Network Security
Homework Assignment 4

Due date: November 14th
8 points

CS5285
2016 Fall

Problem 0:

In this homework assignment, you will find and exploit web vulnerabilities. As your 0th task, please download the virtual machine for this assignment from Blackboard and import it into Virtualbox. Since you will have to access the webserver running on the virtual machine, you will first need to set up host-only networking before you can start the virtual machine:

- open Preferences in VirtualBox, and select Network / Host-only Networks
- click "Add new host-only network", select the new network, and click "Edit the selected host-only network"
- make sure that the IPv4 network mask is 255.255.255.0 and the IPv4 address is 192.168.56.1 (or any other address in that range except for 192.168.56.2, which is used by the virtual machine)
- select the virtual machine and click Settings, and select Network / Adapter 1
- enable Adapter 1, attach it to "Host-only Adapter", and select the host-only adapter that you have just created.

After you have finished with the above steps, start the virtual machine. Once the machine is up and running, you will be able to access the webserver by typing the address 192.168.56.2 in your web browser. When you open this address, you should see a simple login page.[1]

Attached to this problem description, you will find the PHP script files running on the webserver:

- login.php: Displays the login webpage, which consists of a simple HTML form.
- dologin.php: Handles login request, displays simple error messages if the login information is invalid, and redirects the user to index.php if the login information is valid. Note that it also saves the login information in a cookie, so that users stay logged in.
- index.php: Displays the welcome webpage, which consists of links pointing to forum.php and gallery.php, as well as a simple HTML form for changing the color theme.
- forum.php: Implements a simple web forum, enabling users to post and view messages.

[1] Please note that the homework assignment has been tested with the latest versions of Google Chrome and Apple Safari, but it should work with any other modern browser as well.

- search.php: Enables users to search the messages.
- gallery.php: Implements a simple web gallery, enabling users to upload, view, and remove JPEG images.
- logout.php: Enables users to log out by removing the login information cookie.
- checklogin.php: Checks the existence and validity of the login information cookie, and redirects the user to login.php if they are not logged in. Included by all webpages that are accessible to only logged in users.
- header.php: Basic HTML and CSS formatting for webpages based on the color theme selected using index.php. Included by most webpages.
- footer.php: Displays login information and the option for logging out, as well as closes the HTML document. Included by most webpages.
- dark.css and light.css: Simple CSS style files used by header.php.

There are two additional files on the webserver:

- resetdb.php: Resets the database into its initial state. To access it, simply type 192.168.56.2/resetdb.php into your web browser.
- dbconn.php: Connects to the SQL database. Included by webpages that need to use the database server.

Happy hacking!

Please feel free to provide feedback regarding the course at:
https://docs.google.com/forms/d/e/1FAIpQLSc8YlsfcQKNvhAfYI7_ilkaqRk3UVmCJ67O2tdlAScptRO0Fg/viewform

Problem 1 (2 points): SQL Injection

Use the SQL injection vulnerability at line 8 in dologin.php to gain access to the website!

- Look at line 8 in dologin.php to learn the name and the columns of the table that stores login information.
- Construct a simple SQL query that creates a new user.
- Inject the SQL query on the login page.

Briefly describe how you would fix or avoid this vulnerability!

Problem 2 (2 point): Cross-Site Scripting

Use the cross-site scripting (XSS) vulnerability in forum.php to inject a client-side script into the website using a specially-crafted message!

- Look at line 16 in forum.php to see what defense the website has against XSS!
- Construct a script “tag” that evades (or rather “tricks”) this filter (see lecture slides on XSS for help)!
- Inject Javascript code that creates a pop-up message (alert function) displaying the user’s cookies (document.cookie), demonstrating that the injected code has access to the user’s login information.

Briefly describe how you would fix or avoid this vulnerability!

Problem 3 (1 point): File Inclusion

Use the file inclusion vulnerability in header.php to access files on the webserver!

- Change the theme on the welcome page and look at the URL of the request!
- Look at how the request is used by header.php.
- Include the file /etc/passwd to demonstrate that you can read arbitrary files (to which the webserver has access). Note that the script header.php runs in the directory /var/www/, so be sure to use the right path.
- Open the source of the response webpage (e.g., right click and "View Page Source" in Google Chrome) to verify that /etc/passwd was indeed included.

Briefly describe how you would fix or avoid this vulnerability!

Problem 4 (1 points): Command Injection

Use the command injection vulnerability in gallery.php to run arbitrary system commands!

- Look at line 17 of gallery.php to see how the removeImage GET parameter is used.
- Construct a parameter value that executes another command in addition to the rm command (see lecture slides for examples on multiple commands in a single line).
- Inject the command "cat /etc/passwd" to demonstrate that you can run arbitrary system commands, and verify that /etc/passwd is displayed.

Problem 5 (1 point): File Upload

Use the previous file inclusion vulnerability and the unsafe file upload implementation in gallery.php to upload and execute a script on the webserver!

- Upload the script shellcode.php using the file upload form of the gallery. Look at line 23 of gallery.php to see how simply you can trick the webserver into accepting arbitrary files.
- Use the file inclusion vulnerability from Problem 3 to execute the shellcode. Note that the server stores each file using a random name, so you first have to look at the gallery to see what filename the shellcode got.

Note that you can recover normal functionality of the website by erasing your cookies in your web browser or by using the file inclusion vulnerability to include light.css.

Problem 6 (1 point): Reflected XSS

Use the reflected XSS vulnerability in search.php to construct a URL that injects a client-side script into a client that requests this URL.

- Look at line 18 of search.php to see how the keyword is used. Notice that it is used inside an attribute inside of an HTML tag, which means that your specially-crafted keyword will first have to close those.
- Try using the exploit that you used for Problem 2 in the search field to inject a client-side script displaying the user's cookies in a pop-up message (of course, you now have to start the exploit with closing the attribute and tag).

- On most modern browsers, nothing will happen due to countermeasures built into the web browser. More specifically, the browser will detect that the server has reflected the script that was sent in the request URL, and will block the execution of the script.
- In Google Chrome, you can check this by opening the menu / “More Tools” / “Developer Tools” and looking at the XSS Auditor error. In Apple Safari, you can open the “Develop” menu and select “Show Error Console”.
- Defeat this countermeasure using the filtering implemented on the webserver. More specifically, modify the name of the alert function so that the filtering will transform it back to “alert”.

Note that before you start working on this problem, you should first use the resetdb.php page to reset the database, and then redo the SQL injection exploit to gain access to the website. Otherwise, the search page might show the result of your earlier XSS exploit, which will make it difficult to verify if you have solved the current problem.