8.2 Solutions and Guided Walkthrough

8.2.1 Solutions

1. *Stop Serving Page*

   In order to stop serving webpages on a web server, run the following command from a terminal:

   `sudo systemctl stop nginx`

   To start serving the page again, run the following command from a terminal:

   `sudo systemctl start nginx`

2. *Identify Key File Locations*

   For this task, the two key file locations are:

   - Web Page Content - `/var/www/html`
   - Web Server Configuration - `/etc/nginx`

3. *Enable HTTPS*

   In order to enable HTTPS on your web server, there are a few steps which need to be followed:

   - Use OpenSSL to generate a certificate and key

     To perform this task, run the following command:

     ```
     openssl req -x509 -newkey rsa:4096 -keyout public.key -out
     certificate.cert -days 365
     ```

     You will be prompted for a passphrase, chose a passphrase which is easy to remember and store it securely.

   - Configure Nginx to listen on port 443

     To complete this task, navigate to the directory `/etc/nginx/conf.d` and edit the file **ssl.conf**. Find the `server` block in the configuration file (**/etc/nginx/conf.d**) and add content to match Listing **8.2**:

Listing 8.3: *Enabling SSL*

```
server{
    listen 443 ssl;
    server_name <name>;
    ssl on;
    ssl_certificate <path to certificate>;
    ssl_certificate_key <path to key>;
}
```

Listing 8.4: *Disabling Unnecessary HTTP Methods*

```
if($request_method !~ ^(GET|HEAD|POST)$$)
{
    add_header Allow "GET, HEAD, POST" always;
        return 405;
}
```

Listing 8.2: *Enabling HTTPS*

```
server{
    listen 443 ssl;
    ...
}
```

- Configure Nginx to use the certificate and public key generated

  To accomplish this task, edit the file at `/etc/nginx/conf.d/ssl.conf` again and ensure that it matches the content in listing **8.3**:

4. *Disable Unnecessary HTTP Methods*

   In order to disable the HTTP methods excluding GET, HEAD, or POST, insert the segment seen in listing **8.4** into the file located at `/etc/nginx/conf.d/default.conf`:

5. *Prevent Giving Away Server Content Details*

   In order to reroute the 401, 403, 404, and 405 error codes to the default page for 404, insert the following line into the file located at `/etc/nginx/sites-enabled/default`:

   `error_page 401 403 404 405 /404.html`

6. *Creating a Web Application*

   See the walkthrough for detailed steps on this task.

8.2.2 Guided Walkthrough

In order to complete the challenges in this laboratory exercise, see the steps in this walkthrough. This guided walkthrough will outlines the steps when using VMware as the hypervisor, though VirtualBox can also be used and has similar functionality.

Creating the Virtual Machine

In order to begin this exercise, you will need to have access to a virtual machine running Ubuntu 16.04 Desktop as the operating system. If you do not already have access to a virtual machine running Ubuntu 16.04 Desktop, creating your own is a relatively simple. The directions from here forward are for a machine running Windows 10, though similar steps can be followed for other OSs.

Installing VMware

If an organization has the resources to purchase a license for VMware [41] Workstation Pro, they should do so. VMware workstation has extended capabilities including taking snapshots, creating and managing encrypted VMs, customizing virtual networks, and virtual machine cloning [24]. Although these extended capabilities can be convenient, everything necessary for this exercise can be performed with the free VMware Player. In order to download VMware Player, visit the VMware website and navigate to the Downloads tab. From the Downloads tab, navigate to VMware Workstation Player. From here, select the download button for Windows. The following link is valid at the time of creation (Setptember 2019): `https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html` [32].

After downloading the installer, follow the steps that the installation platform provides. A webpage created by Shailesh Jha [33] contains a walkthrough of the installation process: **https://www.shaileshjha.com/step-by-step-install-vmware-workstation-player-12-in-windows-10/**.

Downloading Ubuntu 16.04 ISO

Although this exercise could be performed on many different OSs, the one chosen is Ubuntu in the interest of the development teams familiarity with Ubuntu for web development. In order to create a virtual machine running Ubuntu, the ISO file will need to be downloaded. In order to download the ISO, visit the Ubuntu website. On the website, navigate to the Download tab. In the Download tab, click on the Older Releases under the *Other Ways to Download* section. Next, click the link with Name: 16.04. Depending on whether your host machine is 64-bit or 32-bit, select the appropriate download link for the Desktop image. The download should begin automatically. At the time that this document is being created, the following link will take you to the download page: `http://releases.ubuntu.com/16.04/` [34].

Creating a Virtual Machine

In order to create a new virtual machine, run the VMware Workstation Player application you installed earlier. One of the options should be to *Create a New Virtual Machine*. Select that option and then the *New Virtual Machine Wizard* will appear. Radio boxes will appear asking where to install the operating system from, select the option which says *Installer disc image file (iso)* and then browse to wherever you downloaded the Ubuntu 16.04 ISO file. Select the next button.

Create a username and password for the machine. It does not matter what you choose as long they are credentials which you will have access to later on to log in. Select the next button.

Give the virtual machine a name. This is the name that will appear in your VMware Workstation Player GUI. Also specify where you would like the virtual machine stored. Select the next button.

Specify the Disk Capacity. This section can be left with the default values. Select the next button.

If you are satisfied with everything up to this point, select the finish button. You can otherwise customize the hardware before finishing, although it is not necessary. You should now have a new virtual machine.

Installing Ubuntu on the Virtual Machine

At this point, either the virtual machine powered on automatically and you are viewing the console to it or you will have to power the machine on in the VMware Workstation Player GUI. Once the virtual machine is running, Ubuntu should begin installing on its own using the EasyInstall. If not, follow the steps in the installation process. A webapge created by Krishna [35] provides an in-depth guide on installing Ubuntu in VMware with images: `https://www.maketecheasier.com/install-ubuntu-in-vmware-player-windows/`.

Setting Up the Web Server

Setting up the Ubuntu virtual machine to be a web server requires a few steps. Follow the instructions to have a web page capable of serving PHP content and interacting with a MySQL database.

Installing Nginx

In order to install Nginx, open a terminal. This can be accomplished by selecting the Search Computer for Applications icon on the task bar. Search for **Terminal** and then select the Terminal icon to fire it up. Alternatively, pressing Ctrl + Alt + T will open a terminal. Type the following command to install Nginx:

```
$ sudo apt-get install nginx
```

To verify that Nginx was properly installed, open a browser (Firefox comes preinstalled on Ubuntu 16.04). In the URL bar, type **localhost**. If the installation was performed properly, you should be greeted by a *Welcome to nginx* page.

Installing PHP

At this point, your web server can serve HTML pages, but is not capable of using PHP which is integral to allowing server side capabilities to a web server. Open a terminal (see above for details on how to do this). At the terminal, type the following command:

```
$ sudo apt-get install php-fpm php-mysql
```

This will install PHP on your machine. Although PHP is installed, the step is not com-

plete; you must now tell Nginx how to use PHP. This is done by editing the file located at /texttt/etc/nginx/sites-available/default. Type the following command in the terminal in order to edit the file:

```
$ sudo nano /etc/nginx/sites-available/default
```

There are a few locations in the file which must be changed. In the server block, there is a line specifying file names for the index file (the landing page). You should see the following line:

```
index index.html index.htm index.nginx-debian.html;
```

Add the text index.php to the beginning of the list of index file names so that the line now appears:

```
index index.php index.html index.htm index.nginx-debian.html;
```

The next location which must be edited is in the sub block which is labeled `location ~ \.php$`. By default, this sub block is commented out. Remove the comments necessary so that the following lines are no longer commented.

```
location ~ \.php$
include snippets/fastcgi-php.conf;
fastcgi-pass unix:/var/run/php/php7.0-fpm.sock;
```

At this point, Nginx should be configured properly to serve PHP content. In order to test this, navigate to the `/var/www/html` directory. Here, create a new file called **index.php**. Do this by running the following command in the terminal:

```
$ sudo nano /var/www/html/index.php
```

Insert the following text into the file:

```
<?php
   phpinfo();
?>
```

Save the file and exit. Visit the web browser again. Once again, type **localhost** into the URL bar. If everything was done correctly, you should be served a page with all the information about your PHP installation such as the version. If this is not the case, please retry the steps ensuring that you have correctly performed the installation and file configurations [31].

Installing MySQL

Your web server should now be able to serve PHP content. The next step is to install MySQL so that the PHP pages can interact with MySQL databases. In order to install MySQL, run the following command in the terminal:

```
$ sudo apt-get install mysql-server
```

Done properly, the installation should begin and a prompt will appear asking you to set a password for the MySQL root user. As described in the prompt, this step is not mandatory, but strongly recommended. Set a strong password because the root user has full privileges to all tables in all databases. After entering the password, the installation process will continue. To test that MySQL was installed properly, enter the following command in the terminal:

```
$ mysql -u root -p
```

You will be prompted for the MySQL root user password which was set previously. After entering the password, you will be presented with a MySQL console. From here you can perform any MySQL commands including but not limited to [31]:

- Creating/Deleting users
- Modifying user privileges
- Creating/Deleting databases
- Creating/Deleting tables
- Entering data into tables

Creating the Vulnerable Application

Provided that you have not run into any errors up to this point, you are ready to create the vulnerable application. Prior to creating the files for the application, you will need to make some modifications in MySQL. Enter into the MySQL console by entering the following command in the terminal:

```
$ sudo mysql -u root -p
```

For the purposes of this tutorial, a weak password, namely: **sql**, has been selected for readability purposes.

After entering the password, you will be presented with a MySQL console. Once here, create a new database. You can name this database whatever you would like, but to follow along with the code presented later, name it **test**. You can do this by entering the following command in the MySQL console:

```
mysql> CREATE DATABASE test;
```

Next you will need to create a couple of tables. The table names do not have to be specific, but following the commands as written is most compatible with the code presented later. If you choose to use your own names, ensure that they are changed in the code later accordingly. To create the tables enter the following commands in the MySQL console:

```
mysql> USE test;
```

This command will change the active database to your newly created database.

```
mysql> CREATE TABLE employees (name VARCHAR(255), password VARCHAR(255));
```

This command creates a new table in the **test** database called **employees**. The fields, or columns, in this table are **name** and **password**, each of type VARCHAR with up to 255 characters (effectively a 255 character string).

```
mysql> CREATE TABLE startdates (name VARCHAR(255), date VARCHAR(255));
```

This command creates a new table in the **test** database called **startdates**. The fields, or columns, in this table are **name** and **date**, each of type VARCHAR with up to 255 characters (effectively a 255 character string).

Next, you will need to enter data into the **employees** database. Follow the command below to enter the data:

```
mysql> INSERT INTO employees (name,password) VALUES ("<username>","<password>");
```

Replace the username and password values with usernames and passwords you would like. Enter the data for six users in this manner. You may use whatever names and passwords you like, but to follow along with what was entered during development, use the following commands:

```
mysql> INSERT INTO employees (name,password) VALUES ("liam","password");
mysql> INSERT INTO employees (name,password) VALUES ("emma","superman");
```

Listing 8.5: *vw-index.html*

```
<html>
	<body style="background-color: black; color: white">
		<div style="text-align: center">
			<form action="vw-auth.php" method="get">
				Username: <input type="text" name="user"><br>
				Password: <input type="text" name="pass"><br>
				<input type="submit">
			</form>
		</div>
	</body>
</html>
```

```
mysql> INSERT INTO employees (name,password) VALUES ("william","mustang");
mysql> INSERT INTO employees (name,password) VALUES ("sophia","trustno1");
mysql> INSERT INTO employees (name,password) VALUES ("mason","hockeymason");
mysql> INSERT INTO employees (name,password) VALUES ("mia","curiousgeorge");
```

Now that MySQL is configured as needed, you are ready to move onto writing the code necessary to create the vulnerable web application. For this section, you can either follow along with the explanation of the code. Create the following files in the `/var/www/html` directory.

index.html

This file, vw-index.html, listing **8.5**, contains a simple form into which a user will enter a username and password. Upon submitting their input, the contents of the text boxes will be passed to a page named **vw-auth.php** using the HTTP GET method.

vw-auth.php

In this file, vw-auth.php, listing **8.6**, variables with the servername, username, password, and database are used to establish a connection with the MySQL database. Next the values entered into the username and password fields on the **vw-index.html** page are used in a SQL query, the variable for this query is named **$sql**. The SQL query is then performed and the result is stored in the **$result** variable. Next, a check is performed to see if any matches were return (in theory a match is only returned if the correct username and password are entered). If matches were not returned, the user is served the **vw-invalid.html** page. If matches were

Listing 8.6: *vw-auth.php*

```php
<?php

$servername = "localhost";
$username = "root";
$password = "sql";
$dbname = "test";

$conn = new mysqli($servername,$username,$password,$dbname);

$sql = "SELECT * FROM employees WHERE name='" . $_GET["user"] . "' AND password='
    " . $_GET["pass"] . "';";

$result = $conn->query($sql);

$rows = array();

if($result->num_rows > 0)
{
        while($row = $result->fetch_assoc())
        {
                array_push($rows,$row);
        }
        print "Click your name to access your date entry form!<br>";
        foreach($rows as $r)
        {
                $n = $r['name'];
                print "<a href='vw-insertdate.php?name=$n'>" . $n . "</a><br><br>";
        }
}
else
{
        header("Location: vw-invalid.html");
}

?>
```

Listing 8.7: *vw-insertdate.php*

```
<?php
        $n = $_GET['name'];
?>

<html>
        <body style="background-color: black; color: white">
                <div style="text-align: center">
                        <form action="vw-insert.php" method="get">
                                Enter a date in MM/DD/YYYY format<br>
                                <input type="text" name="date"><br>
                                <input type="text" name="user" value="<?php echo $n;
                                    ?>" readonly><br>
                                <input type="submit">
                        </form>
                </div>
        </body>
</html>
```

returned, then the code loops over all the matches and creates a link to the **vw-insertdate.php** page for the associated user.

vw-insertdate.php

In this file, vw-insertdate.php, listing **8.7**, the name of the user clicked on the **vw-auth.php** page is passed via the HTTP GET method. Then an HTML page is created which allows the user to enter a date into a text field. The requested format is MM/DD/YYYY. Another text field is auto populated with the username of the associated user; this field is read-only. Upon submission, the values in both text fields are passed to the **vw-insert.php** page via the HTTP GET method.

vw-insert.php

In this file, vw-insert.php, listing **8.8**, variables with the servername, username, password, and database are used to establish a connection with the MySQL database. Next the values of the username and date entered into the text fields on the **vw-insertdate.php** are stored in variables. There is a check to ensure that the connection to the database was successful. Next, a SQL query is constructed to insert into the **startdates** table. The query is constructed so the username and date passed to the page via the HTTP GET method will be inserted into a row of the table. The query is then performed. Finally, the user is presented with an HTTP page

Listing 8.8: *vw-insert.php*

```
<?php

$servername = "localhost";
$username = "root";
$password = "sql";
$dbname = "test";

$user = $_GET['user'];
$date = $_GET['date'];

$conn = new mysqli($servername,$username,$password,$dbname);

if($conn->connect_error)
{
	die("Connection Failed: " . $conn->connect_error);
}

$sql = "INSERT INTO startdates (name, date) VALUES (\"$user\",\"$date\");";

$conn->query($sql);

?>

<html>
	<body style="background-color: black; color: white">
		<div style="text-align: center">
			<h1>Insert Complete!</h1>
			<button onclick="location.href='vw-index.html'">Return to
			    Login Page</button>
		</div>
	</body>
</html>
```

Listing 8.9: *vw-invalid.html*

```
<html>
	<body style="background-color: black; color: white">
		<div style="text-align: center">
			<h1>Invalid Logon Attempt!</h1>
			<button onclick="location.href='vw-index.html'">Return to
			    Login Page</button>
		</div>
	</body>
</html>
```

which states that the insertion has been completed and provides a link back to the logon page.

vw-invalid.html

In this file, vw-invalid.html, listing **8.9**, a simple HTML page is presented to the user. The page states that the logon attempt was invalid and provides a link back to the logon page.