Exercises – Basics of Web Security

Experiential Learning Workshop

1 General Guidelines

1. Make a team of two unless stated otherwise.
2. For each exercise, use wireshark capture to verify contents
3. Ensure to use proper capture filter and don't capture irrelevant traffic
4. Where appropriate or applicable, use **wget** or **nc** to access the web server.
5. The default client for accessing web server is assumed to be browser, preferably **firefox**. You can use Chrome or any other browse as well.
6. The webserver in the example below is taken as 'myweb.com'. Please use your hostname or corresponding IP address instead in your exercise.
7. To kill any program in the **Linux** terminal, please press **Ctrl-C** and not **Ctrl-Z**. The latter will suspend the program and not stop it.

2 Hands-on 1: Installing SSL Certificate

Note: It is assumed that Apache web server has been installed on one of the machine and let us call it web server. In case, it is not, please install it by invoking the command

```
sudo apt install apache2.
```

2.1 Invalid SSL certificate.

In the exercises below, we will work with SSL Certificate that are not compatible with browser expectation.

2.1.1 Accessing websites with invalid certificate.

1. Find the IP address of www.google.com, (e.g. `ping -c2 www.google.com`). Let this IP address be `a.b.c.d`
2. Open Firefox (or chrome) browser and access `https://a.b.c.d`.
3. The browser should display a web page with warning rather than displaying the google search page.
4. Analyze the warning. Click thru this warning and browser should display google search page.
5. Make an entry for this IP address a.b.c.d in your **/etc/hosts** file like below. Use editor of your choice to make an entry (e.g. nano, edit, gedit, vi etc.)
   a. a.b.c.d mygoogle.com
6. In the browser type the URL, https://mygoogle.com
7. Analyze the web page response.

2.1.2 Running your own web site with HTTPS

1. Generate and deploy a web site SSL certificate e.g. for (`mywww.com`) on Apache web server. The steps are given in Appendix.
2. Create an entry in the **`/etc/hosts`** file of the client machine (creating this entry on the web server doesn't really matter). The entry should be something like as below where 10.1.1.101 is server's IP Address
   a. `10.1.1.101 mywww.com`

3. Verify that Apache web server is running. In browser of client machine, first enter `http://mywww.com`. It should display Apache welcome page.
4. Now, in the browser, enter the URL `https://mywww.com`.
5. Analyze the web page response. It should be warning again, but this warning should have different indicator compared to when accessed Google search page with `https://mygoogle.com`.
6. Analyze the warning web page thrown by the browser. The warning should correspond to unknown certificate authority rather than website name (as was the case earlier when accessing Google with its IP address or any other locally defined name.
7. Import the certificate of into browser repository as Exception. Click thru and then browser should display the Apache welcome page. Import
8. Close the browser, reopen it again and access again the web page https://mywww.com. This time no warning should be displayed since browser knows the certificate authority.
9. Access the website, https://mtechthesis.vtu.ac.in. Analyze the error thrown by the browser.

2.1.3 Exercise Expectations

1. Understand SSL certificate errors in case of any mismatch of information (unrecognized certificate authority, website name mismatch, improper certificate validity period.

3 Mixed Content.

In the exercises below, we will work with web pages that has embedded objects with both HTTP and HTTPS.

3.1 Pure content.

1. Create a web pages, e.g. pure.html as below in your directory /var/www/html.

```
<html>
<head>
<title>Pure Content </title>
<body>
<h1>Pure Content Demonstration</h2>
<h2>Image 01 with inherited security access.</h2>
<h2>Image 02 with inherited security access.</h2>
<img src="//mywww.com/img/img-01.jpg" alt="Img 01">
<img src="//mywww.com/img/img-02.jpg" alt="Img 02">
</body>
</html>
```

2. Create the directory `/var/www/html/img` and copy some images with the name `img-01.jpg` and `img-02.jpg` in the images directory.
3. The web page has two embedded objects for which web protocol is not defined explicitly. The browser will inherit the protocol (HTTP or HTTPS) used to access the parent page i.e,. pure.html
4. Access this web page using HTTP and HTTPS in the browser i.e. http://mywww.com/pure.html and https://mywww.com/pure.html.

Alteratively, you can access these web pages at rprustagi.com/accs/pure.html with HTTP and HTTPS.

5. Browser should show the content properly without any warning. With HTTPS access, it should show Green Lock icon.

3.2 Mixed content.

1. Create a web page like below (mixed.html) in the directory /var/www/html.

```
<html>
<head>
<title>Mixed Content </title>
<body>
<h1>Mixed Content Demonstration</h2>
<h2>Image 01 with inherited security access.</h2>
<h2>Image 02 with insecure access.</h2>
<img src="//mywww.com/img/img-01.jpg" alt="Img 01">
<img src="http://mywww.com/img/img-02.jpg" alt="Img
02">
</body>
</html>
```

2. This web page specified the second embedded object explicitly with HTTP protocol. Thus, even if the parent page (mixed.html) is accessed with HTTPS, the second object will be accessed using HTTP. This web page represents a lot web content on the internet as developers have hard coded the HTTP protocol in the web content by means storing these in Database, java beans, images etc. Possibly, because some of the websites serving these contents support only HTTP and not HTTPS.
3. In Firefox browser, access this web page with HTTP and HTTPS protocol i.e. http://mywww.com/mixed.html, and https://mywww.com/mixed.html. Alternatively, access these web pages at rprustagi.com/accs/mixed.html.
4. The first access will show content properly. The 2nd access with HTTP will show the gray lock icon with a warning sign.

3.3 Mixed content.

1. Create a web page like below (mixed-active.html) in the directory /var/www/html.

```
<html>
<head>
<title>Mixed Active Content </title>
</head>
<body>
<script src="http://rprustagi.com/js/mywww.js">
</script>
<h1>Mixed Content Demonstration</h2>
  <button type="button" onclick="hello()"> insecure
access
  </button>
<h2>Image 02 with insecure security access.</h2>
<img src="//rprustagi.com/img/img-02.jpg" alt="Img
01">
```

```
</body>
</html>
```

2. Create a javascript file mywww.js with following content in the directory like below (mixed-active.html) in the directory `/var/www/html/js` on the web server.
```
<script>
??
</script>
```

3. In the Firefox browser, access the web page https://mywww.com/mixed-active.html (alternately https://rprustagi.com/accs/mixed-active.html).
4. Browser should show gray lock icon with a cross indicating danger. Click on the button "`Insecure Access`". It should pop up "Hello" text.
5. The java script has been obtained explicitly using HTTP and this causes a serious vulnerability to an intruder/attacker.
6. Click on the gray lock/information icon and enable the security protection in the browser. Refresh the web page. This time, browser would show Green lock icon indicating the safe access to the web page.
7. Click on the button "`Insecure Access`", and no pop-up occurs. This is because browser has disabled the insecure access of javascript.
8. It is recommended to always use enable security protection in the browser to ensure that your web access is not compromised.

3.4 Exercise Expectations

1. Understand lock icons display when web page has insecure references i.e. mixed contents.
2. Understand what kind of contents are secure and insecure and how to disable insecure access.
3. Differentiate between passive mixed content with active mixed content, the latter being lot more dangerous in nature.

4 Hands-on 2: ARP Spoofing and MITM

This exercise requires setup of 3 machines.

4.1 Silent snooping

1. Identify the attacker machine, say X. Identify the IP address of A and B.
2. Install **dsniff** on attacker machine (`sudo apt install dsniff`)
3. On A, ping IP_B and on B, ping IP_A and note down their MAC address in ARP Table (`arp -an`). Verify that ARP table has correct MAC addresses.
4. Run the ARP Spoof attack, i.e. issue following commands on attacker machine X. The first command will send gratuitous APR message and send will enable routing on X.
   a. `sudo arpspoof -I enp1s0 -t` IP_A `-r` IP_B
   b. `sudo sysctl -w net.ipv4.ip_forward=1`
5. Look at the ARP table of A and B. On A, for IP_B, it should show MAC address of X. Similarly, on B, for IP_A, it should show MAC address of B.

6. Run wireshark capture on X with capture filter as host IP_A and host IP_B.
7. Do the chat using nc between A and B. Identify chat contents of A and B on X in wireshark

4.2 Dealing with ARP spoofing

1. In the above setup, issue make the static ARP entry for B on A and vice versa e.g.
   a. (On A) `sudo arp -s` IP_B MAC_B
   b. (On B) `sudo afp -s` IP_A MAC_A
2. Explore if silent snooping by attacker can still work.

4.3 Exercise Expectations

1. Able to understand working of ARP spoofing and simplicity with which it can be done.
2. Able to understand working ARP and how to avoid silent ARP spoofing and snooping attacks.

5 Hands-on 4: Exploration work

5.1 Implement Content Security Policies

5.1.1 CSP for images

Define a web page with embedded images having reference from other machine. Define CSP for image for self and then explore if the images are displayed. Ensure that when access the image directly in the browser, then image display works.

5.1.2 CSP for scripts

Define a web page with scripts linked a javascript file from a different machine. Define CSP to include image only from that URL. Load the web page and ensure the web page works.

5.2 Implement sslstrip

Install **sslstrip**, explore how sslstrip works. Using sslstrip and arpspoof, become the MITM attacker and capture the traffic of your neighbour. Run the arpspoof for your neighbour and default router, and run sslstrrip. Let your neighbour access flipkart.com and then capture the credentials of your neighbour.

5.3 Implement Secure HTTP headers

Read the article on web security in https://acc.digial, and implement other HTTP headers to enable secure web.

5.4 Exercise Expectations

1. Able to understand working of MITM attack.
2. Understand use of Content Security Policy.

Appendix: Generating and installing SSL certificate

1. Follow the steps below to create a self -igned certificate with a private key for the website mywww.com (or choose your own website name) with validity period of 10 year (3650 days).

2. `sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/self.key -out /etc/ssl/certs/self.crt`

It will prompt to provide following information items:

i. Country Name
ii. State or Province Name
iii. Locality Name
iv. Organization Name
v. Organizational Unit Name
vi. Common Name
vii. Email Address.

Use the value `mywww.com` for Common name. This is the website name that will be used in the browser to access the website using HTTPS. The above command creates the required files in the directory **/etc/ssl** but can be kept anywhere as per your preference.

3. Edit the apache web server configuration file `/etc/apache2/site-available/default-ssl.conf` and make/update the following entries.

```
ServerName      mywww.com
SSLEngine on
SSLCertificateFile  /etc/ssl/certs/self.crt
SSLCertificateKeyFile   /etc/ssl/private/self.key
```

4. Enable the SSL changes in Apache web server.
   ```
   sudo a2enmod ssl
   sudo a2ensite default-ssl
   ```

5. Restart Apache web server.
   ```
   sudo systemctl restart apache2
   ```

← end of exercises →