PACE UNIVERSITY

Hands-on Teaching Modules for Secure Web Application Development

ACM SIGCSE 2011 Workshop 27

Li-Chiou Chen and Lixin Tao

March 12, 2011

 The authors would like to acknowledge the support from 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.

Table of Contents

Introduction

This workshop will discuss security issues in web application development and demonstrate a set of teaching modules in this area through hands-on exercises, developed by a NSF-funded project called SWEET (Secure WEb dEvelopment Teaching). The workshop will guide the participants to run through a couple of web security hands-on exercises, including web server threat assessment, security testing, and secure web transactions. All exercises are pre-configured in Linux virtual machines. The workshop will also discuss examples of incorporating SWEET in computing curriculum.

SWEET features virtualized web servers and a development platform that allows instructors to teach the security issues in web application development using regular computer laboratories. It includes teaching modules that are composed of the lecture materials and hands-on exercises.

The workshop DVD includes the laboratory exercises for the workshop, the presentation slides and the following sub-directories:

- Modules: All SWEET teaching modules including labs
- Solutions: Sample solutions for lab questions
- Tools: VMware Player[1]
- VM: SWEET virtual machines
- Tutorial: Linux & HTML tutorials

The exercises in this document are excerpted from SWEET teaching modules, which are included in the workshop DVD. For updates of SWEET teaching materials, you should visit the project website at **http://csis.pace.edu/~lchen/sweet/.** Below is the agenda of this workshop.

- Introduction to the SWEET project (10 minutes)
- Virtualization technology (30 minutes)
  - Exercise 1-3: Starting Linux virtual machine
- Security issues in web application development (40 minutes)
  - Exercises 4-8: Web server threat assessment
- Web application security testing (40 minutes)
  - Exercises 9-10: Web server scanning
- Digital certificate, HTTPS & SSL (40 minutes)
  - Exercises 11-13: Secure web transactions
- Wrap up & discussions (20 minutes)
  - Exercise 14: Turn off Linux virtual machine
  - Course integration, support, and others

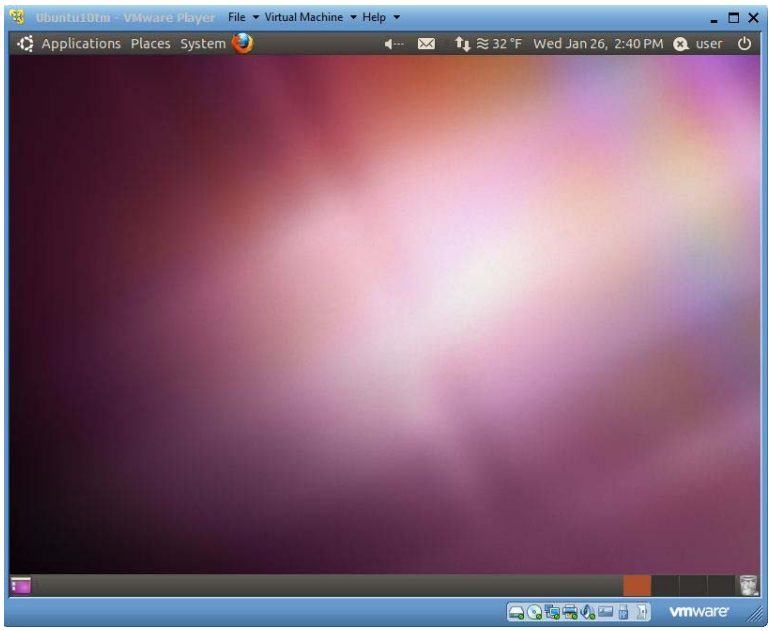
[1] VMware player (for Windows) is free for downloading and VMware Fusion (for MacOS X) is free for 30-day evaluation at www.vmware.com.

Exercise 1: Virtual Machine Installation

1. Copy all DVD materials to your computer where you will run the exercises.
2. On your computer, under the folder Tools, double click on VMware-player-xxxx.exe to install VMware player on your Windows machine or install VMware-Fusion-3.1.2-332101-light.dmg on your MacOS.
3. On your computer, under the folder VM, extract unbuntu10tm.zip to obtain the virtual machine.

Exercise 2: Boot up Linux Virtual Machine

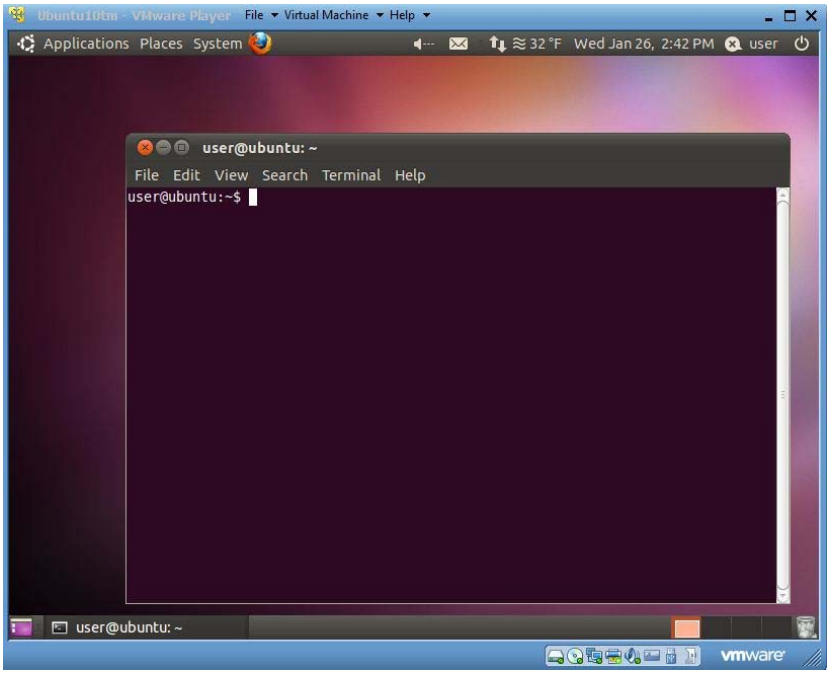
1. After VMware Player (or VMware Fusion) is installed, run the software and you should see a blue VMware Player Window pops up. Click on "Open a Virtual Machine" and select "Ubuntu10tm.vmx" from the "ubuntu10tm" folder under the ubuntu10tm folder.
2. Click on "play virtual machine". When being asked "Did you move this virtual machine, or did you copy it?" check "copy it".
3. When being asked "Would you like an attempt to be made to connect this virtual device every time you power on the virtual machine?", press "No" to avoid connecting to a virtual floppy.
4. When being asked if you would like to download VMware tools for Linux, answer "remind me later." Linux will boot up in about 2-3 minutes.
5. Login Linux using username "ubuntu10tm" and password "123456". After logging in, you will see Ubuntu 10 GNOME interface. The virtual machine runs Linux as if it is an independent computer. Actually, the Linux is run in the memory of the computer and simulate another physical machine that the virtual machine (VM) was created.
6. Once you logging in the system, you will see the Linux desktop, which looks like the screen below.



7. Below are some basic skills to use a virtual machine
   - To start directing mouse and keyboard input to a running virtual machine, type Ctr+g or click anywhere in the virtual machine window.
   - To start directing mouse and keyboard input to the host PC, type Ctr+Alt.
   - To get the logon window for Windows, use Ctr+Alt+Insert, instead of Ctr+Alt+Delete.
   - Scroll the bar on the right and at the bottom of the virtual machine window to see a wider screen.
   - To transfer files between the host and a running Windows virtual machine, just drag-and-drop the files.
   - USB disk is also a convenient way for transferring files between the host PC and a virtual machine. Inserting a USB disk to your PC when the virtual machine is active will attach the USB disk to the virtual machine.
8. Check out the menu bar for Linux GUI on the top panel of the window. The menu bar includes Applications (similar to Windows Start Panel), Places (all devices and storages), and System (Linux system functions).

Exercise 3: Basic Linux Commands

1. Click on Applications, Accessories and Terminal (You may need to scroll the window down to see Terminal if your screen is not big enough).
2. It opens up a Linux command prompt like the screen below.



3. Try Linux commands under the command prompt "user@ubuntu~$" (we will use $ referring the command prompt for all the instructions below). We will practice several basic Linux commands. For more Linux commands, please read the Linux Tutorial.
4. Try the following to see the files in this directory.

   $ls -al

Question 1: What are your results from "ls –al"? Copy and past the last three lines below.

Question 2: What does each line above mean? Please explain it. (**Hint**: In Linux, if you do not what a command mean, simply type "man command-name" to figure it out. For example, in this case, you can type "man ls")

Exercise 4: Observing HTTP Communications with *Paros*

In this exercise, you will use the *Paros* proxy server on your *Ubuntu virtual machine*, and use *Paros* to observer HTTP communications. The *Paros* proxy server will run at port 8088. You will set up the *Firefox* web browser so that when you use the browser to visit any web site, the HTTP request will be first forwarded to the *Paros* proxy server running at port 8088, which displays the HTTP request information in its graphic user interface, lets the user to have a chance to review and modify the request, sends the request to its destination server, and forwards the HTTP response from the web server back to the *Firefox* web browser. In this exercise you mainly use *Paros* to intercept HTTP GET/POST requests.

1. To start Paros, you need another Linux terminal window. Select Applications > Accessories > Terminal. Run these commands in the terminal window:

   cd ~/tools/paros

   sh startserver.sh &

   The Java-based Paros will execute and you will be greeted with its interface.

2. Open a Firefox browser. Now, you will need to change the proxy server settings in Firefox to redirect the web traffic to the proxy server. The proxy server is run under localhost and port 8088.

   - Go back to your browser. Select Edit > Preferences > Advanced > Network Tab > Settings.
   - Select the Manual Proxy Configuration radio button.
   - Enter these values into the fields: HTTP: 127.0.0.1 Port: 8088
   - If there are any values in the No Proxy For: text field, delete them. This is important to make the proxy work successfully.

3. Test the Paros proxy server by visiting http://localhost with your Firefox web browser. Once the browser contacts your web server, Paros starts to display HTTP request information in its graphic user interface.
4. You have just enabled all HTTP traffic generated by Firefox to be sent to the running Paros proxy server which can analyze HTTP traffic before it is sent off to its final destination.
5. Reload http://localhost. Go back to Paros, click on Sites to reveal “http://localhost”, and select it.

Question 3: What HTTP request was used in the step above? _____

Question 4: Copy and paste below the Paros window with the HTTP request information.

6. Click on the HTTP request and the Response tab. You will see the HTTP response sent from the local Apache web server that Paros reads while being transmitted.

Question 5: What is the server version? _____

Question6: Copy and paste below the Paros window with the HTTP response information.

Exercise 5: Starting WebGoat

1. The web server has already been pre-installed and all you need to do is run the program.
2. From the menu bar on the top of the VM, select Applications > Accessories > Terminal. This will open up a Linux shell command terminal.
3. Run the following commands to start the web server.

   ```
   cd ~/tools/tomcat/bin
   sh startup.sh
   ```

4. Go to the Apache server homepage on your VM by entering http://localhost:8080/ into the Firefox address bar.

   (Tip: When running a program on port 80, you do not have to specify the port number as it is the default HTTP port used by most web applications and servers.)

Question7: Take a screenshot of the Apache Tomcat welcome page and paste it below.

Exercise 6: Web Goat Login

1. Go back to the browser in Ubuntu. Browse to http://localhost:8080/WebGoat/attack.
2. When prompted for login information, the user name is guest and the password is guest. On the WebGoat home page, click Start WebGoat.
3. Go back to Paros. You should see Paros logs of the connection between the browser and WebGoat.

Question 8: What is the first HTTP request in these transactions? _____

4. Go back to your browser. The left side of the screen is a list of WebGoat exercises you can try. We will try the first lesson, click on "General" and underneath it choose "HTTP Basics".
5. This WebGoat exercise will accept a value in the text box that you enter and reverse it. In the "Enter your name" textbox, type your name and click on Go.
6. You can see the parameters sent between your browser and WebGoat with Paros.
   - In Paros, look under Sites > http://localhost > WebGoat > POST:Attack(Screen,menu).
   - Change the view from Raw View to Tabular View.
7. You can see the session ID by looking under the Request tab in Paros next to the Cookie: reference.

Question9: There are two parameters in the transactions using POST: person and SUBMIT. The value for person is _____ and the value for submit is _____.

Question 10: Explain the function of the POST command with the two parameters.

8. When you see a green check on the exercise name in the left pane, you have successfully finished the exercise. You can also see the solution for the exercise by clicking on Show Solution in WebGoat. For all the exercises below, you can either read instructions from this manual or read the instructions from Show Solution (We use Paros instead of WebScarab in our lab).

Exercise 7: Injection Flaws – String SQL Injection

SQL injection is used to gain access, extract data or compromise secured databases. The methods behind an attack are simple and the damage dealt can range from inconvenience to system compromise.

1. On WebGoat, click on Injection Flaws and underneath select String SQL Injection in the left pane in WebGoat.
2. The exercise first prompts you to enter the last name Smith (SQL is case sensitive). A simple SQL query to a database would looks like this

   SELECT * FROM user_data WHERE last_name = 'Smith'

3. In this case, Smith is the value you enter in the "Enter your name" textbox. The SQL will select all information about the user 'Smith' from the database.

Question 11: How many entries have you obtained? _____

Are they all information regarding the user "Smith"? _____

4. Let us look at the following SQL command

   SELECT * FROM user_data WHERE name = '**Smith' OR '1'='1**'

5. This SQL command will ask the database to show all user information since '1'='1' is always true. String SQL injection exploit the vulnerability that a database that does not conduct checks on constraints so that attackers can inject SQL commands through the regular user interface.
6. Now, enter **Smith' OR '1'='1** in the "Enter your last name" textbox.

Question 12: Paste a screen shot of your results below.

Question 13: What is the security implication of your results?

Question 14: Describe a method to fix the vulnerability in this exercise.

Exercise 8: Cross Site Scripting (XSS) – Stored XSS attack

Stored XSS attacks allow users to create message content that could cause another user to load an undesirable page or undesirable content when the message is viewed or accessed. In this exercise you will create such a message.

1. On WebGoat, click on Cross-Site Scripting (XSS) and underneath select Stored XSS Attacks in the left pane in WebGoat.
2. In the title text box, type “XSS example”
3. In the message text box, type in the following HTML content.

   <script language="javascript" type="text/javascript">alert("you are hacked");</script>

4. Click on Submit and click on the message you have just posted under message list.

Question 14: What will happen if someone clicks on the message you have just posted?

Question 15: Paste a screenshot of the results below.

Question 16: What is the security implication of your results?

5. Logout WebGoat and close the browser.

Exercise 9: Crawling Web Pages and Hidden Web Directories

We will investigate the web traffic between your browser and a pre-configured vulnerable website, the BadStore.net, using a web proxy called Paros on a same virtual machine. All web communication between the browser and the Web server will be sent to Paros (the proxy server) first before it reaches the appropriate destination. We will be browsing BadStore.net or investigate its vulnerabilities.

In order for an attacker to successfully plan and execute an attack, the attacker must know the website's layout and all the pages that might be available for exploitation. While manual web crawling is an option, it is a very time consuming process. An automated web crawler application will speed up the mapping process significantly.

1. From the menu bar on the top of the VM, select Applications > Accessories >
2. Terminal. This will open up a Linux shell command terminal, execute the command

   ifconfig
3. You will receive several lines of output. You are going to look for the Ethernet interface **(i.g. eth0)**. Find the **inet addr**: field and write down the IP address in the space below.
4. Make sure Tomcat server is not running, otherwise go to Select Applications > Accessories > Terminal. Run following command in the terminal window to shut down tomcat

   tomcat-stop
5. Open a Firefox browser to browse BadStore.net by typing the IP address of your VM in the URL e.g **http:// IP ADDRESS/badstore** (DO NOT browse www.badstore.net directly since it will redirect you to original website if you have an Internet connection).
6. Switch to the Paros application and click **File > New Session** and click **OK** to have Paros start a new session and purge itself from any logged content.
7. In the Paros menu toolbar, navigate to **Tools > Options** and select the **Spider** option. You will change the **Maximum Depth to Crawl** from its default value to the

maximum value of **9**. This will allow Paros to crawl web pages that may be deeply nested in BadStore.net Click **OK** to confirm and return to the main screen.

8. In the **Sites** panel on the left will be all the websites that Paros is logging. It is currently blank, change to the Firefox application and refresh BadStore web page (You may have to clear recent browsing history first to reload the page. **Tools > Clear Recent History**)
9. Switch back to Paros and you will see an arrow next to **Sites** that is point to the right. Click the arrow to un-collapse the logged websites. You will see the IP address of BadStore website.
10. Select the IP address for BadStore under **Sites** in Paros. The IP address will be highlighted in brown and go to **Analyze > Spider** in the Paros toolbar menu.
11. A Spider window will open. In the **URL crawling:** field should be the IP address of BadStore.net. If all checks out, click **Start**.
12. Once crawling has begun, the main Paros window will begin to populate with web pages and images that are hosted within BadStore.net.
13. In the bottom pane of the main screen, you will see a **URL found during crawl:** panel. Notice that it is located under the **Spider** tab near the bottom.
14. Looking through the entries, you will notice that most of the web cgi pages are located in the **/cgi-bin/** directory but some are not. List one other directory that Paros had crawled and one file under this directory

    Directory name:_____

    File name: _____

Question 17: Briefly explain what information one might obtain by crawling a web site.

Question 18: What is the potential risk for a web site being crawled?

Exercise 10: Scanning For Known Vulnerabilities

In the previous exercise, you have mapped BadStore.net; in this exercise you will execute a vulnerability scan on BadStore.net.

1. In the Paros Sites panel, Click on the IP address of BadStore, highlighted in brown.
2. In the Paros toolbar menu, navigate to **Analyze > Scan**. The vulnerability scan will begin. Give it a minute to complete the scan.
3. Once the scan is finished, the results can be viewed on the bottom panel of the Paros application under the **Alerts** tab. If you would like to have an actual report, click on **Report** > **Last Scan Report**. The report will be generated in the **/user/paros/session** folder and it is called **LatestScannedReport.html.** Open it in a browser.
   (Click on the menu bar on the top, click on **Places** to access **Home Folder**)
4. The vulnerabilities are called alerts and are classified as High, Low, and Medium. List two vulnerabilities from the report and discuss the countermeasures to fix them.

Question 19: Vulnerability 1: _____

Countermeasure 1: _____

Question 20: Vulnerability 2: _____

Countermeasure 2: _____

5. Not all web crawlers and web vulnerability scanners are as robust. Commercial web crawlers and vulnerability scanners may perform a much more complete crawl and may list more potential vulnerabilities.
6. Click on **File** > **Exit** to close Paros.
7. Close all the command prompt terminals on your VM.
8. Open a Firefox browser. Now, you will need to change the proxy server settings back. Go back to your browser. Select Edit > Preferences > Advanced > Network Tab > Settings. Select No Proxy radio button and hit OK.

Exercise 11: 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 use **www.BadStore.net** as a domain name. You will also be creating a certificate for www.BadStore.net which is actually hosted on the local virtual machine

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.

Exercise 12: 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, replace localhost text at the end and in its place enter www.badstore.net.

9. Save the file and exit gedit. You have just configured SSL on www.badstore.net, a local Apache web server.

Exercise 13: 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 ***This Connection is Untrusted*** 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? _____

Question 21: Double-click on the silver lock. Click on the ***View Certificate*** button. What is the validation period of this certificate? _____.

Question 22: Click on the ***Details*** tab. Who is the issuer of this certificate?

Question 23: 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]

Exercise 14: Turn off Linux VM

1. Close all the browsers and terminal windows on the VM.
2. Turn off the Linux VM by clicking on the power button on the upper-right corner.
3. This VM is used by all exercises in the SWEET teaching modules. Please keep it on your computer for all the exercises.