Lab 8 – Phishing Attacks

A phishing attack is a specialized form of social engineering combined with an "injection" technique which “spoofs” an entire web site and tricks a user into giving confidential information to the attacker. When an attack finds a vulnerable server on the web, he/she injects a link on the web site with bad code. The attacker then creates an email that looks legitimate, embedding this malicious link. The user is tricked to click on the link in the email, because the email appears to come from a legitimate source; the user is redirected to the vulnerable web site and the malicious code is send back to the user’s machine. The browser than displays the web page and inadvertently executes the bad code. Phishing attacks are usually used to steal personal or corporate information. Since the vulnerable server has nothing to do with the victim’s machine, the injection technique is called Cross Site Scripting (XSS). The best line of defense is training to recognize emails and web messages that could be phishing attacks:

Exercise 1: Phishing Attack Test

1. Navigate to https://www.sonicwall.com/phishing-iq-test/

   This web site archives real phishing scams and educates users on how to spot them. The test consists of 7 fake and genuine email messages.

2. Read the purpose and instructions before taking the test.

3. Click **Start the Test**.

4. When you are finished click **GET SCORE**. Use the "**Why?"** link on the right side to review your wrong answers. After reviewing your answers, take a screen capture of your score page and save the screen shot and save the file as **LearnName_Test.jpeg**.

Exercise 2: Finding Vulnerable Servers

In this exercise, you will use the same technique an attacker uses to find vulnerable servers. Attackers will write automated spider programs to search for web sites with vulnerable text boxes. When a vulnerable box has been found, the URL to the web site is recorded for future user.

The web site you are going to was improperly coded using CGI.pm, so that you can inject script code into the text box.

1. Navigate to **matrix.senecacollege.ca/~majid.shahravan**

2. Click on the **“Login”** on link.

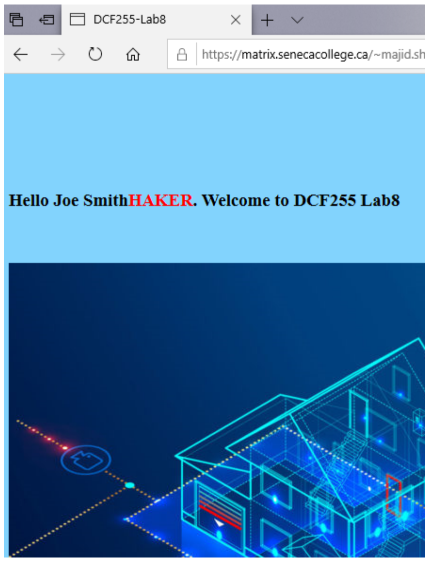
3. Type your name and immediately after your last name (no spaces) type the following:

   <script>document.write ("HAKER".fontcolor("red"))</script>

4. Click **Submit Query**.

When the form is submitted, you should see a web page like Figure 1 with your full name showed with "HAKER" in red color. This is the response attackers are looking for; it shows that the text box allows the injection of script code.

5. Click **OK** to dismiss the alert box.
6. Vulnerability.jpeg



Hello Joe SmithHAKER. Welcome to DCF255 Lab8

Figure 1 Testing the user entery vulnerability

How do you prevent XSS Injection?

To prevent XSS, you can use regular expressions to sanitize user input or approved encoding libraries. For example, if the text box string was passed to the **HTTPserverUtility.HTMLEncode** method. The above input string would be encoded as:

value="&lt;script&gt;alert(&quot;XSS&quot;)&lt;/script&gt;

Notice that the client input has been sanitized and special characters have been converted to their HTML escape characters, for example "<" replaced with "&lt". In addition, the string has been tokenized by semicolons. These combined techniques prevent the browser from running executable code returned from the server.

If using regular expressions, sanitize the string using a "white list" approach (only allowed characters are not filtered out). This approach is better than a "black list" (only disallowed characters are filtered out) because over-time other vulnerabilities may develop based on characters which are not included in the disallowed list which could become a future exploit. For example, in Perl:

White list approach: $var =~tr/A-Za-z0-9\ //dc;
Black list approach: $var =~ s/[\<\>\\"\'\%\;\)\(\&\+] //g;

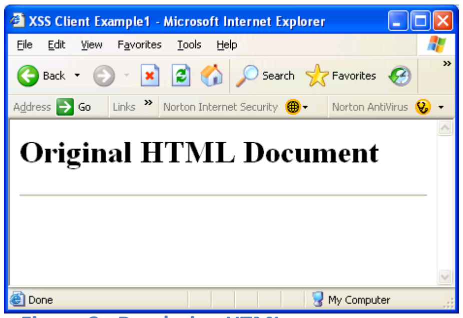
Exercise 3: Client Processing of HTML

After finding a vulnerable server, the attacker will then use standard HTML tags to inject bad code. HTML tags such as <script>, <object>, <applet>, <embed>, <img> or <iframe> can all be used to pass malicious code to the browser. Unfortunately, the current design of browsers aggravates the problem by executing all executable code.

1. Copy and paste the following code to create an HTML document using a text editor

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
<TITLE>XSS Client Example</TITLE>
</HEAD>
<STYLE>
   body { background-color:"#FFFFFF}
</STYLE>

<BODY>
<H1>Original HTML Document</H1>
<HR>
</BODY>
</HTML>
```


XSS Client Example1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address Go Links Norton Internet Security Norton AntiVirus
Done
My Computer

Original HTML Document

Figure 2: Rendering HTML

2. Save the file as **xss1.html.**

3. Open the browser and execute the file. You should see a web page like Figure 2. The browser is the client program of an HTTP server. It has been designed to automatically and quickly execute any executable code sent from a web page in an HTTP Response message

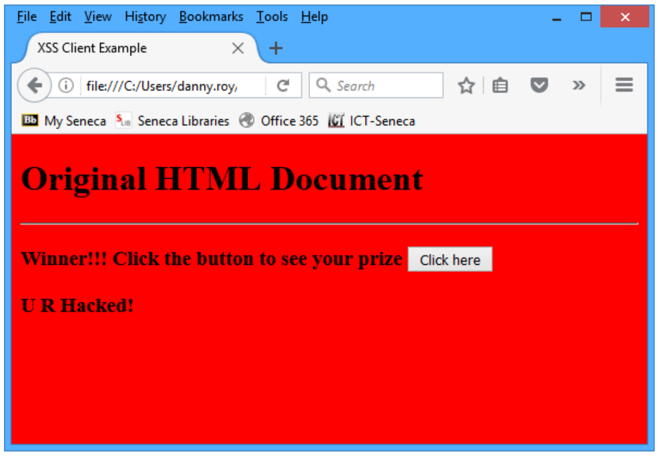
4. Take the screen shot and save the file as **xss1.jpeg.**

5. Now copy and paste the following lines below the <HR> tag

```
<p><H3>Winner!!!  Click the button to see your prize</H3</p>
<button onclick="badcode()">Click here</button>
<p id="demo"></p>
<script>
      document.getElementByID("demo").onclick = function()
      {badcode()};
```

```
        function badcode() {
        document.getElementById("demo").innerHTML = "U R Hacked!";
        document.body.style.backgroundColor= "red";
        }
</script>
```

6. Resave the file as xss1_bad.html.
7. Open the file in your browser.
8. Take the screen shot and save the file as LearnNamexss1_bad.jpeg.



Original HTML Document

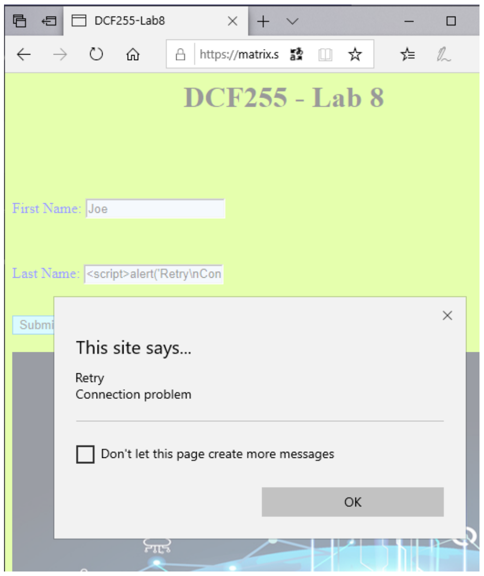
Winner!!! Click the button to see your prize

U R Hacked!

Figure 3: Bad Script Injected into Web Page

Note: If you are using IE7, an information bar will appear.

Click **Allow the Blocked Content**. If you don't see the Information Bar, lower your default Internet security setting. If using Firefox, you may need to enable Javascript in browser properties. You should see a web page like Figure2. (**on the lab computers using FireFox, no changes were necessary**



DCF255 - Lab 8

First Name: Joe

Last Name: <script>alert('Retry\nCon

This site says...

Retry
Connection problem

Don't let this page create more messages

Figure 4: Scripted Distraction

Exercise 4: Exploiting a Server Vulnerability

Once an exploitable site has been found the attacker must construct an attack URL. This is done by viewing the source code of the return string and combining the vulnerable server with the "target system". To distract or to obscure the true intention of the URL, the hacker will often cause some on screen distraction or try to obfuscate the URL; common techniques include **using pop up boxes, using the IP address instead of the domain name, or using some form of encoding to hide** the true web site.

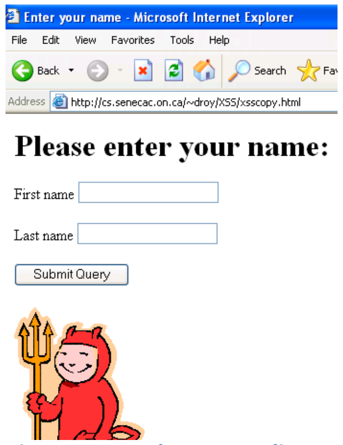
1. Open **matrix.senecac.on.ca/~majid.shahravan** again and enter your name again and add the following code immediately after your last name:

```
<script>alert('Retry\nConnection
problem');window.location="https://scs.senecac.on.ca/~danny.roy/XSS/xsscopy.html" </script>
```

The alert message is to distract the user. When the user clicks the OK button, the script redirects the user to a web site controlled by the hacker. Notice the web site looks the same, the user is now on the SCS server, not Matrix. A little "devil", as in Figure 3, indicates that the web site has been redirected.

Entering information here allows the hacker to capture login information for later account access or retrieve bank account\credit card information. The hacker after collecting personal information could then redirect the user back to the original page – completely transparent to the user (if they were not running Anti-phishing software.

2. Take a screen capture and save the file as **LearnName_Redirect.jpeg**


Enter your name - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Fav
Address http://cs.senecac.on.ca/~droy/XSS/xsscopy.html

Please enter your name:

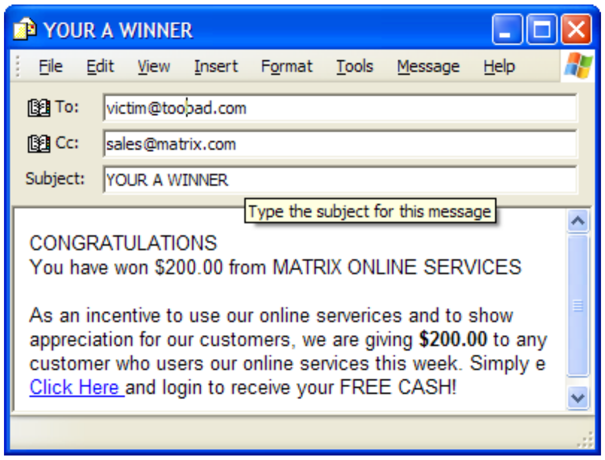
First name

Last name

Submit Query

Figure 5: Attack URL -- Redirected

After finding a vulnerable server and constructing an attack URL, the attacker would send an email message to potential victims – appealing to the basic human instincts. Using the email forging techniques the hacker makes the email seem to come from a legitimate source, bank, paypal, Microsoft, etc. Clicking on the link in Figure 5 would navigate the user to the matrix logon server which would send the embedded JavaScript and HTML code back to the browser to be executed and redirect the user to an identical, but malicious site. The attacker could just as easily have created a virus which would delete or corrupt local files. Lastly, note that even an SSL connection is not secure, if the malicious code is inserted before the encrypted traffic begins. Scripts can then send encrypted data to the server or even to another server, and if that server is SSL capable, it won't even display a warning about an insecure connection. An excellent web site is www.CERT.org which acts as a clearing house for all malware and software vulnerabilities. As a web developer, it would be prudent to join CERT's mailing list. To read more see http://www.cert.org/tech_tips/malicious_code_mitigation.html


YOUR A WINNER
File Edit View Insert Format Tools Message Help
Type the subject for this message

To: victim@toobad.com

Cc: sales@matrix.com

Subject: YOUR A WINNER

CONGRATULATIONS
You have won $200.00 from MATRIX ONLINE SERVICES

As an incentive to use our online serverices and to show appreciation for our customers, we are giving **$200.00** to any customer who users our online services this week. Simply e Click Here and login to receive your FREE CASH!

Figure 6: Malicious Email with Forged Address

For Grading:

Submit the following files for grading.

- **LearnName_test.jpeg**
- **LearnName_vulnerability.jpeg**
- **LearnName_xss1.jpeg**
- **LearnName_xss1_bad.jpeg**
- **LearnName_Redirect.jpeg**