Homework 7 – Session Security

Lab Information

Due Date:

Homework 7 Dropbox Deadline

Objectives/Goal:

In this homework we will be investigating part one of client side attacks. While you are free to experiment however you'd like – note that this lab is not intended to be completed by leveraging XSS. The environment which you are entering is also hostile so be aware and use private browsing as needed.

Deliverables:

- A demonstration of the exploit in Activity 1 with the fixed code.
- Provide a screenshot proving the existence of the attack and a PowerPoint explaining the fix
- Show that you can login as a result of the data from Activity 3
- Demonstrate that you can successfully Brute Forced two accounts

Table of Contents:

Lab Information 1

Activity 1: A Fixation on Security 2

Activity 2: I Predict Weak Security in Your Future 2

Activity 3: Caffeine Hi for Jack. 3

Activity 4: Take a Sip of This Brute 3

Activity 1: A Fixation on Security (30%)

We are going to retarget our vulnerable client again on the backend. This script will click any link that you post to the following message board (http://csec380-core.csec.rit.edu:5007/). He also really likes Armbook and will check it whenever given the opportunity to do so.

Step 1: Send that Session ID (0%)

As we learned in class, it is often possible to provide a user with a session. Of course we demonstrated using a GET parameter, but this is unrealistic as we know that no one would DREAM of sending session ID's or other sensitive information via GET. Take a look at the NEW ARMBOOK (http://csec380-core.csec.rit.edu:97). Investigate how it stores sessions (take a look at the provided source code).

NOTE: The port number on the assignment is 97, not 87. This is due to the restrictions of browsers. Browsers try and restrict connection to ports that are associated with services that are plaintext. The easiest example to picture would be port 23. This would be the port for Telnet. If it were possible to access Telnet servers via a browser, attackers could try and send XHR requests that trigger Telnet commands on local servers.

Step 2: Find out about the Client (15%)

Once you've figured out the session management, you need to trick the client such that you'll be able to get access.

Either take a video or a number of screenshots that demonstrate you actively exploiting the attack against the message board client.

Step 3: Fix the client (15%)

Now take the code for the new Armbook and fix the session management such that this problem doesn't occur again. You may not use PHP sessions. You should rearchitect the application to prevent this issue, not merely make it more difficult to exploit. *Note/Hint: If you're having problems logging in after you setup the submitted code, may need to configure the "request_order" option within your php.ini to be "GPC".*

Provide the fixed code as part of a Dockerfile that launches Armbook.

Activity 2: I Predict Weak Security in Your Future (25%)

Alright you have recommended a fix for the previous session issue, great job, but there are other problems with our sessions. Someone has breached even the fixed version of Armbook and it is your job to figure out how they did that!

Step 1: Testing for fitness (15%)

Both Burp and ZAP have some capabilities to analyze session information. Use either Burp or ZAP to collect and analyze the security of the session ID generated by the new Armbook. Figure out how they snuck through by demonstrating it.

Demonstrate the issue by showing a screenshots of the randomness results from either Burp or ZAP.

Step 2: Recommended changes (10%)

Rather than fixing it in the code, the developers wanted to take a crack at fixing this one themselves, explain what went wrong and how they would fix it. Make a mini presentation for them highlighting the areas in the code that are the problem and the right way to go about fixing it in their language.

Make a Powerpoint presentation that describes what is needed, at the code level, to fix this issue correctly. Take a detailed look at the documentation to ensure that it is indeed secure.

Activity 3: Caffeine Hi for Jack. (20%)

Man there is nothing our Hooli employee loves more than Coffee in the morning. You might find that there is a Starbucks Coffee nearby, that's probably where he went. It is pretty likely that our Hooli employee will login and browse Armbook every so often over this connection. (See the capture file)

Step 1: Punish Him!

Leverage our Hooli employees need for caffeine to gain access to his account.

Show that this attack is functional by finding the username and password, and logging in.

Activity 4: Take a Sip of This Brute (25%)

If all else fails, you can always go for broke. In most cases this is some sort of Brute Force attack. It is important to note that while many authentication systems have limits, often alternative login methods like mobile API's will not.

Step 1: Finish Him!

Use Burp or ZAP in order to brute force the login credentials. Note, while we have demonstrated Burp in class, it will be throttled in its demo version. Demonstrate that you now know at least two (non-student) user account passwords

Provide the password for two accounts that you didn't have access to before.

Signoffs

Activity 1.1 – Demonstrate that you can trick the client and log in as them. (15%)

Activity 1.3 – Provide your fixed code as a Docker compose file. (15%)

Activity 2.1 – Demonstrate evidence of the discovered problem. (15%)

Activity 2.2 – Provide your presentation on how to fix the project. (10%)

Activity 3.1 – Provide a screenshot showing that you isolated the username and password and that you can login. (20%)

Activity 4 – Show that you can brute force the password (25%)