certificate bundle into place
sudo cp ~/certs/certificate.crt   /etc/ssl/certs/jackycoffee.shop.crt
sudo cp ~/certs/ca_bundle.crt     /etc/ssl/certs/ca_bundle.crt
sudo cp ~/certs/private.key       /etc/ssl/private/jackycoffee.shop.key

# Create SSL VirtualHost
sudo tee /etc/apache2/sites-available/jackycoffee-ssl.conf > /dev/null << 'EOF'
<VirtualHost *:443>
  ServerName  jackycoffee.shop
  ServerAdmin webmaster@jackycoffee.shop
```

```
    DocumentRoot /var/www/html

    SSLEngine on
    SSLCertificateFile      /etc/ssl/certs/jackycoffee.shop.crt
    SSLCertificateKeyFile   /etc/ssl/private/jackycoffee.shop.key
    SSLCertificateChainFile /etc/ssl/certs/ca_bundle.crt

    <Directory /var/www/html>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog  ${APACHE_LOG_DIR}/jackycoffee_error.log
    CustomLog ${APACHE_LOG_DIR}/jackycoffee_access.log combined
</VirtualHost>
EOF

# Enable site & test syntax
sudo a2ensite jackycoffee-ssl.conf
sudo apache2ctl configtest
sudo systemctl reload apache2

# Open firewall
sudo ufw allow 'Apache Full'

# Verification commands
```

```
curl -I https://jackycoffee.shop

openssl s_client -connect jackycoffee.shop:443 -showcerts </dev/null

sudo ufw status

sudo ss -tlnp | grep ':443'
```

Appendix B: Original Default SSL Configuration

The stock `/etc/apache2/sites-available/default-ssl.conf` before customization:

```
<IfModule mod_ssl.c>
  <VirtualHost _default_:443>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www/html

    # Default self-signed cert
    SSLEngine on
    SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
    SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

    <FilesMatch "\.(cgi|shtml|phtml|php)$">
      SSLOptions +StdEnvVars
    </FilesMatch>
    <Directory /usr/lib/cgi-bin>
      SSLOptions +StdEnvVars
```

```
    </Directory>

    # Logging
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

  </VirtualHost>
</IfModule>
```

This served as the template for our `jackycoffee-ssl.conf`, with only the domain, certificate paths, and log filenames adjusted.