Public-Key Infrastructure Lab

Telle Miller

Liberty University – CSIS 463

10/13/2021

**Introduction**

The goal of this lab is to demonstrate the pitfalls of public-key infrastructure (PKI). When one side of the PKI exchange is captured, namely one of the public keys, the system becomes susceptible to man-in-the-middle attacks. This lab gives a deeper understanding of the Certificate Authority system and how certificates are issued. It also introduces a hands-on look at the X.509 standard and issuing a certificate to an HTTPS website. This lab involved creating a Certificate Authority, generating certificates, establishing a server/webpage, issuing certificates, and performing a man-in-the-middle attack.

**Setup**

As with the previous labs, I completed this lab using the resources provided by SEED labs. This included their pre-built Ubuntu 20.04 VM and the setup files outlined in the lab PDF. This lab also required an Apache webserver to host the real and fake webpages. The setup for this webserver was outlined in the lab guide and was constructed inside of a Docker container.

**Tasks**

Task 1: Becoming a CA Authority

The goal of this task was to set up my system as a Certificate Authority (CA) so that I can issue certificates for the rest of the tasks. This tasked involved changing some information in the *openssl.cnf* files and creating a new key. This process is documented below.

Text

Description automatically generated

After establishing myself as the CA, I used the command demonstrated in the images below to inspect the decoded content of my X509 certificate and the RSA key. This also displayed all my components required in the RSA algorithm, namely the public/private exponent, modulus, and two secret numbers.

Text

Description automatically generated

Text

Description automatically generated

Task 2: Generating a Certificate Request for Your Web Server

The goal of this task was to generate a public-key certificate from our CA for a company named “miller2021.com”. This involved generating a certificate signing request (CSR) for the company, which is then sent to the CA, who then returns a signed certificate. The images below show the process of creating the CSR and verifying the certificate information.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

The X.509 specification also defines extensions to be attached to a certificate in the case that a company has many URLs that are similar and point to the same location. This extension is called a subject alternative name (SAN). Using a similar command to before, I added alternate site names to be signed by the CA, which is shown below.

Text

Description automatically generated

Task 3: Generating a Certificate for your server

The goal of this task was to generate a certificate for the server I set up at the start of the lab. To ensure all the URL extensions were covered by the certificate, I had to change the *opensll.cnf* file, as shown below.



After this change, I generated the certificate for the server using the *openssl ca* command shown below. The second image displays the X509 alternative names for the server, indicating that the generation successfully copied the extensions.

Text

Description automatically generated



Task 4: Deploying Certificate in an Apache-Based HTTPS Website

In this task, the goal is to translate the certificates we generated as the CA earlier into usable certificates for an HTTPS website. This is done using an Apache-based HTTPS website inside of the Docker container. After adjusting some settings in the Apache configuration file, I started the server using the command shown below. This command requires the password for each certificate listed in the configuration file.

Text

Description automatically generated

Now that the server is live, I tried to navigate to the webpage, but was presented with this error. This was expected, as the CA (me) who issued the certificates for the website was not set as “Trusted” in the preferences of the Firefox browser. After adding the CA I created earlier to the accepted list of CAs in Firefox, I was able to successfully browse to the website, as shown in the last image.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Task 5 & 6: Launching a Man-In-The-Middle Attack

The goal of these tasks was to launch a man-in-the-middle (MITM) attack between myself and the server I created. During a MITM attack, the secret sent from the client to the server is intercepted and encrypted using the attacker’s secret. This allows the attack to unencrypt the secret returned from the server, which can result in loss of information. Task 4 involved setting up the system in which I can intercept communication between my system and the server being hosted by the Docker container. This was a simple process and involved following the steps in Task 4 to create a decoy website named [*www.example.com*](http://www.example.com). I used the *index\_red.html* file to indicate that this is a malicious website, as shown below.

Graphical user interface, application

Description automatically generated

For task 6, the goal is to create a MITM attack that does not raise any flags in the browser or alert that the attack has taken place. The setup for this task is the same as task 5, but I swapped the *index.html* file so that the two sites are indistinguishable.

**References**

Delfs, Hans, and Helmut Knebl. *Introduction to Cryptography: Principles and Applications.* Springer, 2007.

Du, W. (2019). *Computer & internet security: A hands-on approach* (2nd ed.). Wenliang Du.

*"In Our Time: Cryptography".* BBC Radio 4. Retrieved 11 October 2021.

Fruhlinger, Josh (29 May 2020). *"What is PKI? And how it secures just about everything online"*. CSOOnline. Retrieved 13 October 2021.

Chokani, S. (2003, November). *Internet X.509 Public Key Infrastructure*. The Internet Society. Retrieved from https://www.ietf.org/rfc/rfc3647.txt.