Web Security Hands On Activity (Part 2)

In this session we will continue to explore web attacks possible on the Damn Vulnerable Web Application (DVWA). You will submit another lab report as you did for Part 1, with your observations and description and screenshots of all steps performed. For your convenience, all the instructions pertaining to what you should include in your report are in **green** font below. You may do this activity in pairs; please submit your report as a pdf file named as follows:

Section_Task2_yourRollNumber_yourPartner'sRollNumber.pdf

Only one group member should submit the report; if your pair is cross-section, please write your section as CD.

Part 1: CSRF Attack Under Low Security

Part 1A

Open Burpsuite and make sure Intercept is OFF. Now open the integrated Chromium browser and go to localhost/dvwa. Set the security to LOW. Now go to the CSRF tab from the menu on the left. View the source code to understand how you can exploit the CSRF vulnerability. In your report, under "Part 1A", write down your observation regarding the security of this code.

Part 1B

Now submit a legitimate password change request through the CSRF page open in Chromium, and observe your packet in Burpsuite (from HTTP History). Observe where the new password is being transmitted. In the Repeater, edit the request to initiate a new password change, and repeat the request. Logout and then login to DVWA again to make sure that your password change through a repeated request from Burpsuite went through successfully. In your report, under "Part 1B", show a screenshot of Burpsuite where your new password is visible in the HTTP request that you have edited, and also describe what you did (i.e. which part of the request you edited).

Part 1C

Now your job is to figure out how you can achieve this via a CSRF attack. Basically under a CSRF attack you need to create a separate HTML file with code that sends the same request as the password change request you submitted in Part B. When you open that HTML file in the same browser that is running DVWA, and click the button behind which your request-sending code is written, it should send the request to DVWA, appending your cookie to it.

Here's how you can do this: you should select the "Change" button on the CSRF page in DVWA and right click, and select "Inspect Element". This shows you the code behind the button. You can copy this code to a separate HTML file, making appropriate changes to the input types (make them "hidden" instead of "password"). Then open this HTML file in another tab of Chromium that is running DVWA and click the Change button. It should end up changing your DVWA password. You need to verify this by logging out of DVWA and logging back in using the new password that you changed through CSRF attack. However, this may not work because of an incorrect cookie value. Turn Intercept to ON in Burpsuite and intercept the request that is sent when you click the button in the malicious html page you created. Check whether it is sending the right cookie value; if not, edit it manually until your attack succeeds.

First try this on your own; if you get stuck, you can refer to this link for more help: https://medium.com/@xBBsec/csrf-low-dvwa-writeup-8f7c9285dd84

In your report, under "Part 1C", describe how you carried out this attack, any issues you ran into, and how you fixed them. Also describe how a real attacker would carry out this attack.

Part 2: CSRF Attack Under Medium Security

In DVWA, change the security level to Medium and then go to CSRF again. First make sure you turn Intercept to ON in Burpsuite before you begin the attack; it seems there is a bug in DVWA that always sets security level = impossible in your cookie when carrying out CSRF attacks, so you will need to intercept the request, change the security level to medium from the cookie, and then forward the request. Only then it will take you to the medium security level.

Once you are sure that security level has been set to medium, view the source code and observe the changes in the code. In your report, under the heading "Part 2", describe the new security measures. Do you think your attack from Part 1C should work? Why or why not? Answer this question before actually trying out the attack.

Now execute the same attack that you did in Part 1C. Report your findings (with a screenshot) – did it work? Was this what you expected? In case you encountered unexpected behaviour, think of an explanation and write it down in your report.

Part 3: CSRF Attack Under High Security

Part 3A

Now change the security level to High and then go to CSRF again. Once again, make sure that you are seeing the code high.php, not impossible.php. If you keep seeing impossible,php, intercept your request with Burpsuite and see whether any part of the request is specifying impossible security level instead of high. Change it and get to high.php. In your report, under "Part 3A", describe how you got high.php and what part of the request (if any) you had to change in Burpsuite. Now observe the code and describe the security measures that have been put in place. Discuss any way you can think of that it can still be exploited.

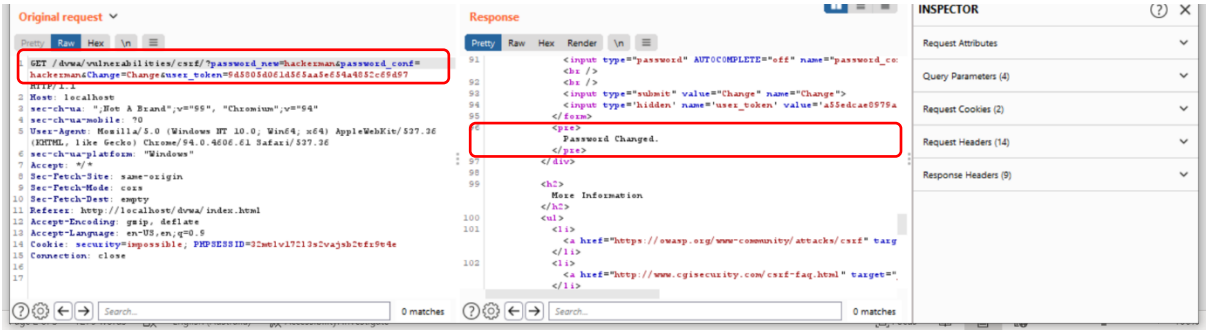
To describe the security measures fully, look at where the token is being generated? You have to look at the inspect source and see that a token is being generated as a hidden input, which then gets passed in the request (you can see in Burpsuite). Attach screenshots of both tokens, one in the inspect element and one in Burpsuite.

Now once again try to execute the same attack that you did in Part 1C. What is the result? Attach a screenshot.

Part 3B

Actually, to successfully carry out this attack you need to somehow grab the user token and attach it to your password change request. One way to do this is through a script that gets the user token from the DOM. See the attached index.html file; it is trying to obtain the user's token from the CSRF page. Put this file in your DVWA folder and then open it in Chromium (by typing localhost/dvwa/index.html). Make sure Intercept is ON in Burpsuite as you need to manually edit the cookie to set security level to high (not impossible). See how eventually it sends a password change GET request with the user token included. Observe the HTTP history in Burpsuite and make sure that this last GET request gets a response which says "Password changed". In your report, under "Part 3B", show a screenshot of this

request and response history from Burpsuite, with the GET request holding the user token visible in the request and the “Password Changed” line visible in the response. A sample screenshot is attached below.



Part 3C

Essentially this is a stored XSS attack. If you were to place this file anywhere else except inside your Xampp directory, and then opened it in your browser (which is akin to visiting the page hosted anywhere except the server running DVWA), the Same Origin Policy would stop this script from executing on the DVWA server. However we manually placed it inside Xampp. An attacker would use some server-side vulnerability to put this file on the server. For example, using the file upload vulnerability on DVWA, you can upload this file and it actually gets placed inside the DVWA/hackable/uploads directory. Try doing this and, under “Part 3C”, attach a screenshot of (1) uploading the file and (2) the file appearing inside Xampp/hdocs/DVWA/hackable/uploads.

Part 3D

Now set the security level to impossible. Observe the code to see what has changed, and why it is now impossible to attack the password change feature through CSRF. Describe your understanding in your report, under “Part 3D”.

This completes your lab session.

Related resources:

Read the following article to understand more about the File Upload vulnerability:

https://www.acunetix.com/websitesecurity/upload-forms-threat/

This article is included in your course material and you may be asked about it in your quizzes and exam.