Name: Michael Young (myoung10)

Course: Adv Hacking

Assignment: 7-2

*NOTES: Two interesting spaces I did not explore, because I already discovered 2 ‘vulnerabilities/issues’ are in continuing down the Postman path (#3) and messing with the headers and seeing what I can inject in the upload. The other interesting opportunity is the ‘eval’ command, and how to get by some of the text preprocessing to allow me to input a malicious payload into eval. And maybe changing the content type to be something fun (an executable possible?). Maybe forwarding the user to another page where more damage can be done. Ideas are plentiful, but here is my submission.*

what is the vulnerability/problem (1 pts x 2 vulns)

1. You can download the previous cached file by the system.
2. Filename download input is not parsed correctly and can create an ‘error’ page
3. Unresponsive page without (seemingly) timeout that ties up all the resources if sending modified POST payload.

what is the payload/attack (1 pts x 2 vulns)

1. Just click the ‘Download your sample!’ button, and the previous file a user created will be downloaded.
2. \n\n<script>alert(document.domain);</script><!-- or for the full URL to give to the user: <http://localhost:8000/download?file=%5Cn%5Cn%3Cscript%3Ealert(document.domain);%3C/script%3E%3C!-->
3. I went into Postman and entered <http://localhost:8000/upload> for the URL, POST for method, entered in ‘asdfasdfasdf123’ for the payload, and added headers of ‘Content-Length: 100’. There seems to be a lot of opportunities around modifying the headers to cause unusual behavior.

what observation you made which lead to you finding the issue (1 pts x 2 vulns)

1. I modified the file, clicked download, refreshed the page without a query, and hit download again.
2. I input a bunch of \n\n\n\n\n in the download file input field and realized it went to another page instead of downloading the file, and also a backslash was added to the display screen.
3. I simply tried to do a POST request outside of the page (Postman) and modified a single header I thought might be important.

the impact of the issue for a victim/user (1 pts x 2 vulns)

1. Potentially sensitive information from the previous user can be retrieved.
2. If a user is to click this link for a downloaded file, I can retrieve any sensitive cookie information from that domain for that user. Or I can embed javascript that will perform operations on the user’s behalf.
3. After sending this request and seeing it run for 5 minutes (at least) without timing out, I visited the localhost:8000 in my browser, and viewed that it would no longer load. This is effectively a DOS attack on the system.

how you would fix the problem (1 pts x 2 vulns)

1. Create/download the file in the same step, and send it via buffer to the user directly rather than saving to disk. If needing to save to test, uniquely identify it, prevent other Ips from downloading it, expire the file, prevent more than 1 download, etc.
2. In addition to #1, I would only allow alphanumeric characters in the field for the name (or generate an automatic unique name, not controllable by the user). Also, if there does happen to be an error, don’t display the page, but rather fail with a generic error message.
3. Have a timeout for the webserver if it fails to serve the request within X seconds. Deny the request if it doesn’t have the appropriate headers/information. Thread the application so one long request doesn’t hang the whole system. Separate the API from the server and protect the API so that only the main server can make requests to it.