HACETTEPE UNIVERSITY
ENGINEERING DEPARTMENT
COMPUTER ENGINEERING

LESSON
BBM 465 INFORMATION SECURITY LAB.

ADVISOR
Yasin Şahin, Işıl Karabey

NAME :	Hasan Hüseyin TOPÇU	-	212228764
	Taha BAŞKAK	-	21228104

GROUP NO : 30

TECHNICAL INFORMATION

Server and Client Defination

Server : A generic name given to hardware or software that distributes a program or information on any network to different users or systems. The servers must be connected to a computer network. The servers must be in continuous operation due to the task of storing and distributing information. Server is the opposite of the word is Client. **[1]**

Client : A request from a server over a network is one that can use the server's data to the extent allowed by the server. The client can give you the web site as an example. **[2]**

SSL/TSL

Secure Sockets Layer (SSL) is a security protocol used to secure the data communication between the server and the client. This protocol has been developed over time and has been TLS (Transport Layer Security) with name change.

SSL / TLS provides consistent and secure data communication over the network in server-client applications by avoiding listening, stealing, or modifying data. SSL / TLS ensures that the information sent is strictly decipherable only in the correct address. Information is automatically encrypted before it is sent and can only be decrypted by the correct recipient. Verification is done on both sides to protect both the confidentiality and integrity of the process and the information.

They use X.509 certificates in SSL / TLS and therefore the identity of those communicating with the other party is done by asymmetric encryption and understood on a symmetric key. This session key is then used to encrypt the data stream between the parties.

Extensive applications like Internet Explorer, Mozilla Firefox, Opera, Safari SSL is used for secure data communication in millions of websites. The most common use is to enable the communication between the SSL server and the client, which is in the web environment on the internet. **[3]**

SNI

SNI allows us to publish multiple certified websites over a single IP. To summarize the SNI operation logic, a TLS attachment SNI sends a host header request to IIS, meaning that at the beginning of a handshake operation, it recognizes which web site is being called, and after the public key client sends the request, it verifies the certificate and multiple certificates are now available. **[4]**

HTTPD (APACHE2)

Developed by the Apache Software Foundation (ASF). It is an open source and free web server program and is free. The Apache server is a kind of parser for those who want to create a security system on the web server. This program is a free program that allows you to create your web server, send your internet directory to this presentation, and create your directory.

Unix, Windows, Unix ... etc. can run on operating systems. Since 1996 Apache has been the most popular server on the Internet. **[5]**

mod_ssl and openSSL

Mod_ssl was developed by Eric A.Young and Tim J. Hudson and is a mod. It provides strong encryption for Apache 1.3 web server via Secure Socket Layer (SSL v2 / v3) and Transmission Layer Security (TLS v1) protocols, with SSLeay based OpenSSL.

This module is needed to perform HTTPS requests to the web server. Apache patches the source code and expands its application development interface (API), the so-called extended application development interface (EAPI).

OpenSSL is an open source implementation of SSL and TLS protocols. The main library written in C programming language implements basic cryptographic functions. There are also intermediate software developed to use OpenSSL with different programming languages. **[6]**

Certificate Authority

The Certification Authority (CA) is the authority to issue certificates to persons or entities. As an example for the Certification Authority organization Verisign, Globalsign, Comodo… etc.

If both parties are using a certificate signed by a jointly trusted certification authority, they can trust each other's public keys. In this trust relationship, one is the third party that the certificate holder and the trusting party can trust. So it's a reliable tool. The most common public key infrastructure (PKI) schemes are schemes used to implement HTTPS on the web worldwide. All of these are based on the X.509 standard and feature CAs. **[7]**

x.509

X.509 was originally published on July 3, 1988 and started in connection with the X.500 standard. Then, by adding certifications, 3 kinds of x.509 standards were created as V1, V2, V3. V3 includes features in two other versions.

X.509 is an important standard for a public key infrastructure that manages the TLS protocol, which is used to secure digital certificates, public key cryptography, and web communications. X.509 sets public key certificates, certificate revocation lists, certificate properties, and the public form for the certificate validation algorithm. **[8]**

EXPERIMENT PHASES

This experiment was conducted in Ubuntu 15.10, the Linux version. The following commands and configurations are shown in Ubuntu 15.10 operating system. The mod_ssl and httpd installations mentioned above are done by installing apache2 because apache2 has mod_ssl and openssl is ssl on this operating system.

Server Side

Set Up and Publish Two Web Sites

sudo apt install apache2
#install apache2

sudo mkdir -p /var/www/websites/site1/public_html
#create “websites/site1/public_html” folders in “/var/www path”. This is our first publish website #folders.

sudo mkdir -p /var/www/websites/site2/public_html
#create “websites/site2/public_html” folders in “/var/www path”. This is our second publish #website folders.

sudo vi /var/www/websites/site1/public_html/index.html
#create index.html for site1 using vi editor and write sample website html code. For example site1:

```
<!DOCTYPE html>
<html>
        <head>
                <title>www.site1.com</title>
        </head>
        <body>
                <p>It work! Welcome to www.site1.com</p>
        </body>
</html>
```

sudo vi /var/www/websites/site2/public_html/index.html
#create index.html for site2 using vi editor and write sample website html code. For example site2:

```
<!DOCTYPE html>
<html>
        <head>
                <title>www.site2.com</title>
        </head>
        <body>
                <p>It work! Welcome to www.site2.com</p>
        </body>
</html>
```

sudo chown www-data:www-data /var/www/websites

sudo chmod -R 755 /var/www/websites
#provide permissions for "/var/www/websites".

sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/site1.conf
sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/site2.conf
#create site1.conf and site2.conf in "/etc/apache2/sites-available/" path. Actually, copy
#000-default.conf file to site1.conf and site2.conf.

sudo vi /etc/apache2/sites-available/site1.conf
#open site1.conf file using vi editor and write these configurations:

```
<VirtualHost *:80>
        ServerAdmin webmaster@site1
        ServerName site1.com
        ServerAlias www.site1.com
        DocumentRoot /var/www/websites/site1/public_html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

sudo vi /etc/apache2/sites-available/site2.conf
#open site2.conf file using vi editor and write these configurations:

```
<VirtualHost *:80>
        ServerAdmin webmaster@site2
        ServerName site2.com
        ServerAlias www.site2.com
        DocumentRoot /var/www/websites/site2/public_html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

sudo ln -s /etc/apache2/sites-available/site1.conf /etc/apache2/sites-enabled
sudo ln -s /etc/apache2/sites-available/site2.conf /etc/apache2/sites-enabled
#create symbolic links for site1.con and site2.conf in "/etc/apache2/sites-enabled".

sudo a2dissite 000-default.conf
sudo a2ensite site1.conf
sudo a2ensite site2.conf
#disables the default site and enable site1 and site2 using the these commands.

sudo vi /etc/hosts
#open hosts file using vi editor and write ip address that can be changed according to the connected
#internet address for site1.conf and site2.conf

192.168.2.152   site1.com
192.168.2.152   site2.com

sudo service apache2 restart
#restart apache2

Setting Up The Certification Authority (CA)

su
#be root user.

mkdir /root/CA
#create CA file in "/root" path.

cd /root/CA
#enter "/root/CA" path.

mkdir newcerts certs crl private requests
#create newcerts, certs, crl, private and requests folders.

touch index.txt
echo '01' > serial
#create some necessary files for Certification Authority.

openssl genrsa -des3 -out private/cakey.pem 4096
#generate CA's private key using openssl.
#used encryption algorithm is RSA and des3 and 4096 bit long.
#created cakey.pem file is under private folder.
#want to password from us and enter password for cakey.pem

openssl req -new -x509 -key private/cakey.pem -out cacert.pem -days 3650 -set_serial 0
#use the root key (cakey.pem) to create a root certificate (cacert.pem) using x509 and openssl.
#the root certificate expiry date is 3650 days.
#once the root certificate expires, all certificates signed by the CA become invalid.
#and root CA informations are these:

Country name = TR
common name = Kardeşler CA
email = hasanhuseyintopcuu@gmail.com
others are optional so we enter nothing its empty

vi /etc/ssl/openssl.cnf
#enter openssl.cnf file and edit "dir" line. Make "dir= /root/CA".

chmod -R 600 /root/CA
#limit access rights

<u>This part for site1.com(Certification Sign Request (CSR) and Sign the CSR)</u>

cd /root/CA/requests
#enter "/root/CA/request" path.

openssl genrsa -des3 -out site1key.pem 2048
#generate site1key.pem using openssl.
#used encryption algorithm is RSA and des3 and 2048 bit long.

openssl req -new -key site1key.pem -out site1cert.csr -days 3650
#use site1key.pem to create certificate request that is site1certs.crs for 3650 days.
#enter these informations:

password = ******
country name = TR
comman name = site1 Kardeşler CA

openssl ca -in site1cert.csr -out site1cert.pem
#sign certificate(site1cert.pem) using site1cert.csr file and ofcourse ca sign it.

<u>This part for site2.com(Certification Sign Request (CSR) and Sign the CSR)</u>

cd /root/CA/requests
#enter "/root/CA/request" path.

openssl genrsa -des3 -out site2key.pem 2048
#generate site2key.pem using openssl.
#used encryption algorithm is RSA and des3 and 2048 bit long.

openssl req -new -key site2key.pem -out site2cert.csr -days 3650
#use site2key.pem to create certificate request that is site2certs.crs for 3650 days.
#enter these informations:
password = ******
country name = TR
comman name = site2 Kardeşler CA

openssl ca -in site1cert.csr -out site1cert.pem
#sign certificate(site2cert.pem) using site2cert.csr file and ofcourse ca sign it.

Virtualhost Configuration For Two Website On One Ip

cd /var/www/
#enter "/var/www/" folder.

mkdir certs
#create certs folder under "/var/www/" folder.

cd /root/CA/requests
#enter "/root/CA/request/" folder.

mv site1cert.pem /var/www/certs/
#move site1cert.pem file under "/var/www/certs/" folder.

mv site2cert.pem /var/www/certs/
#move site2cert.pem file under "/var/www/certs/" folder.

vi /etc/apache2/sites-available/site1.conf
#open site1.conf file using vi editor and write virtual host configuraiton for https connection

```
<VirtualHost *:443>
        ServerAdmin webmaster@site1
        ServerName site1.com
        ServerAlias www.site1.com
        DocumentRoot /var/www/websites/site1/public_html

        SSLEngine on
        SSLCertificateFile /var/www/certs/site1cert.pem
        SSLCertificateKeyFile /root/CA/requests/site1key.pem
</VirtualHost>
```

vi /etc/apache2/sites-available/site2.conf
#open site2.conf file using vi editor and write virtual host configuraiton for https connection

```
<VirtualHost *:443>
        ServerAdmin webmaster@site2
        ServerName site2.com
        ServerAlias www.site2.com
        DocumentRoot /var/www/websites/site2/public_html

        SSLEngine on
        SSLCertificateFile /var/www/certs/site2cert.pem
        SSLCertificateKeyFile /root/CA/requests/site2key.pem
</VirtualHost>
```

service apache2 restart

#restart apache2

Client Side

sudo vi /etc/hosts
#enter hosts file and wirte these:

192.168.2.152   site1.com
192.168.2.152   site2.com

#download CA so the "cakey.pem" file in computer.
#import CA(cakey.pem) to any browser.
#and ready to secure connection for two websites.

https://site1.com
https://site2.com

REFERENCES

[1] http://www.netcom.com.tr/index.php/coezuemlerimiz/sunucu/item/48-sunucu

[2] http://www.mehmetkirazli.com/istemci-sunucu-mimarisi/

[3] http://www.bilgisayarsistemleri.net/mail-ve-browser/ssl-ve-tls-nedir-t43.html
https://tr.wikipedia.org/wiki/Transport_Layer_Security
http://blog.btrisk.com/2014/04/ssl-nedir-2bolum_16.html

[4] http://www.yazilimsinifi.com/server-name-indication-sni-nedir/

[5] http://www.dijitalders.com/icerik/30/4671/apache_nedir.html#.VkN42L-vv20
https://tr.wikipedia.org/wiki/Apache_HTTP_Sunucusu
http://www.techgriff.com/nedir/apache-server-nedir-ne-ise-yarar-14877

[6] https://tr.wikipedia.org/wiki/OpenSSL
http://belgeler.org/howto/apache-compile-howto-apache.html

[7] http://csirt.ulakbim.gov.tr/dokumanlar/ssl_sayisal_sertifikalar.pdf
https://en.wikipedia.org/wiki/Certificate_authority

[8] http://www.e-imza.com.tr/questions.php
https://en.wikipedia.org/wiki/X.509#PKI_standards_for_X.509

http://acidx.net/wordpress/2012/09/creating-a-certification-authority-and-a-server certificate-on-ubuntu/
https://jamielinux.com/docs/openssl-certificate-authority/index.html
https://networklessons.com/linux/openssl-certification-authority-ca-ubuntu-server/
https://www.liberiangeek.net/2014/09/run-multiple-websites-single-ubuntu-server-using-apache2/
https://www.liberiangeek.net/2015/07/how-to-enable-and-run-multiple-websites-

using-apache2-on-ubuntu-15-04/