**INSPECT HTTPS CERTIFICATES AND UNDERSTAND THE TRUST CHAIN**

**1. Introduction**

HTTPS certificates are essential for secure web communication, providing encryption and identity verification. The **trust chain** (or certificate chain) is a hierarchical system where Certificate Authorities (CAs) vouch for the authenticity of websites. Understanding this chain helps in troubleshooting SSL/TLS issues and ensuring secure connections.

**Key Concepts:**

* **Root Certificate**: The top-level CA certificate that self-signs its own certificate
* **Intermediate Certificate**: Middle-layer certificates that bridge root and end-entity certificates
* **End-Entity Certificate**: The actual website certificate presented to visitors
* **Trust Chain**: The hierarchical validation path from end-entity to root certificate

**2. Pre-Requisites**

Before starting, ensure you have:

* Basic command-line knowledge
* Access to a terminal/command prompt
* Internet connection
* For GUI methods: A modern web browser
* Target website URL for inspection

## 3. Installation Procedures

### OpenSSL (Command Line Tool)

**Windows:**

bash

choco install openssl

**Linux (Ubuntu/Debian):**

bash

sudo apt update && sudo apt install openssl

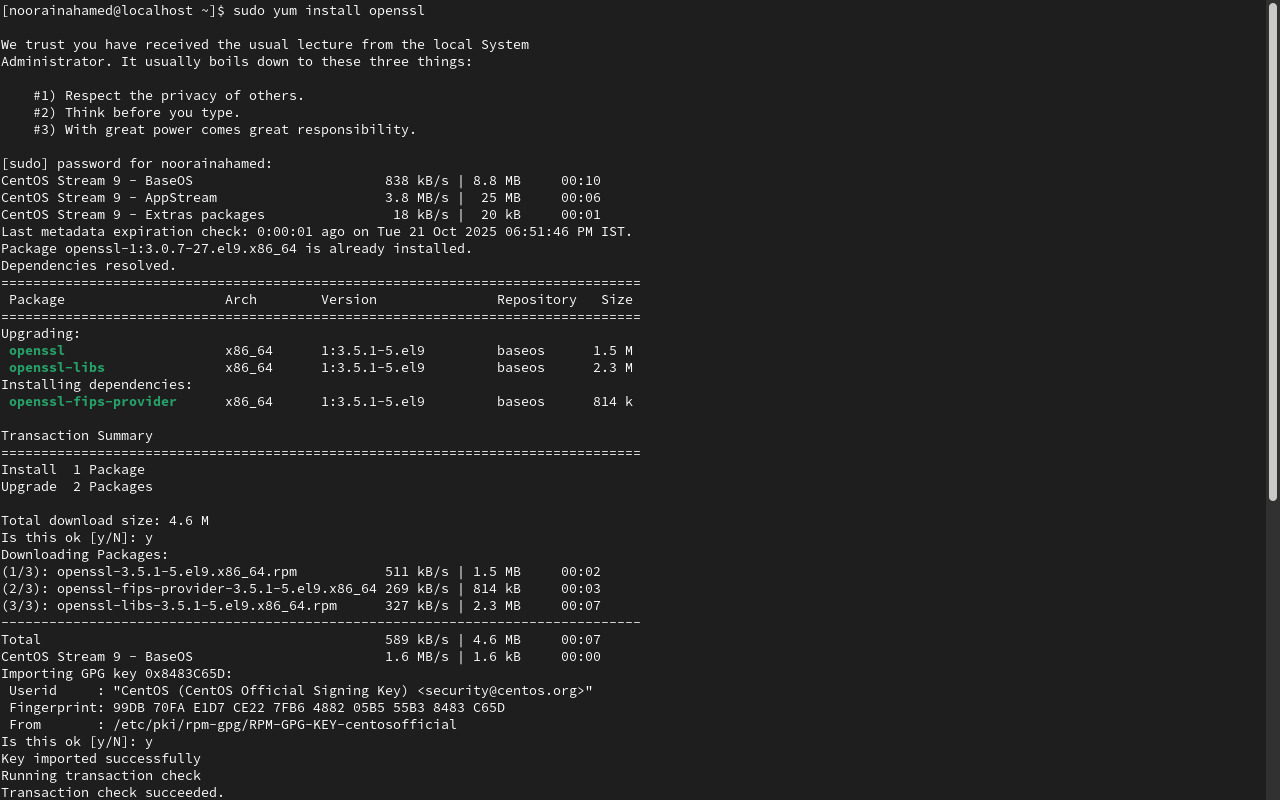
**Linux (CentOS/RHEL):**

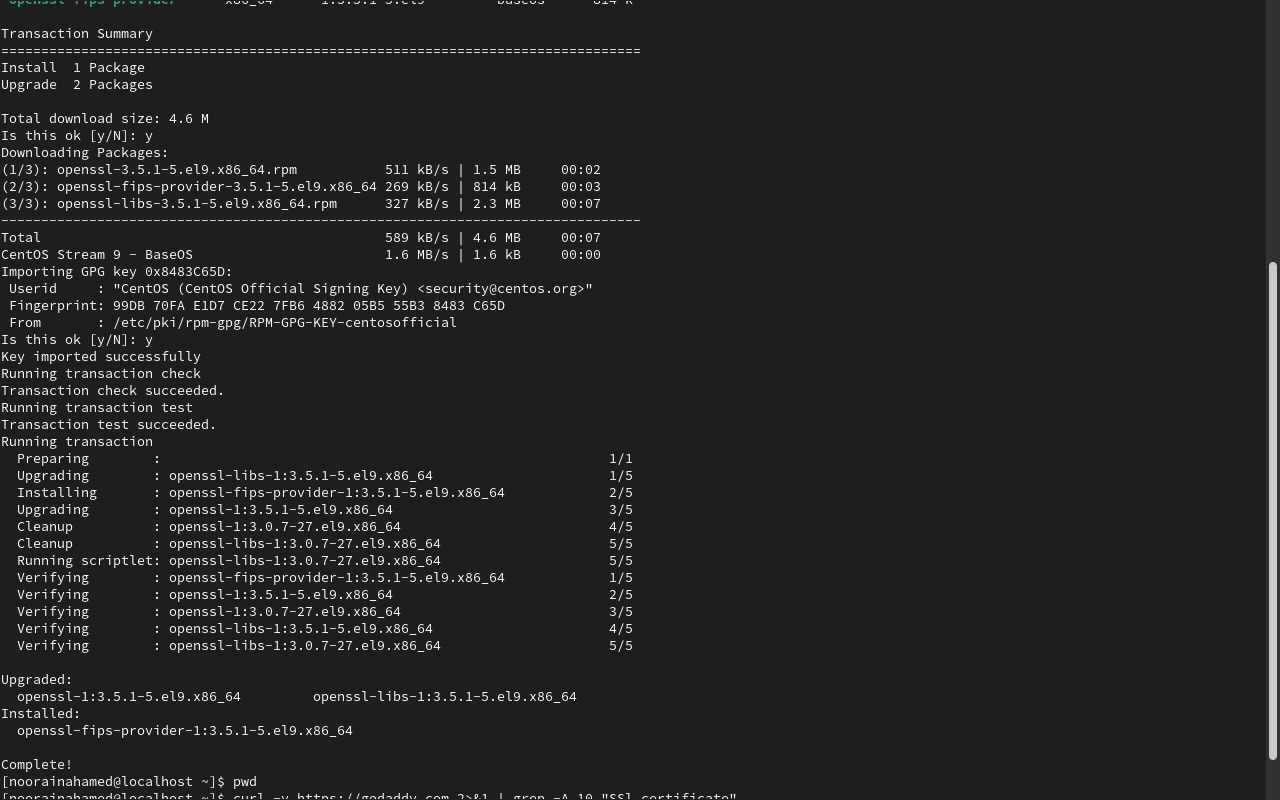
bash

sudo yum install openssl

*# or for newer versions*

sudo dnf install openssl





## 4. Commands for Certificate Inspection

### Method 1: Using OpenSSL (Command Line)

#### **Basic Certificate Information**

bash

*# Get certificate details from a website*

openssl s\_client -connect example.com:443 -servername example.com < /dev/null | openssl x509 -text -noout

#### Extract Full Certificate Chain

bash

*# Show complete certificate chain*

openssl s\_client -connect example.com:443 -showcerts -servername example.com < /dev/null

#### Save Certificate to File

bash

*# Save certificate to file*

openssl s\_client -connect example.com:443 -servername example.com < /dev/null | openssl x509 -out certificate.crt

#### Check Certificate Validity Period

bash

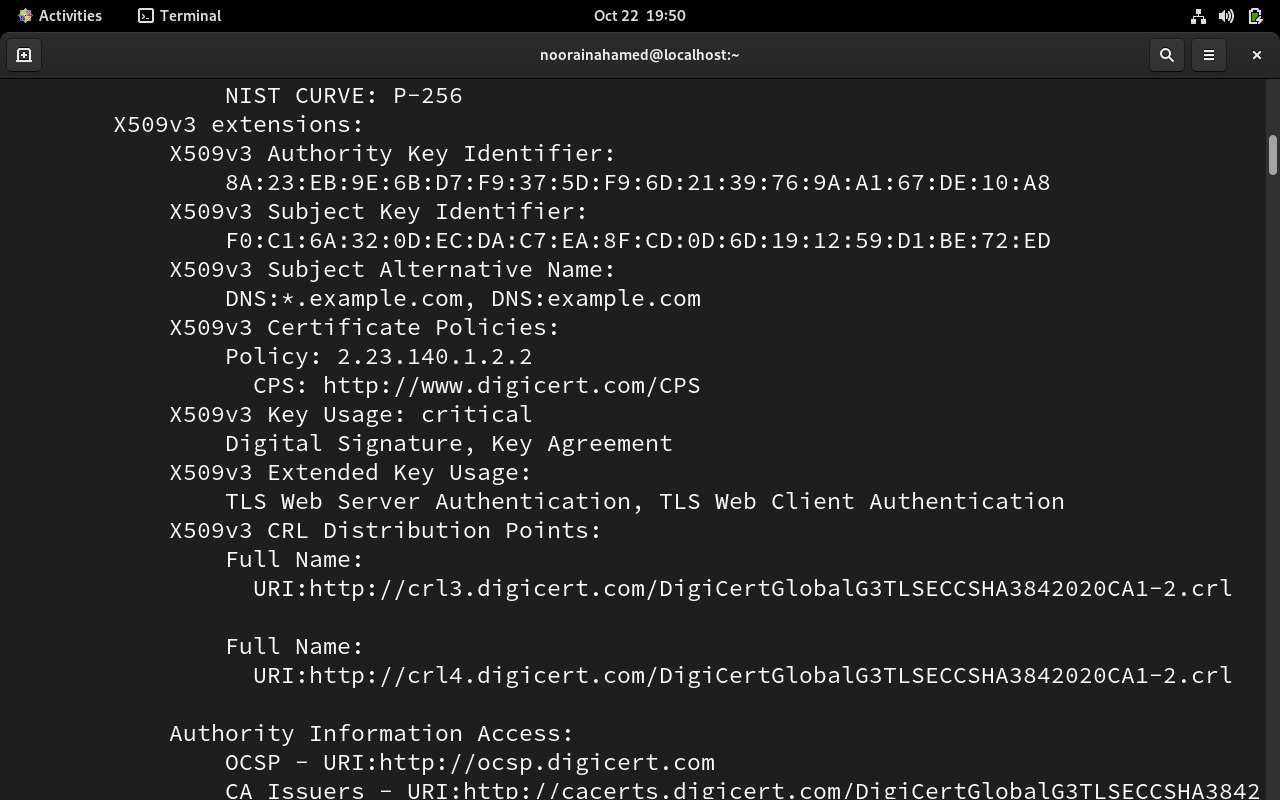
*# Check certificate expiration*

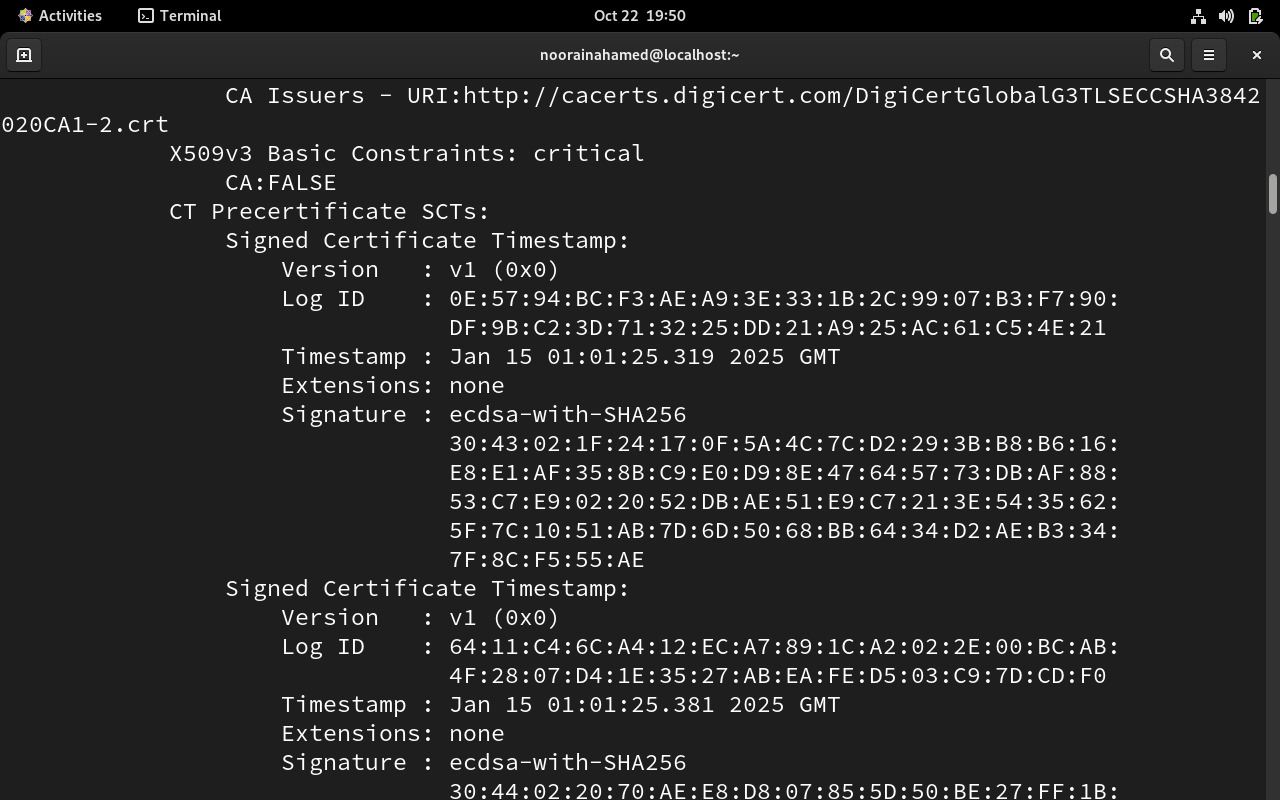
openssl s\_client -connect example.com:443 -servername example.com < /dev/null 2>/dev/null | openssl x509 -noout -dates

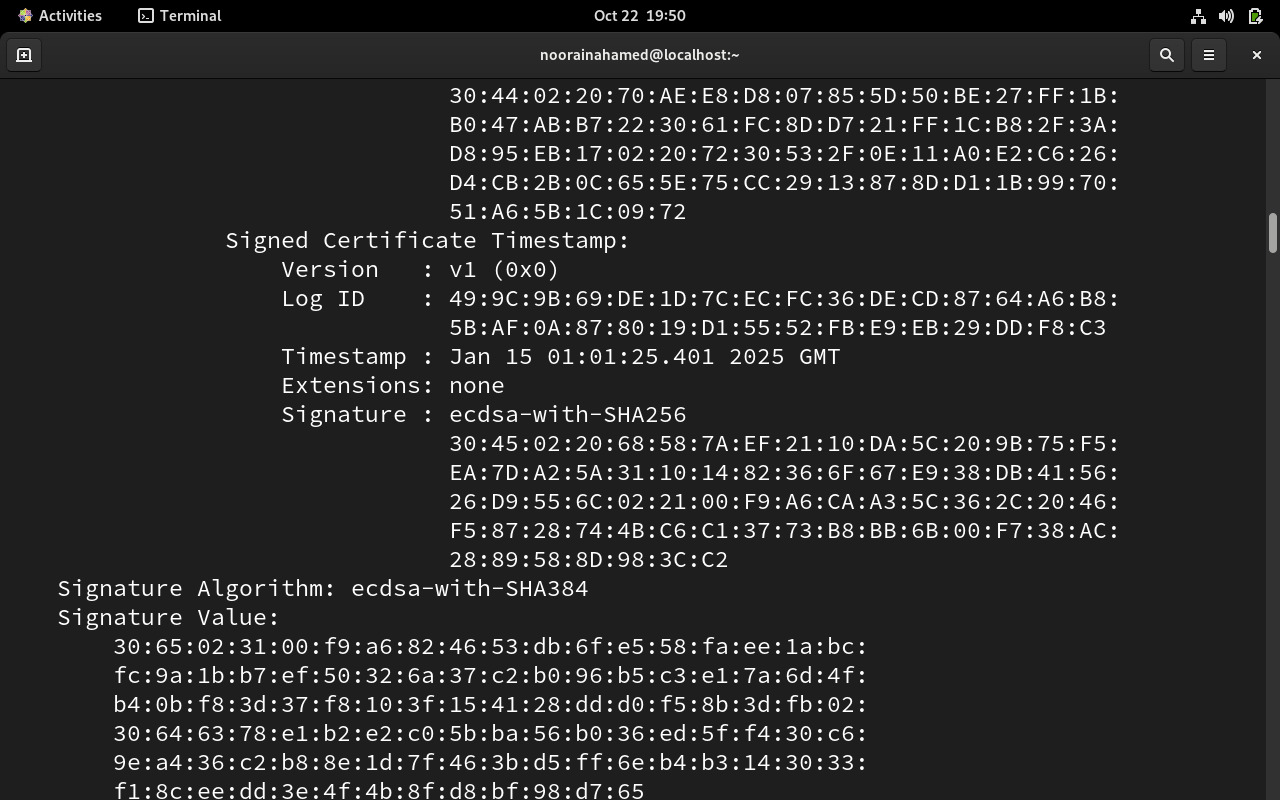
**5. Upload Screenshots**

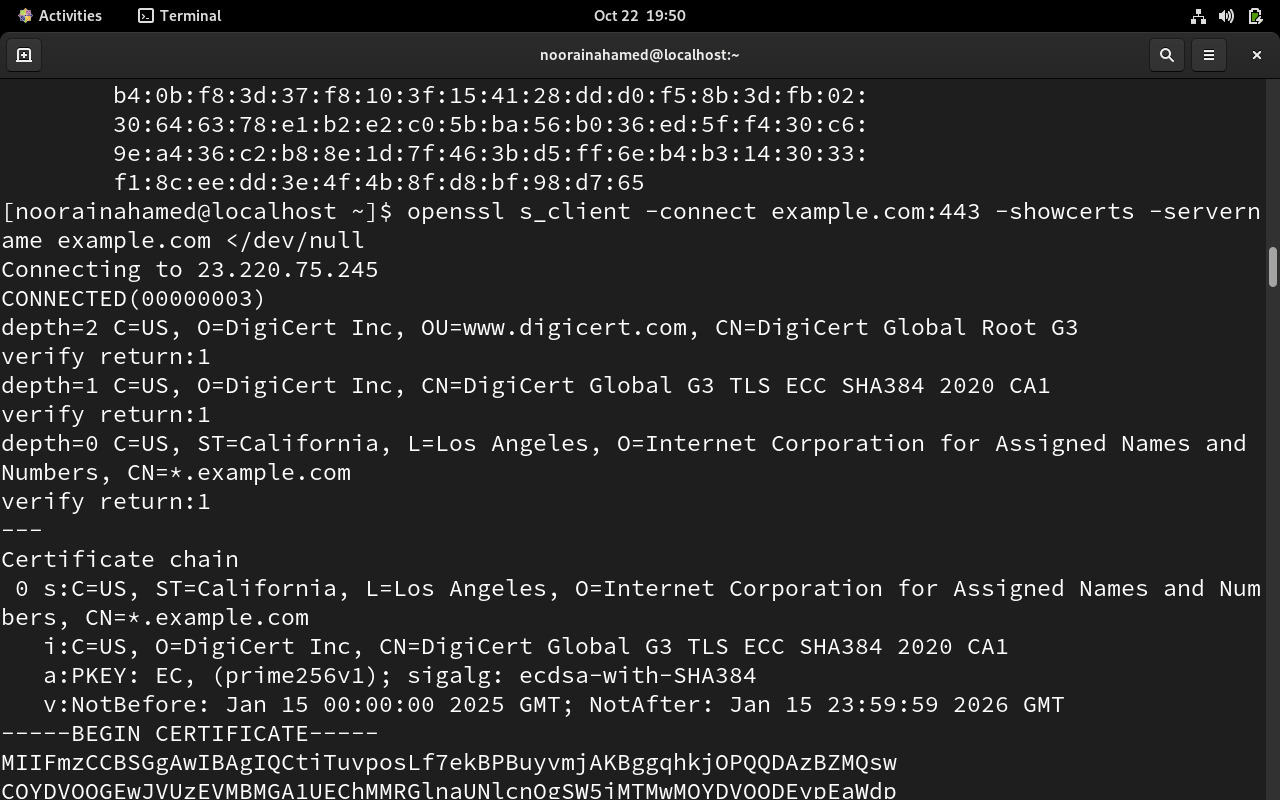




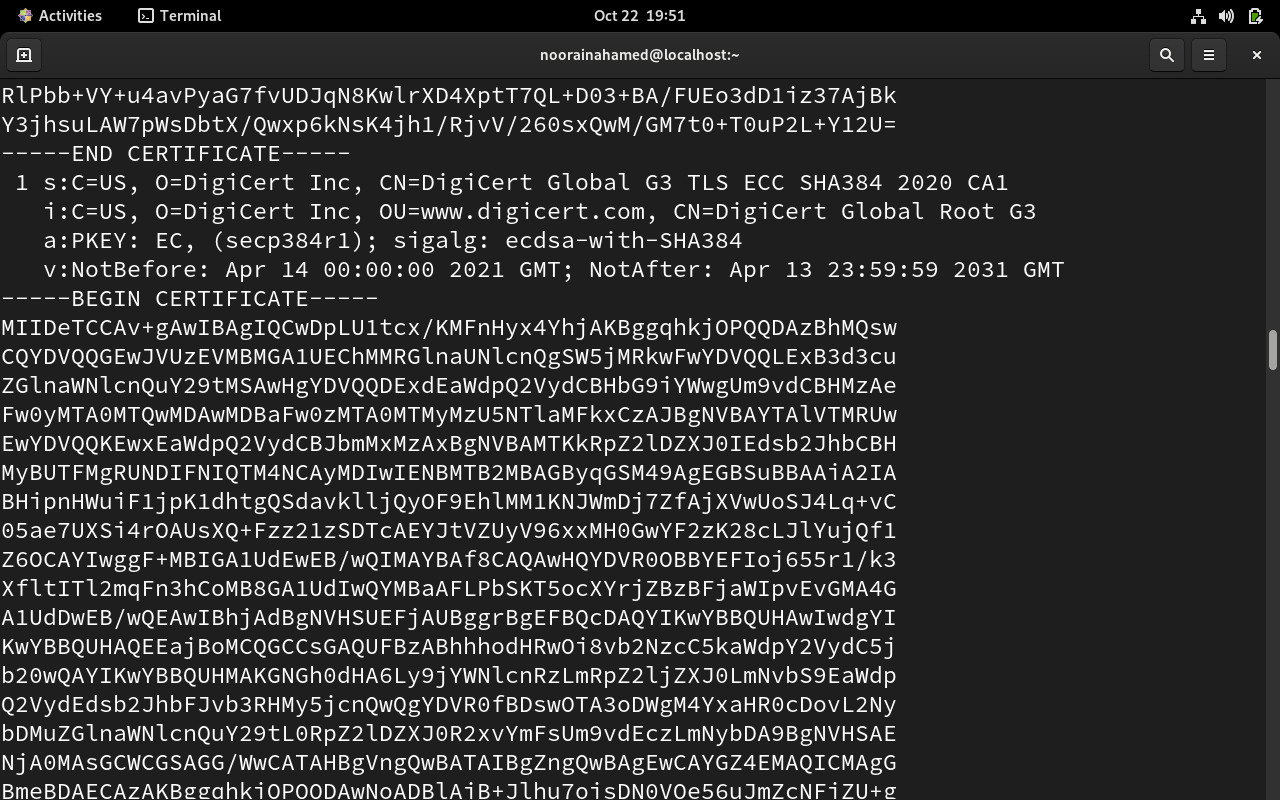


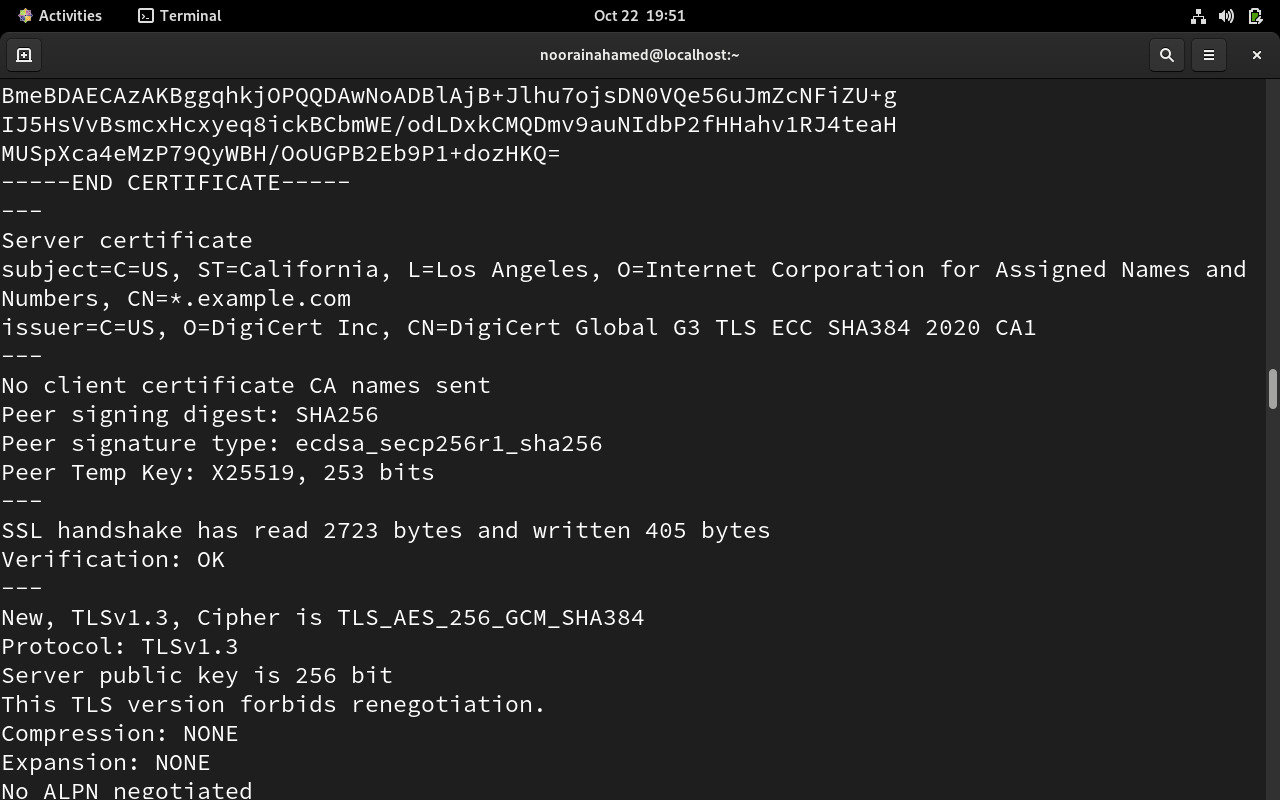


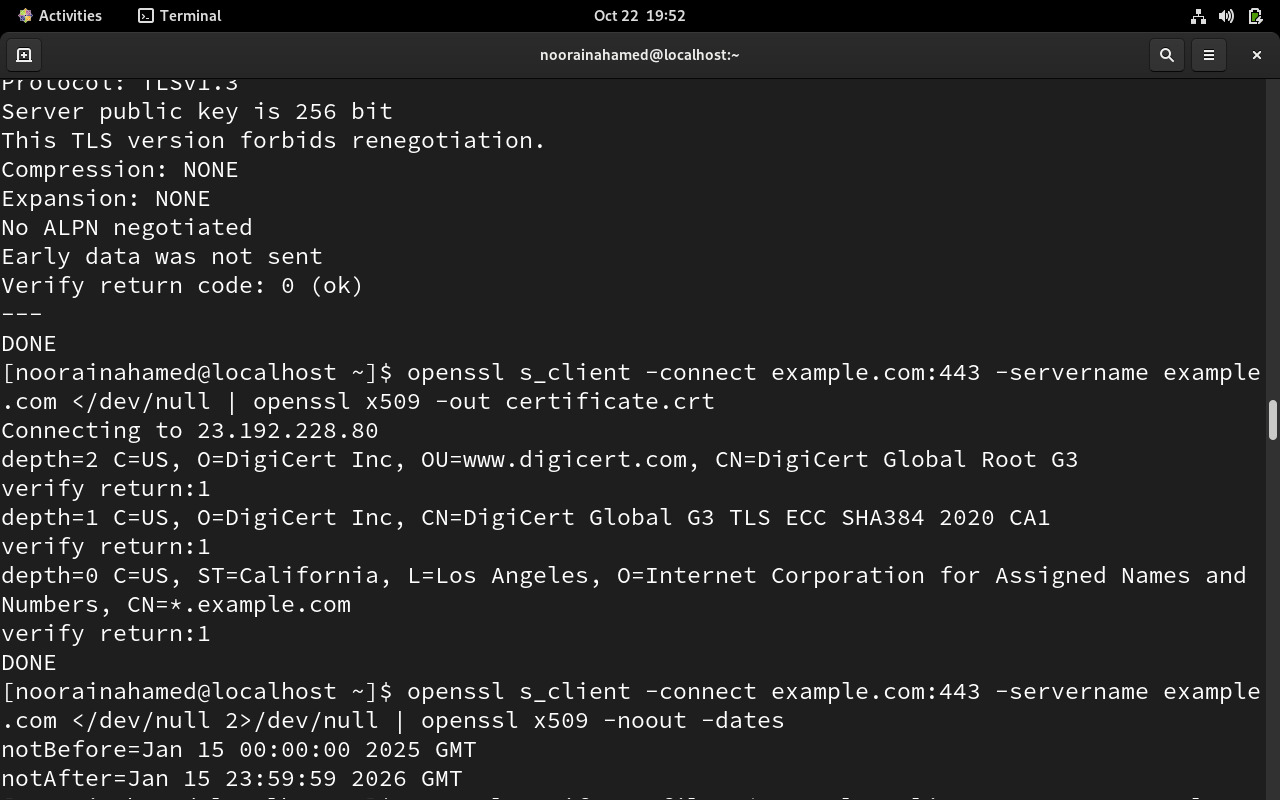












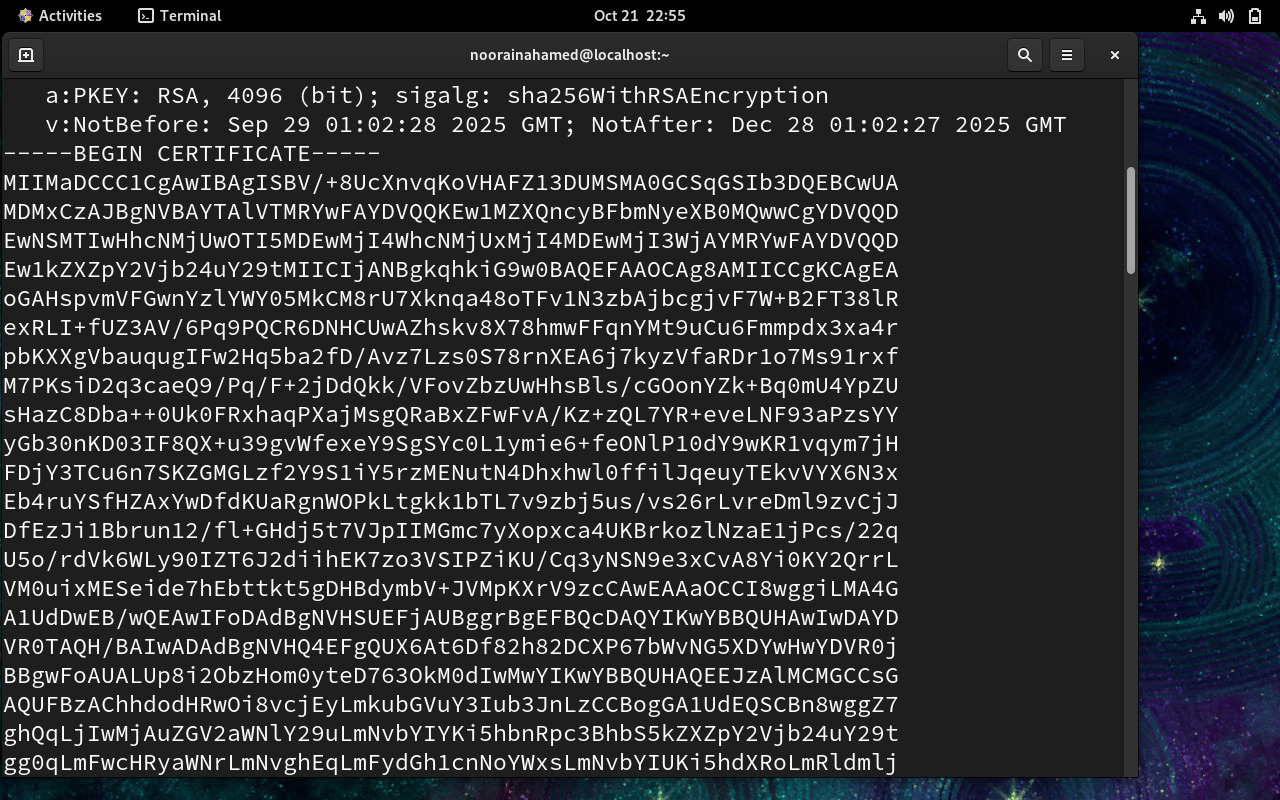
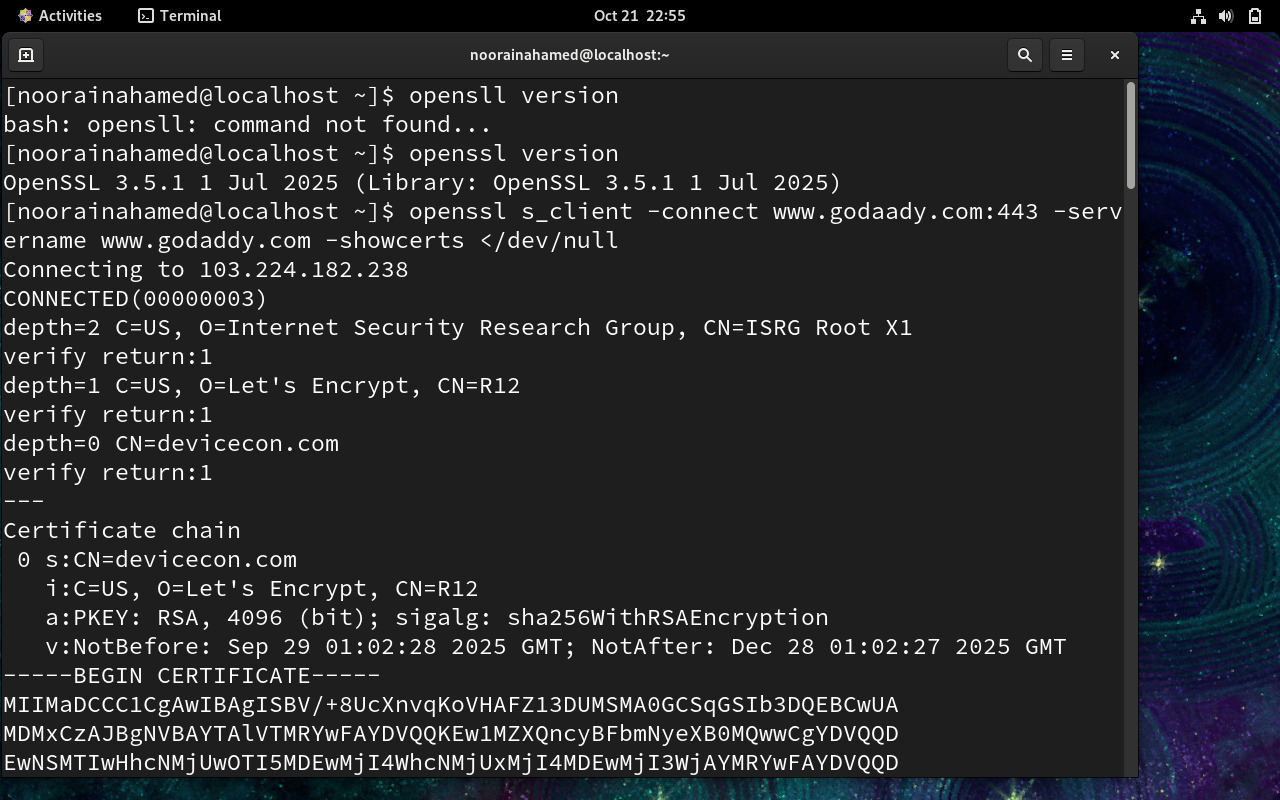
**EXAMPLE – website ( godaddy)**

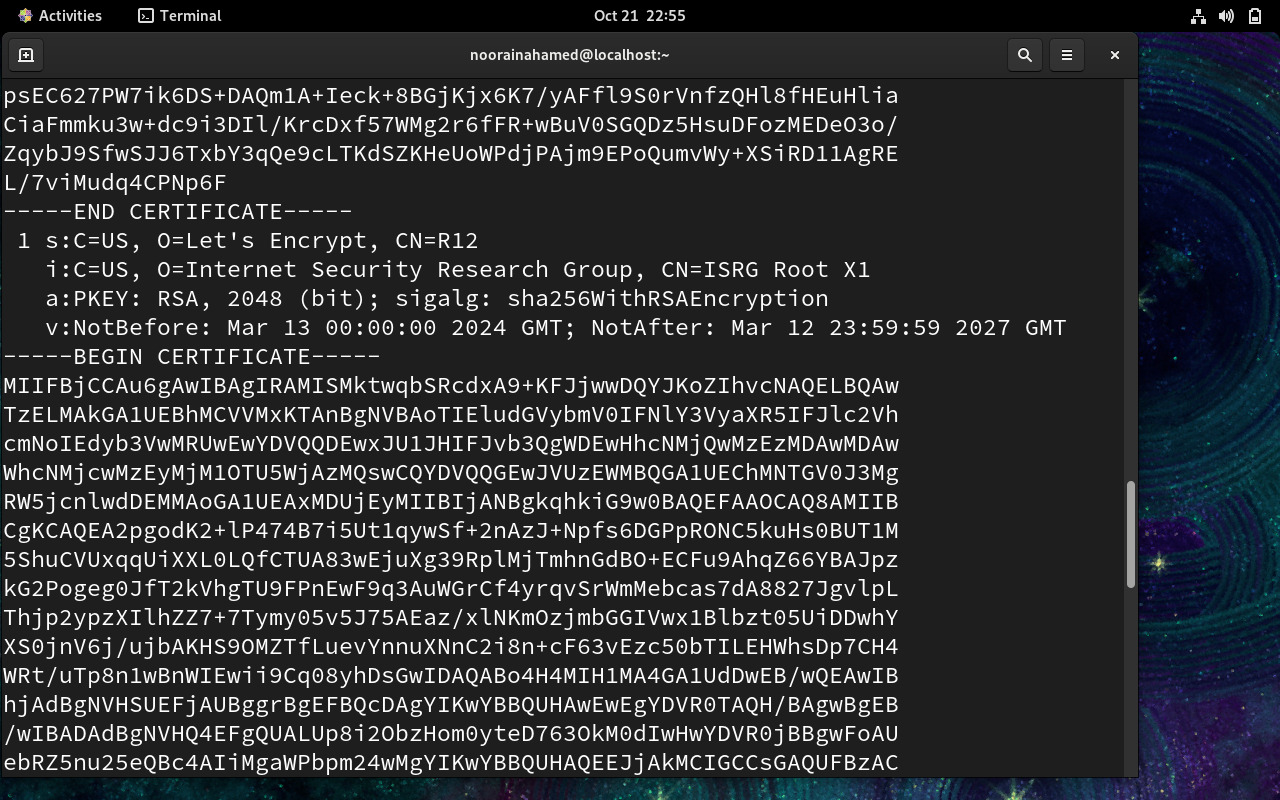
**Verify the SSL certificate (and trust chain)** of the **GoDaddy website** on **CentOS Linux** using **OpenSSL**.

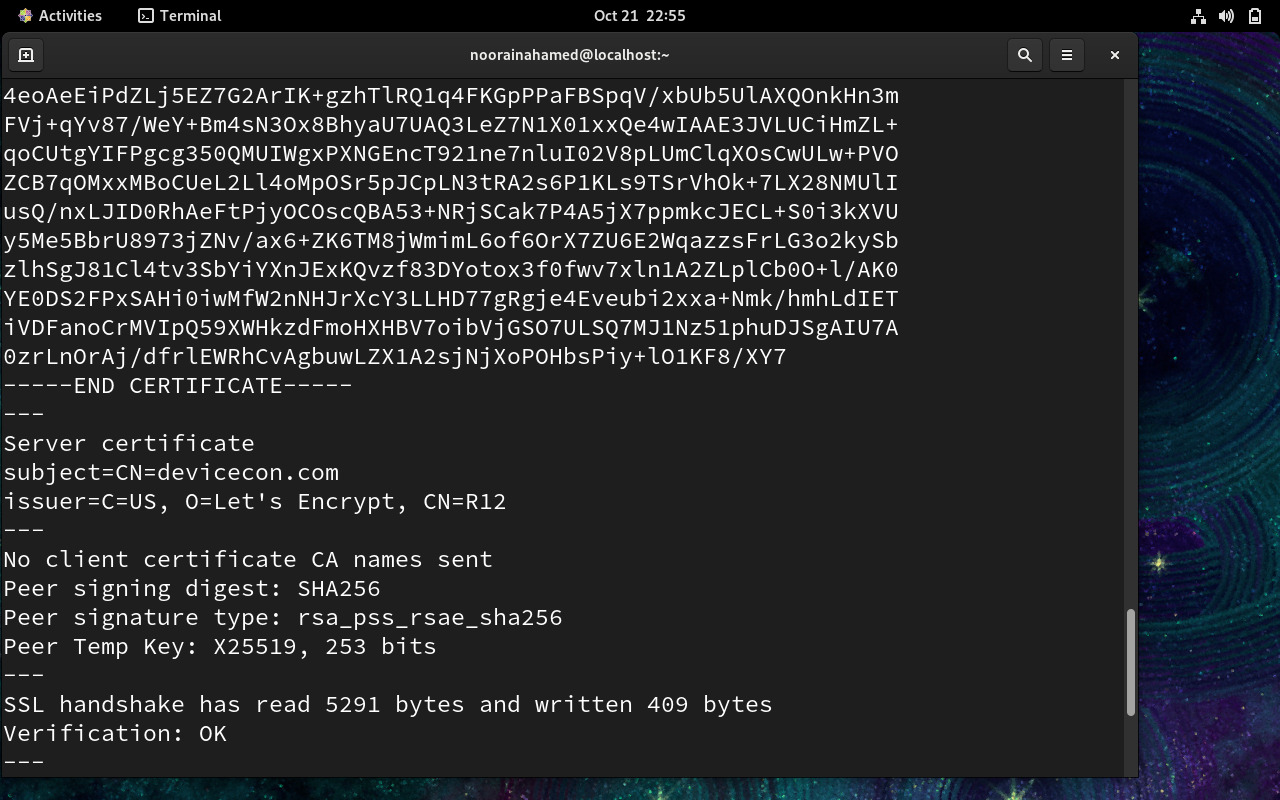
## Fetch and display GoDaddy’s HTTPS certificate chain

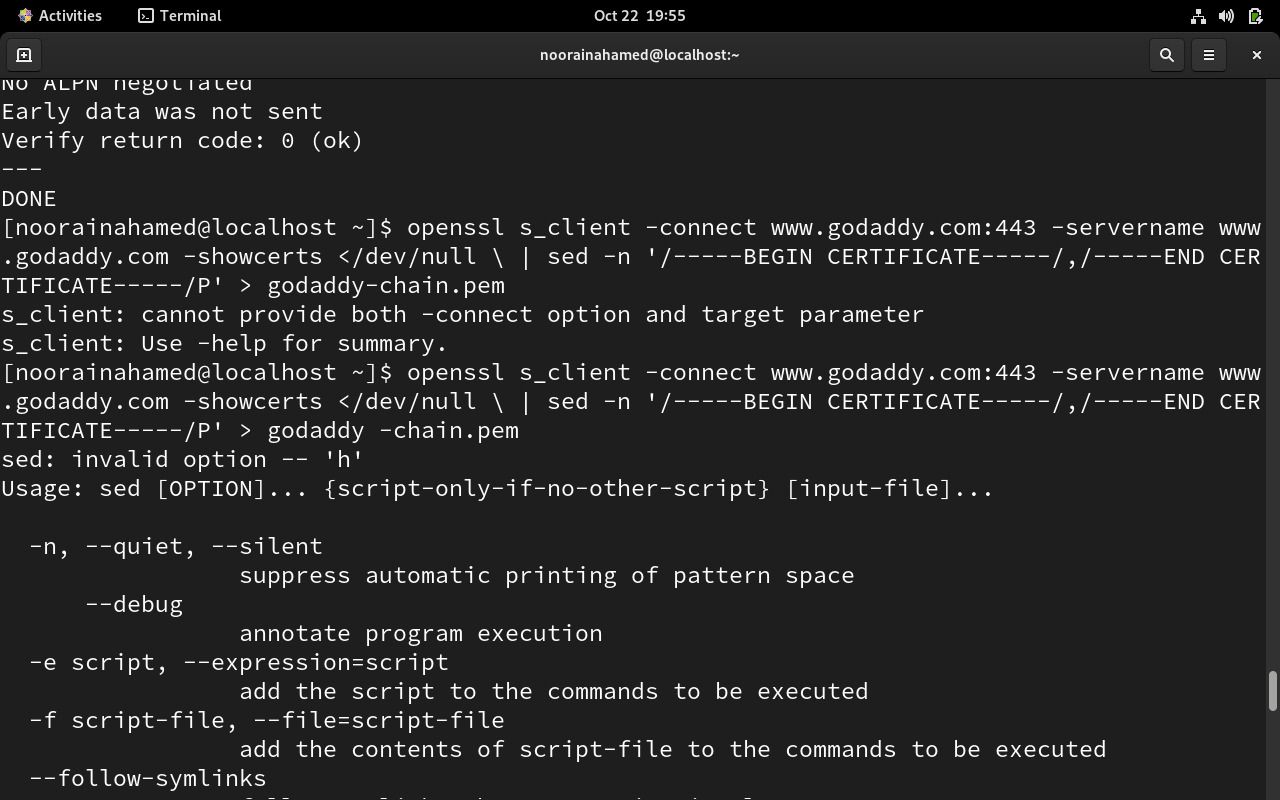
Run this command:

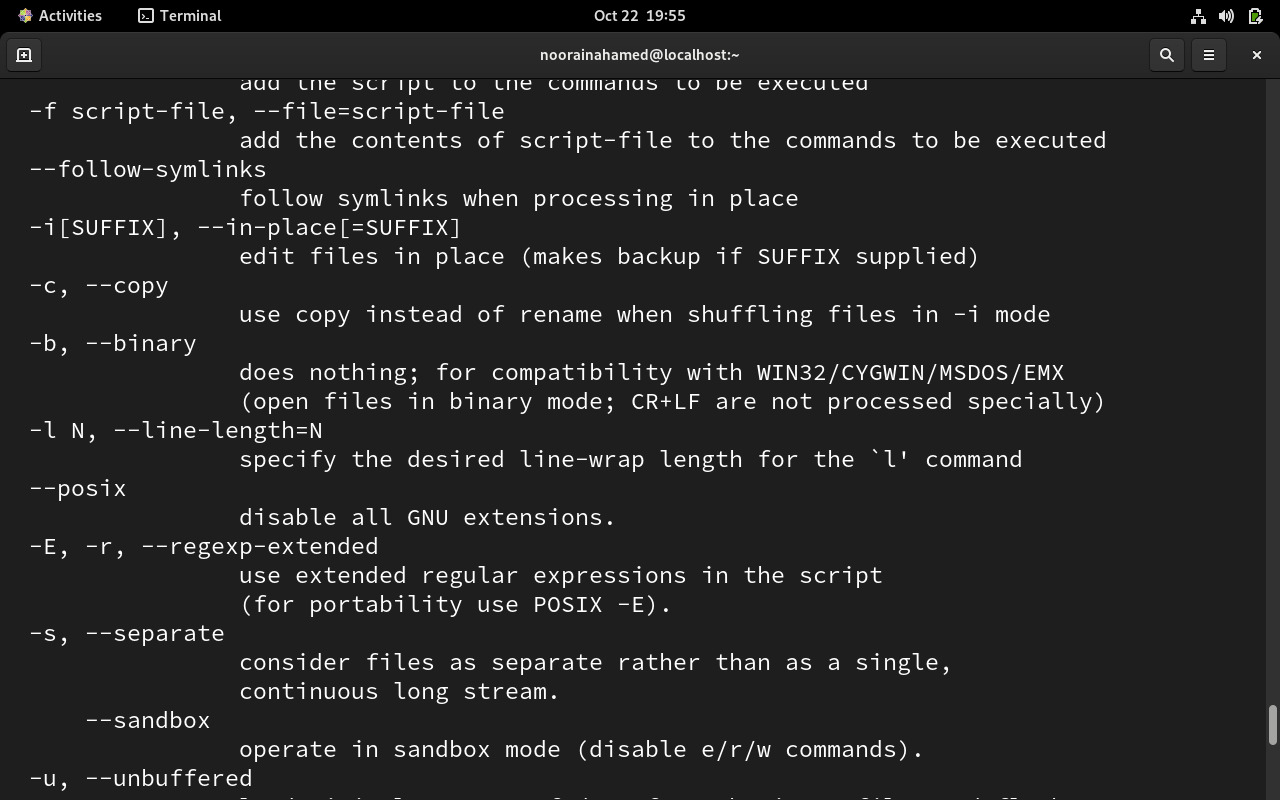
openssl s\_client -connect www.godaddy.com:443 -servername www.godaddy.com -showcerts </dev/null

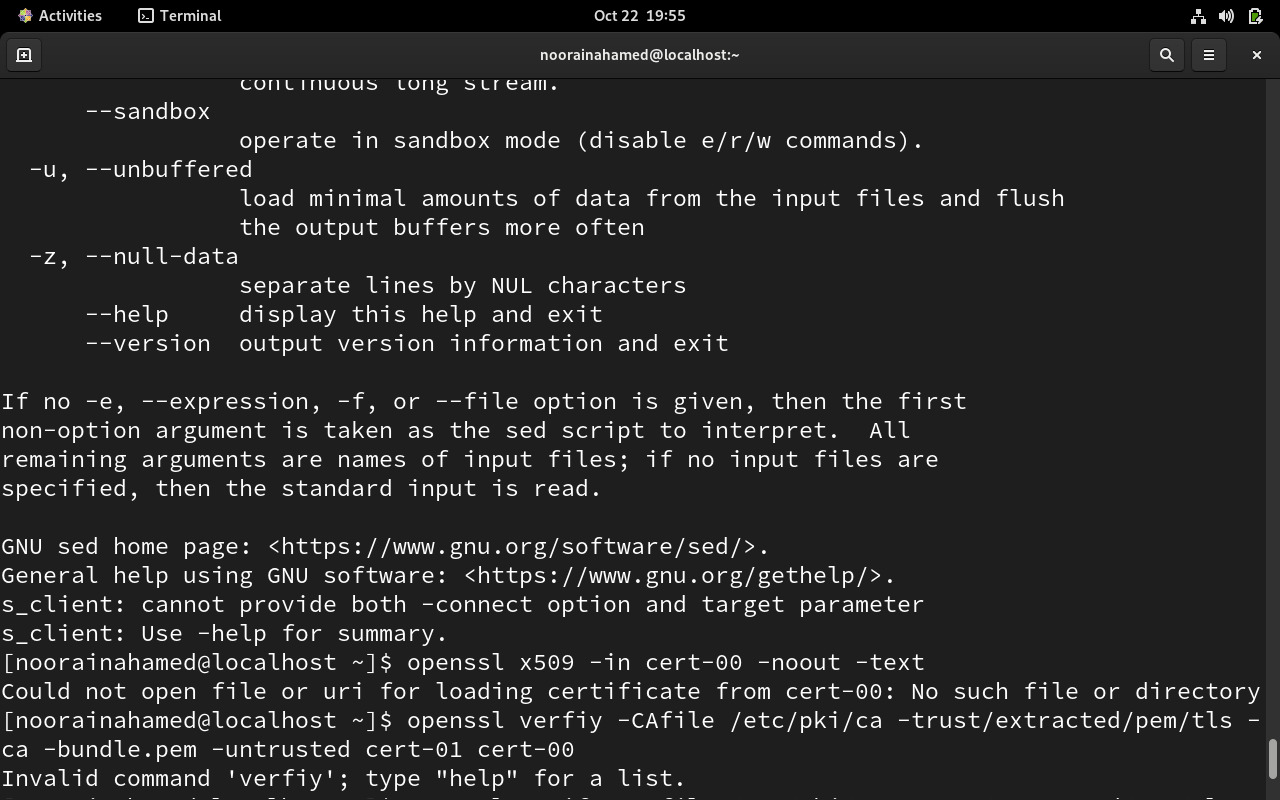












## Using your web browser (fastest way)

### **🧩 Google Chrome / Microsoft Edge**

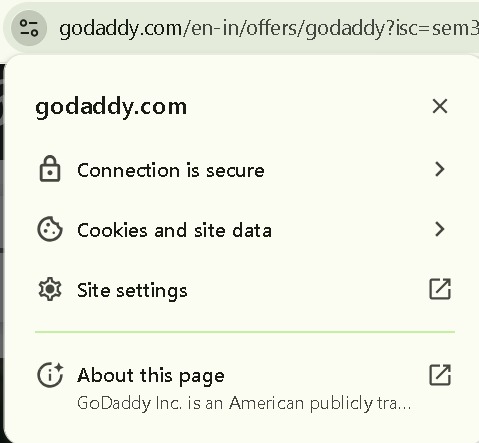
1. Open your browser.
2. Go to:  
   👉 **https://www.godaddy.com**
3. Click the **padlock icon** 🔒 in the address bar.
4. Select **Connection is secure** → **Certificate is valid**.
5. A window opens showing:
   * **Issued to** → www.godaddy.com (the website’s domain)
   * **Issued by** → e.g. Go Daddy Secure Certificate Authority – G2 (the CA)
   * **Valid from / to** → dates of validity
6. Click the **Details** tab to view:
   * Subject (the domain)
   * Issuer (CA)
   * Validity period
   * Subject Alternative Names (SANs)
   * Certification Path (Root → Intermediate → Leaf)

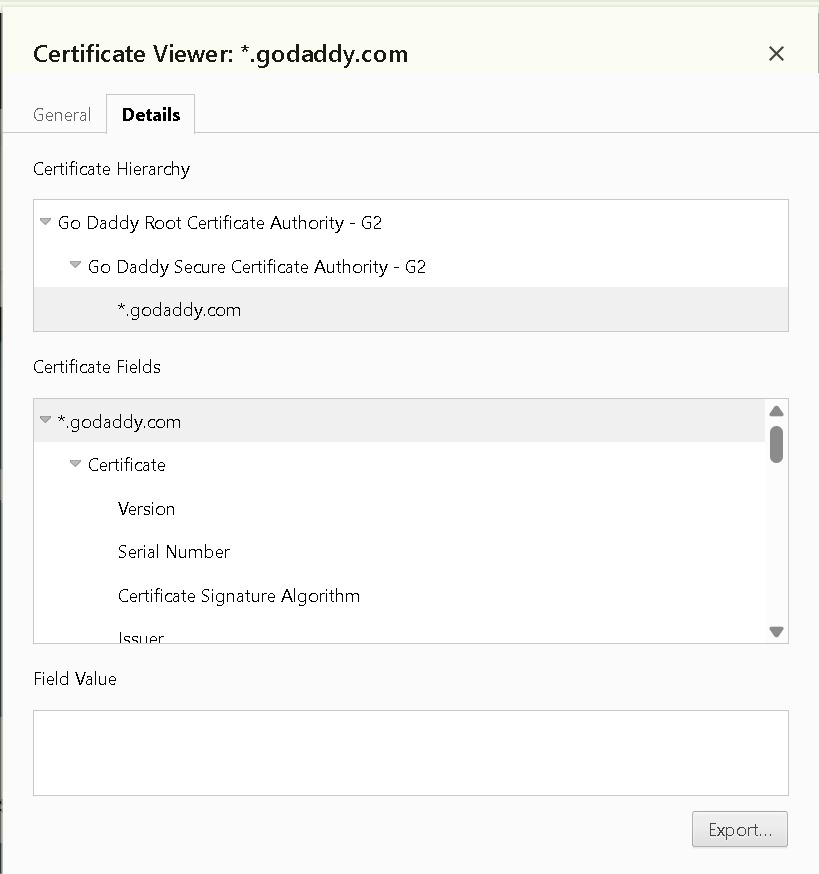
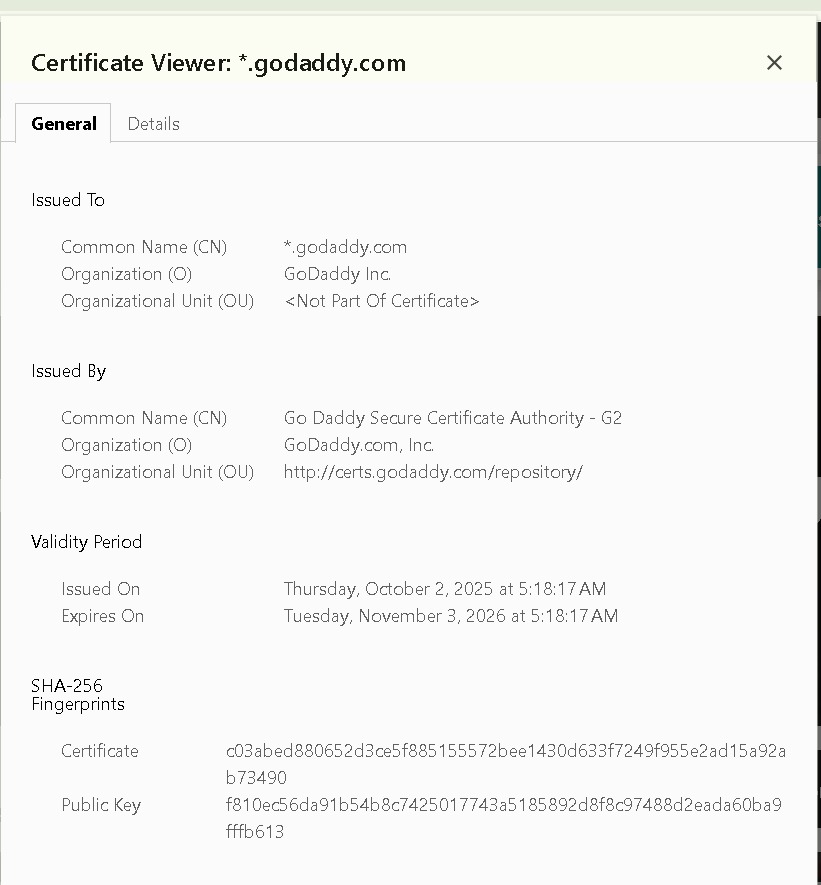
#### Chrome/Edge:

1. Press F12 or Ctrl+Shift+I
2. Go to **Security** tab
3. Click **View certificate**
4. Navigate through certificate details

#### Firefox:

1. Press F12
2. Go to **Security** tab
3. Click **View Certificate**





## \*Server Certificate\*

## The server certificate is issued to a specific website domain, such as www.example.com, and is used to establish a secure connection between the user's browser and the server. It contains the site's public key, validity period, and identity details. When a user visits the site, their browser checks this certificate to ensure the site is authentic and encrypted.

## \*Intermediate Certificate\*

## Intermediate certificates act as a bridge between the server certificate and the trusted root certificate. They are issued by a root Certificate Authority (CA) and used to sign server certificates. This layered approach adds security, allowing root certificates to remain protected while still enabling widespread trust through intermediates.

## \*Root Certificate\*

## Root certificates are the foundation of the trust chain. They are self-signed and pre-installed in browsers and operating systems. Issued by trusted Certificate Authorities like DigiCert or Let's Encrypt, root certificates validate intermediate certificates and ultimately confirm the authenticity of the server certificate. If the root is trusted, the entire chain is trusted

## 6. Summary

### Key Takeaways:

1. **Certificate Hierarchy**: Understand the three-tier system (Root → Intermediate → End-Entity)
2. **Validation Process**:
   * Browser checks certificate validity period
   * Verifies digital signature using issuer's public key
   * Confirms domain name matches
   * Checks revocation status (CRL/OCSP)
3. **Common Certificate Fields**:
   * **Subject**: Entity the certificate belongs to
   * **Issuer**: Certificate Authority that signed it
   * **Validity Period**: Start and expiration dates
   * **Public Key**: Used for encryption
   * **Signature Algorithm**: Hashing and encryption method
4. **Troubleshooting Common Issues**:
   * **Expired Certificate**: Check validity dates
   * **Name Mismatch**: Verify Subject Alternative Names
   * **Untrusted Root**: Ensure root CA is in trust store
   * **Chain Incomplete**: Missing intermediate certificates
5. **Security Best Practices**:
   * Regularly monitor certificate expiration
   * Use strong encryption algorithms
   * Ensure proper chain configuration
   * Implement certificate transparency monitoring