Homework Assignment 4

Cybersecurity

COSC 3371 / 2022 Spring

In this homework assignment, you will find and exploit vulnerabilities in a website. The website will be served by a webserver running on the virtual machine that you used in Homework Assignment 3. First, please make sure that you can connect to the webserver running on the virtual machine:

- First, make sure that the networking settings of the virtual machine allow connection from your host: right-click on the virtual machine in the main screen of VirtualBox to open the virtual-machine settings window (note that these settings are different from the general preferences of VirtualBox), select the "Network" tab, make sure that "Enable Network Adapter" is checked, and select an adapter that enables connection from your host (e.g., "Bridged Adapter").
- Once you have logged in to the virtual machine through the VirtualBox window, you can determine the IP address of the virtual machine by running the command `ifconfig` and reading the address after `inet addr:` (make sure that you are looking at the real interface, not the loopback `lo` interface).
- Then, you should be able to connect to the webserver by entering `http://<IP address>/index.php` into a web browser running on your host.

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 error messages if the login information is invalid, and redirects the web browser to `index.php` if the login credentials are 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 contains links to `forum.php` and `gallery.php`, as well as a simple HTML form for changing the color theme.
- `forum.php`: Implements a web forum, which allows users to post and view messages.
- `search.php`: Enables users to search the messages posted on the forum.
- `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 web browser to `login.php` if it is not logged in. Included by all webpages that are accessible only to users who are logged in.
- `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 as well as a link for logging out using `logout.php`, and 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 SQL database to its initial state. To run it, simply type `http://<IP address>/resetdb.php` into your web browser.
- `dbconn.php`: Connects to the SQL database. Included by webpages that need to use the database server.

As your solution, please submit a single text file (e.g., Word, PDF, simple text) on Blackboard. For each problem, please **explain your solution** and—where requested—**describe how you would fix or avoid the vulnerability**.

Happy hacking!

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 columns of the table that stores user login information.
- Construct a simple SQL query that creates a new user.
- Inject the SQL query on the login page.

As your solution, submit the exploit and briefly describe how to fix or avoid this vulnerability.

Problem 2 (2 points): 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.

As your solution, submit the exploit and briefly describe how to fix or avoid this vulnerability.

Problem 3 (2 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 observe 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 select "View Page Source" in Google Chrome) to verify that `/etc/passwd` was indeed included.

As your solution, submit the exploit and briefly describe how to fix or avoid this vulnerability.

Problem 4 (2 point): 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.

As your solution, submit the exploit and briefly describe how to fix or avoid this vulnerability.

Problem 5 (2 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` (attached on Blackboard) 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 was assigned to the uploaded shellcode (e.g., right click and "Open Image in New Tab").
- Note that you can recover the normal functionality of the website by erasing the cookies in your web browser or by using the file inclusion vulnerability to include `light.css`.

As your solution, describe your exploit, including how you can use the uploaded script to display `/etc/passwd` (e.g., asking to execute `cat /etc/passwd`).

Extra Credit:

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 any web browser that requests this URL.[1]

[1] 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.

- Look at line 18 of `search.php` to see how the search keyword is used. Notice that the keyword is included in an attribute that is inside an HTML tag, which means that your specially-crafted search keyword will first have to close the attribute and then the tag.
- 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 the tag).
    - On some web browsers (e.g., some older versions of Google Chrome), nothing will happen due to countermeasures built into the web browser. More specifically, the browser will detect that the server has reflected a script that was sent with the request URL, and it will block the execution of the script.
    - In older versions of Google Chrome, you can examine this functionality 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".
    - You can defeat this countermeasure by exploiting the filter implemented on the webserver. More specifically, modify the name of the `alert` function so that the filter in `search.php` will transform it back to "`alert`".

As your solution, submit the exploit and briefly describe how to fix or avoid this vulnerability.