Project 6 Self Signed Server

FAQ/update

- Mar 4: Project released

Overview

TLS protocol supplies secure communication between a client and a server. TLS uses a handshake process to establish certain criteria for the stateful TLS session.
In this project, you are going to create a self signed server using go. You will use the certificate to build and run a https server. Different from other projects, this project will be a self-learning project: You will follow a tutorial and implement the project.

Tutorial

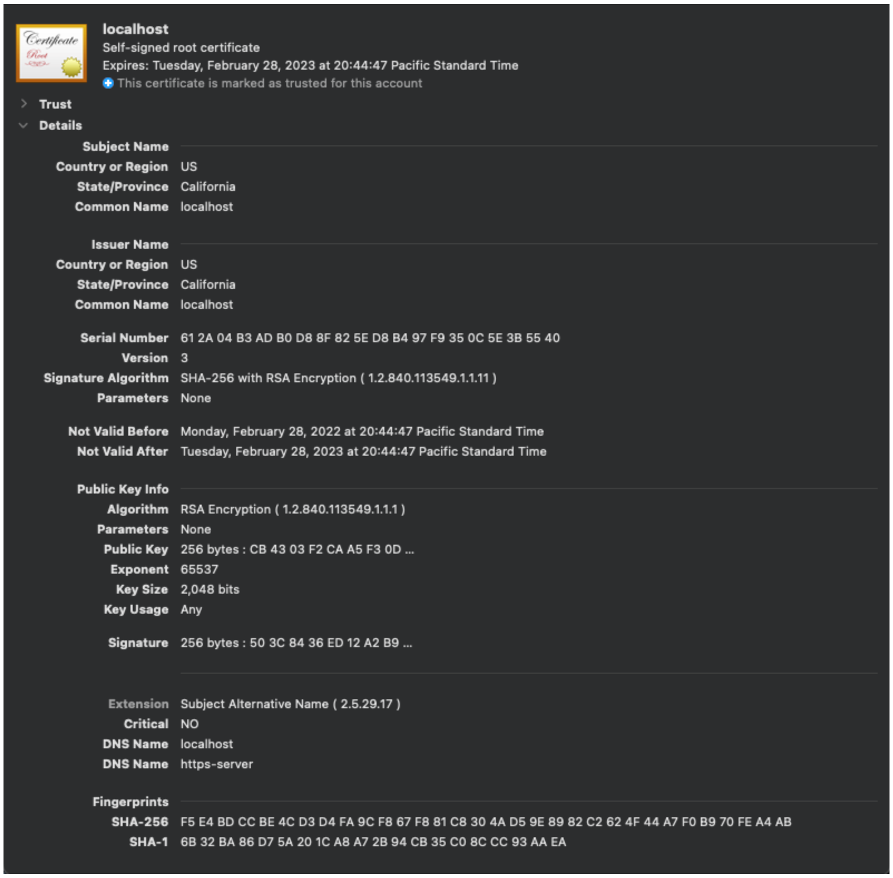
https://luizlelis.com/blog/go-lang-self-signed

Note:

1. Make sure you have docker and docker compose installed. https://docs.docker.com/engine/install/ https://docs.docker.com/compose/install/
2. If you want to test the server with a client running using VM (e.g. AWS), you need to first trust the server certificate in your local trust store (MacOs, windows, Linux).Then you could run the command “docker-compose up server” in VM. In a browser, type “ https://<Public IPv4 address>:8081/home”. You will see the response message.

Submission Format

You do not need to submit any code to the gradescope. Please screenshot the certificate you generated and upload to gradescope in a **PDF** format.



localhost
Self-signed root certificate
Expires: Tuesday, February 28, 2023 at 20:44:47 Pacific Standard Time
This certificate is marked as trusted for this account

> Trust

⌄ Details

Subject Name	
Country or Region	US
State/Province	California
Common Name	localhost
Issuer Name	
Country or Region	US
State/Province	California
Common Name	localhost
Serial Number	61 2A 04 B3 AD B0 D8 8F 82 5E D8 B4 97 F9 35 0C 5E 3B 55 40
Version	3
Signature Algorithm	SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters	None
Not Valid Before	Monday, February 28, 2022 at 20:44:47 Pacific Standard Time
Not Valid After	Tuesday, February 28, 2023 at 20:44:47 Pacific Standard Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	None
Public Key	256 bytes : CB 43 03 F2 CA A5 F3 0D ...
Exponent	65537
Key Size	2,048 bits
Key Usage	Any
Signature	256 bytes : 50 3C 84 36 ED 12 A2 B9 ...
Extension	Subject Alternative Name (2.5.29.17)
Critical	NO
DNS Name	localhost
DNS Name	https-server
Fingerprints	
SHA-256	F5 E4 BD CC BE 4C D3 D4 FA 9C F8 67 F8 81 C8 30 4A D5 9E 89 82 C2 62 4F 44 A7 F0 B9 70 FE A4 AB
SHA-1	6B 32 BA 86 D7 5A 20 1C A8 A7 2B 94 CB 35 C0 8C CC 93 AA EA

MacOS - Example

Optional project extensions

The below extensions are not required for the course, and there is no extra credit offered. However achieving “100% completion” on this assignment will give you bragging rights and an impressive demo to show to recruiters.

In Project 4, you built a DropBox clone called “SurfStore”. There are several other features we could add to the Surfstore.

Option1: Creating a TLS-enabled gRPC server (book Page 286)

gRPC supports the ability to encrypt each of the RPC calls for security purposes. Project 6 gives you some insight about how TLS works. This time you could apply it to the surstore we implemented in the previous project. Before you could add any TLS support, you need to have certificates generated. On page 256 of the textbook (Network Programming with Go), it states how to use go's standard library to generate your own certificate. Then you could add TLS support to the server. On Page 286, there is a section about creating a TLS-Enable gRPC server. It illustrates the way to add a server's key pair and create a new TLS listener. You might also want to go over the section on Page 289 to see how to test the server.

Option2: A web-based client interface for surfstore

The Network Programming With Go book talks about how to support file upload through Go's in-built webserver. Using this feature, and your project 4 client code, write a web-based interface to SurfStore. Through this interface, you should be able to click on files and download the contents via the web, and also upload new files into surfstore from the web rather than the command line.

###