Secure Web Development Teaching Modules[1]

Secure Web Transactions

Contents

1. Concepts

1.1 SSL/TLS

Transport Layer Security (TLS)[2] is a protocol for securing communications among applications over the Internet. TLS secures application layer contents, such as web communications, and is implemented using transport layer protocols, such as TCP. The protocol can provide confidentiality by encrypting the application data sent between a server and its clients and provide authentication of both the server and the clients. Secure Socket Layer (SSL) is the predecessor of TLS.

SSL/TLS is often used together with HTTP, called HTTPS, for securing web transactions. The protocol can also be used with other applications, such as emails, file transfer or virtual private networks (VPN). When using SSL/TLS with IMAP and POP, users can authenticate the email server and encrypt the emails sent between the email server and their email software, such as Mozilla Thunderbird or Microsoft Outlook. However, in this case, the emails are only encrypted over the SSL/TLS channel. The emails are not encrypted when routed through the Internet unless other means of security measures are

[1] Copyright© 2009-2011 Li-Chiou Chen (lchen@pace.edu) & Lixin Tao (ltao@pace.edu), Pace University. This document is based upon work supported by the National Science Foundation's Course Curriculum, and Laboratory Improvement (CCLI) program under Grant No. 0837549. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation. A copy of the license is available at http://www.gnu.org/copyleft/fdl.html.

[2] TLS 1.2 is the latest version, described in IETF RFC 2546 (http://tools.ietf.org/html/rfc5246).

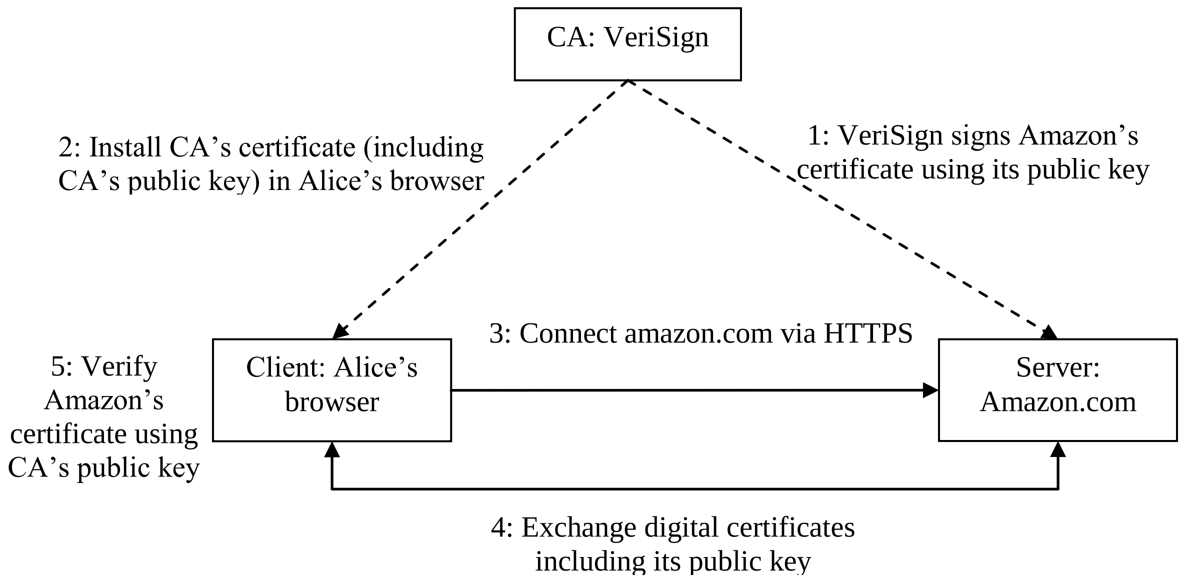
established, such as VPN connections over the infrastructure or PGP encryption over the email content. SSL/TLS can also be used together with FTP for secure file transfer, called FTPS[3]. FileZilla is an example of file transfer software that utilizes FTPS. SSL/TLS has also been applied to build VPN connections. CISCO's AnyConnect VPN utilizes TLS and a modified TLS protocol called DTLS[4] that utilizes TLS on top of UDP datagram.

1.2 HTTPS

HTTPS[5] combines both HTTP and SSL/TLS protocols to provide secure communications between web servers and browsers. HTTPS operates at port 443 instead of the default port 80. It provides encryption of web contents and authentication of the web server. In other words, users will be able to authenticate web sites and encrypt the confidential information sent between their browsers and the web sites through the HTTPS protocol. HTTPS does not apply client authentication so that the web sites cannot authenticate users when they connect through HTTPS. The users must go through additional authentication methods, such as password checking, biometrics or a combination of various user authentication measures.

SSL/TLS communications in HTTPS include two stages: handshaking and data sending. Prior to communications, web sites need to request a certificate authority (CA) to sign its digital certificate which contains the site's public key. Users obtain the CA's digital certificate, called root certificate, when they install their browser. Web browsers such as Internet Explorer or Firefox are installed with hundreds of root certificates from a wide range of companies, such as VeriSign or Entrust, acting as certificate authorities.

As in Figure 1, once a user decides to connect to a web site through HTTPS connection, the web server sends its certificate containing its site public key. The user can then verify the web site's digital certificate using the pre-installed root certificate from the web site's CA.


CA: VeriSign
2: Install CA's certificate (including CA's public key) in Alice's browser
1: VeriSign signs Amazon's certificate using its public key
3: Connect amazon.com via HTTPS
5: Verify Amazon's certificate using CA's public key
Client: Alice's browser
Server: Amazon.com
4: Exchange digital certificates including its public key

[3] FTPS is described in IETF RFC 4217 (http://tools.ietf.org/html/rfs4217).
[4] Modadugu, N., Rescorla, E., The Design and Implementation of Datagram TLS, Proceedings of NDSS 2004, February 2004.
[5] HTTPS is described in IETF RFC 2818 (http://tools.ietf.org/html/rfc2818).

Figure 1: SSL/TLS Handshaking in HTTPS

The second stage of the SSL/TLS communication encrypts data sent between the server and the client based on cryptographic protocols negotiated between the two parties. As in Figure 2, once the site's digital certificate is verified, the browser and the server start to negotiate cipher suite, cryptographic algorithms that can be used by both for encrypting data and verifying digital signatures. If a public key encryption method is chosen, the two parties encrypt information using each other's public key and then decrypt the contents using its own private keys. To save communication time, usually public key encryption is used only for exchanging a session key (a temporary private key) that can be used for data encryption.

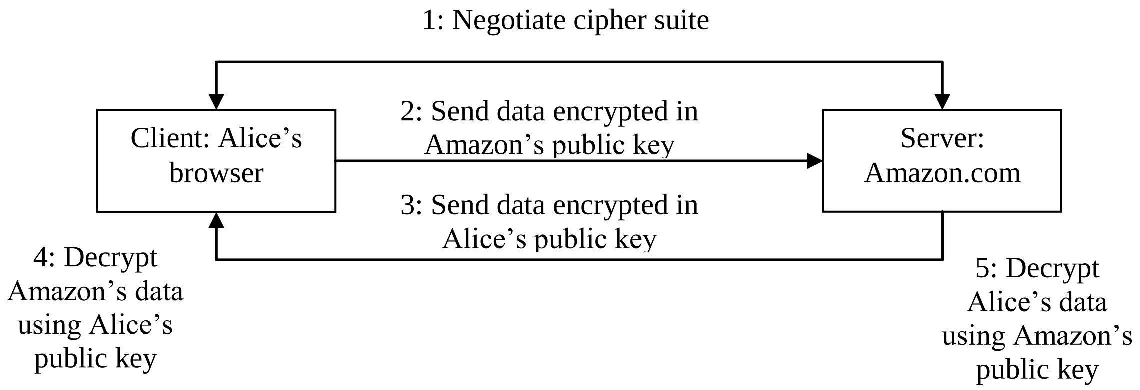
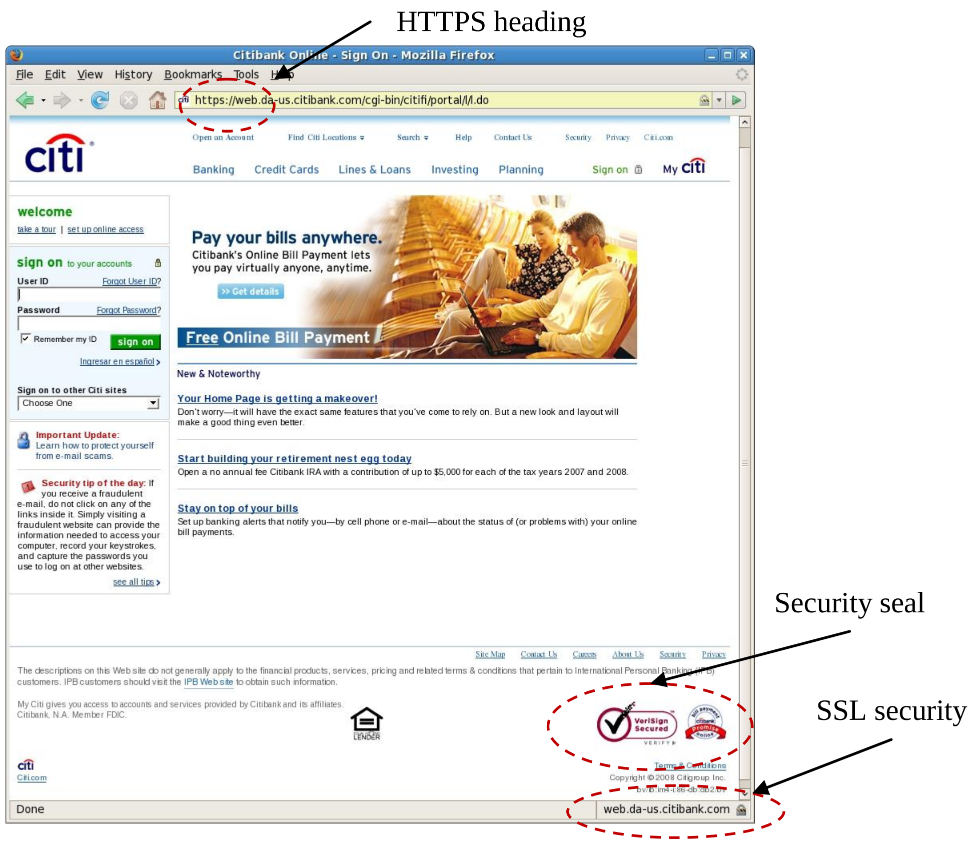

1: Negotiate cipher suite
Client: Alice's browser
2: Send data encrypted in Amazon's public key
Server: Amazon.com
3: Send data encrypted in Alice's public key
4: Decrypt Amazon's data using Alice's public key
5: Decrypt Alice's data using Amazon's public key

Figure 2: SSL/TLS Data Sending in HTTPS

1.3 Site Security Indicators

Most web sites utilize HTTPS to conduct secure transactions with their customers to send sensitive information such as credit numbers, purchase records or personal information. For example, the login pages of most banking sites and the check-out pages of most e-commerce sites are implemented using HTTPS. Users can verify web sites by recognizing site security indicators. They are the indicators for verifying TLS/SSL connections over HTTPS on a web site. These indicators only are displayed when a TLS/SSL session is established between a browser and the web server. Figure 3 illustrates an example. The indicators included the following:

- HTTPS heading in URL that indicates a HTTPS session is established.
- SSL security padlock at the lower right corner of the browser following the domain name of the secure web server. The domain name should be the same as the one in the URL. By double-clicking the padlock, users will be able to see the contents of the digital certificate for the web server.
- Security seal which is an icon that links to the site's certificate issuer, such as VeriSign or Entrust. Once users click on the security seal, they will be prompted with certificate information to verify the web site that displays the seal. A security seal is an additional indicator for users to authenticate the server although a TLS/SSL session can be verified by just the HTTPS heading and the SSL security padlock.



Pay your bills anywhere. Citibank's Online Bill Payment lets you pay virtually anyone, anytime.

New & Noteworthy

Your Home Page is getting a makeover!
Don't worry—it will have the exact same features that you've come to rely on. But a new look and layout will make a good thing even better.

Start building your retirement nest egg today
Open a no annual fee Citibank IRA with a contribution of up to $5,000 for each of the tax years 2007 and 2008.

Stay on top of your bills
Set up banking alerts that notify you—by cell phone or e-mail—about the status of (or problems with) your online bill payments.

Important Update: Learn how to protect yourself from e-mail scams.

Security tip of the day: If you receive a fraudulent e-mail, do not click on any of the links inside it. Simply visiting a fraudulent website can provide the information needed to access your computer, record your keystrokes, and capture the passwords you use to log on at other websites.

The descriptions on this Web site do not generally apply to the financial products, services, pricing and related terms & conditions that pertain to International Personal Banking (IPB) customers. IPB customers should visit the IPB Web site to obtain such information.

My Citi gives you access to accounts and services provided by Citibank and its affiliates. Citibank, N.A. Member FDIC.

Copyright © 2008 Citigroup Inc.
bv/ib:im4-i:86-db:db2/bv

Figure 3. SSL Security Indicators and Security Seal

Digital certificate is a cryptographic method used in TLS/SSL to authenticate both parties of a communication. When a browser is connecting to a web server requesting a TLS/SSL session, the server sends the browser its digital certificate and the browser then verifies the certificate using the public key of the certificate issuer that has been pre-installed in the browser. If the certificate is accepted, the TLS/SSL session is established. A digital certificate contains information for site validation including:

- Common name: It is the server domain name displayed in the certificate. Once a certificate is received, browsers compare the common name with the domain name in the URL. If they do not match, it is likely that the certificate is forged and a warning window would pop up to warn users of the potential risk. We conducted the same comparison.

- Organization name: Each certificate displays the name of the organization that operates the server. The organization name usually is the same as the company name on the web site. However, there are a couple of reasons that the organization name could be different. First, the organization who registers for the certificate is the holding company. Second, companies may outsource its web services to a third party who operates the web server.

- Cipher suite: the cryptographic method used to secure the transaction, which include the key size, the hash function used, the authentication and digital signature cipher and a symmetric key encryption method. Since a browser usually advertises several cipher suites, the web server picks the strongest one it can offer. In our survey, we advertised all cipher suites that are currently available so that we could know the strongest cipher suite that the banking web server adopts.

- SSL version: the most current SSL version is TLSv1 although SSLv3 or SSLv2 are still used by some web servers. SSLv2 and SSLv3 have been proved to have several security vulnerabilities[6].
- Validity duration: the number of years that the digital certificate to be valid.

1.4 OpenSSL

OpenSSL[7] is an open source library that provides cryptographic functionality to applications, such as web servers. It is maintained by a group of volunteers worldwide. The library includes programs for public keys creation, public key and private encryption, digital certificates creation and signing, and SSL/TLS communications. OpenSSL library can be complied under both Windows and Linux. We will utilize OpenSSL in our laboratory exercises to create and install a digital certificate for a unsecure web server.

2 Labs Objectives

From this lab, you will learn about

- Setting Browser Security Policies
- Generating a digital certificate using OpenSSL
- Deploying a digital certificate on an Apache server
- Signing and deploying a digital certificate
- Observe SSL communications

3 Lab Setup

1. You will use the *ubuntu 10* virtual machine for SWEET teaching modules.
2. Extract the virtual machine from *ubuntu10tm.exe*.
3. Under the folder *ubuntu10tm*, double click on *ubuntu10tm.vmx* to start the virtual machine.
4. The username is "user" and the password is "123456".

4 Lab Guide

4.1 Web Browser Certificate Management

1. Open a Firefox browser. In the tool bar, go to Edit -> Preferences ->Advanced -> Encryption.

2. What is the encryption protocols used for web transactions? ____________________

[6] Wagner, D., and Schneier, B. "Analysis of the SSL 3.0 Protocol," *The Second USENIX Workshop on Electronic Commerce Proceedings*, USENIX Press, November 1996, pp. 29-40.

[7] OpenSSL toolkit is maintained by the OpenSSL Project and is available at http://www.openssl.org/.

3. Click View Certificates -> Authorities. You will see a list of Certificate Authorities that are on your computer when the browser was installed. These are the names of companies that offer certificate signing and verification (these are the trusted third parties by default).

List two of their names. 1)______________ 2)______________________

4. Along with the names of the company, you will see the certificates of these companies. These are called "root certificates". Choose one of these certificates and click the View button. Take a screenshot of the certificate and paste it below.

____

5. Now, click on the Details tab. In the Certificate Fields box, find the Certificate Signature Algorithm. This is the algorithm that the CA used to sign this certificate.

What is the algorithm used here? _____________________

6. Look at the Subject's Public Key field. This is the CA's public key.

How long is the Subject's Public Key? _________ bits.

7. Click Close to exit the Certificate Viewer and click OK to leave the Certificate Manager window and go back to the Preference window. Click the Validation button.

8. What is the network protocol listed here that can be used to verify the certificates? ___

What is the reason that a certificate might be invalid? ___________________

9. Close the Firefox Preferences Window and the Firefox browser.

4.1 Creating SSL Certificates Using OpenSSL

In the following exercises, you will learn how to install your very own secure web server which utilizes SSL/TLS for secure communications. To install an open-source secure web server, web developers usually need to install two software packages from different sources. For simplicity, they have already been installed on your Ubuntu virtual machine.

Apache 2.2.11: Apache2 is an open-source web server. You can find more information about Apache2 at *http://httpd.apache.org/.*

OpenSSL 0.9.8k: OpenSSL is a set of open-source toolkits that implement the Secure Socket Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a general purpose cryptography library. You can find more information at *http://www.openssl.org/.*

Mod_SSL 2.2.11: Mod_SSL is an add-on module for older versions of Apache that had to be compiled. With Apache2, mod_SSL is built into the server which provides the interface between Apache and OpenSSL. You can find more information about mod_ssl from *http://www.mod_ssl.org.*

We will need to create a SSL certificate for the web server before we can run the server securely with HTTPS. This exercise creates a public/private key pair, a SSL certificate, a certificate signing request (CSR) and we will also become a Certificate Authority (CA). Usually a commercial server would ask a trusted third party to sign their certificate. For example, VeriSign is one of the most well known companies that signs certificates for commercial servers.

You must have a public/private key pair before you can create a certificate request. You will also need a FQDN (Fully Qualified Domain Name) for the certificate you want to create. Since we are hosting the website locally, you are able to choose any FQDN that you like. For this lab exercise we will **www.BadStore.net** as a domain name. You will also be creating a certificate for BadStore.net.

1. Access the Terminal window by navigating to *Applications* > *Accessories* > *Terminal.*

2. Point the terminal shell to the ***/etc/apache2/ssl*** directory by running command:

   ```
   cd /etc/apache2/ssl
   ```

3. The ***ssl*** directory is where you will store all your private keys, certificate signing request and certificates.

4. There are existing keys and certificates under this directory (you can see them using ***ls*** command). We will start this exercise by a new set of keys. So, please delete the files under this directory before we start.

   ```
   sudo rm *
   ```

   When prompted for the [sudo] password, it is the same password (***123456***) that was used to login.

5. To generate the Certificate Signing Request (CSR), you will need to create your own private/public key first. You will create a key by the name of **server.key**. Run the following command from terminal to create the key:

   ```
   sudo openssl genrsa -des3 -out server.key 1024
   ```

   genrsa indicates to OpenSSL that you want to generate a key pair.

   des3 indicates that the private key should be encrypted and protected by a passphrase.

   out indicates the file name in which to store the results.

   1024 indicates the number of bits of the generated key.

   The result is going to look something like this:

   ```
   Generating RSA private key, 1024 bit long modulus
   ........++++++
   ...++++++
   e is 65537 (0x10001)
   Enter pass phrase for server.key:
   Verifying - Enter pass phrase for server.key:
   ```

   You will be prompted for a pass phrase, once you type in your initial pass phrase, you will be asked to verify this pass phrase. **Write down your passphrase below:**

   ____

If you execute command ***ls*** in terminal you will see a file called **server.key** in the *ssl* directory**.**

6. You will store your passphrase in a password script, so that Apache2 will not prompt you for the pass phrase whenever Apache2 is started or restarted.

   Run command: **sudo gedit /etc/apache2/ssl_passphrase**

   The **ssl-passphrase** script will be opened in a text editor. Enter your private key pass phrase in-between the pink quotes replacing ***123456***. Click *File > Save*.

   If the pass phrase does not match the pass phrase that you had submitted to Apache2 when you created your private key, the private key will not be decrypted when Apache2 starts resulting in non accessible hosted websites.

7. Next you will create a certificate signing request with the private/public key you have just created. This command will prompt for a series of things: When prompted enter the values as follows:

   Country Name: US
   State or Province Name: New York
   Locality Name: New York
   Organization Name: Pace University
   Organizational Unit Name: CSIS-IT300
   Common Name: www.BadStore.net
   Email: Your email address
   A challenge password: enter another password that you can remember.
   An Optional Company Name: CSIS

   A very important step to keep in mind is when filling out the **Common Name (CN)** field. The Common Name should match the web address, DNS name or the IP address you will specify in your Apache configuration. For this lab example the Common Name that we will be using is www.BadStore.net

   Otherwise to create the Certificate Signing Request, run the following command at the terminal prompt, making sure you are still in the ***/etc/apache2/ssl*** directory.

   sudo openssl req -new -key server.key -out server.csr

   You will be prompted to enter your private key passphrase. You will also be prompted to enter the values which were addressed above.

   You may run command **ls** in the *ssl* directory to see a file called ***server.csr*** when you are finished. This is your certificate signing request.

8. Now you will create your self-signed certificate. Make sure you are still in the ***/etc/apache2/ssl*** directory.

   The certificate signing request (CSR) has to be signed by a Certificate Authority (CA). For testing purpose, we will not ask a commercial CA (such as Verisign) to sign our certificate but sign the CSR by ourselves, which is called self-signing. In this case, we are our own CA.

   The following command will create a self signed certificate that will last for 365 days.

   sudo openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt

   The command will prompt you to enter your private key passphrase. Once you enter the correct passphrase, your certificate will be created and it will be stored in the ***`server.crt`*** file.

   The above command, took the certificate signing request, plus your private key in order to make your self-signed certificate.

9. Run command **ls** in the ***/etc/apache2/ssl*** directory and you will see ***server.crt, server.csr***, and ***server.key***.
10. You have just created your very own self-signed certificate.

4.2 Configuring Apache2 with BadStore.net

1. In the **ssl** folder under the Apache2 directory, there should exist three files, server.crt, server.csr and **server.key**. Make sure these files reside in this directory for the next steps to be successful.

2. Enable the SSL module for Apache2

   sudo a2enmod ssl

   ** To disable the SSL module for Apache2, the command is **sudo a2dismod ssl** **
3. A restart of Apache2 is required for the SSL module to be effective.

   sudo /etc/init.d/apache2 restart

   Apache should restart with no errors.
4. Creation of a virtual host for the secured BadStore.net website is necessary so that when the FQDM is entered into a web browser, the secured BadStore.net website will be available at that address. Start by copying the default template which will be modified in the following steps.

sudo cp /etc/apache2/sites-available/default /etc/apache2/sites-available/www.badstore.net

The template copy is now named www.badstore.net and is located in the /etc/apache2/sites-available/ directory.

5. Edit the content of the template copy. This is a very important step. Make sure it is modified correctly.

   sudo gedit /etc/apache2/sites-available/www.badstore.net

   Change the VirtualHost port from 80 to 443. The first line of the file will look like the following:

   <VirtualHost *:443>

   Next declare the server name and the fully qualified domain name plus the HTTPS port number after the **ServerAdmin** line.

   ServerName www.badstore.net:443

   Change the **DocumentRoot** to point to the badstore web directory (setup of the badstore directory will be explained later in this document).

   DocumentRoot /var/www/badstore

   Also change the **<Directory /var/www/>** to reflect the badstore web directory.

   <Directory /var/www/badstore>

   In the line before ErrorLog /var/log/apache2/error.log, you will enter these values

   SSLEngine On
   SSLCertificateFile /etc/apache2/ssl/server.crt
   SSLCertificateKeyFile /etc/apache2/ssl/server.key

   These entries tell the virtual host where the SSL certificate and key are located and to turn on SSL.

6. Save the file.

7. Enable the website.

   sudo a2ensite www.badstore.net

8. Edit the **/etc/hosts** file to resolve the www.badstore.net website to 127.0.0.1 since the website is hosted locally.

   sudo gedit /etc/hosts

   Find the line that begins with 127.0.0.1, remove the localhost text and in its place enter www.badstore.net.

9. Save the file and exit gedit.

4.3 Running a Secure Web Server

1. All the web server settings are contained in the Apache2 server configuration files. For simplicity, the configuration files have already been modified for you. If you have correctly followed the directions for generating SSL certificates, everything will work correctly.

2. You need to restart Apaceh2 for the SSL certificate settings to take effect. Run command:

   ```
   sudo /etc/init.d/apache2 restart
   ```

   If everything is correct, Apache2 will restart and give you an [OK] message.

3. Open Firefox and visit *http://localhost* you should be greeted with a message that states "It Works!"

4. In your browser, visit the following URL https://www.badstore.net

5. **VERY IMPORTANT**: The beginning of the URL is **HTTPS** not **HTTP**

   Firefox will display a message stating ***Secure Connection Failed*** due to the certificate that is used. In Firefox, The certificate is not trusted because it is self signed. We will accept the certificate anyway and add an exception.

   Click the ***I Understand the Risks*** link.

   Click the ***Add Exception*** button.

   Click the ***Get Certificate*** button.

   You can click the ***View*** button to see the self-signed certificate that you have created.

   Make sure the ***Permanently store this exception*** checkbox is selected and click the ***Confirm Security Exception*** button.

6. Take a screenshot of the secured web page and paste it below.

____

7. Do you see a silver lock on the lower right corner of the browser window? _______
8. Double-click on the silver lock.
9. Click on the ***View Certificate*** button. What is the validation period of this certificate? ____________________.
10. Click on the ***Details*** tab. Who is the issuer of this certificate? ____________________
11. What is the functionality of the certificate during web communication between the browser and your web server?

    ____

[**Note**: You will see the same silver lock when you check out your purchase on Amazon.com or other secure web servers. Double-click on the silver lock next time when you shop online and take a look at their certificates]

4.4 Client SSL Handshakes

1. The following will show you what information is transmitted between the secure web server and the browser during SSL handshaking. You will capture the results in a text file (handshake.txt).

   a. In the terminal window, login as the root user with command:

      sudo su

   b. Change your directory to point to the Desktop directory with command:

      cd /home/user/Desktop

   c. You will capture the SSL handshake that takes place between the client browser and Apache2 web server with the following command:

      openssl s_client -connect www.badstore.net:443 > handshake.txt

   d. You should see the web server return some certificate information and the phrase “Verify return:1”. This means that the server has acknowledged your request. At this point, please type

      GET / HTTP/1.1

   e. There will be a new file created on you Ubuntu desktop called *handshake.txt*. It will appear with an orange lock icon in the upper right corner. The orange lock represents the read-only file permission.

2. Double-click *handshake.txt* to open it in the gedit text editor. Copy & paste the results from the text file below:

____

3. Try to reason the content of the file “*handshake.txt*”.

   Why does the web server send you the information? __________________

   What information does this file contain? _____________________________

4. Suggestion: If you have tried the SWEET module on Security Testing, you can try using Paros to intercept the web transactions again. The web content should be able to view and to be tampered since they not now encrypted.

5. Close the gedit text editor, terminal, and shut down your Ubuntu virtual machine.