Lab: Remote code execution via web shell upload

**1. Summary**

**Bug Title:** A file upload vulnerability in image upload function It doesn't perform any validation on the files users upload before storing them on the server's filesystem.

**Severity:** High.

**Description**: A File Upload vulnerability was identified in the image upload function of the application. This vulnerability allows an attacker to upload a php web shell and gain RCE

**Date Discovered:** 17/8/2024.

**Status:** Solved.

**2. Bug Details**

**Vulnerability Type:** Unrestricted File Upload .

**Affected URL/Endpoint:**  /my-account/avatar.

**Description:** The application has a critical File Upload vulnerability in its image upload function. This vulnerability arises from the absence of validation or sanitization on the files uploaded by users before they are stored on the server's filesystem. Without proper validation, an attacker can upload malicious files, such as scripts or executables, which can then be executed or accessed by unauthorized users, leading to severe security implications. The image upload functionality is designed to allow users to upload images, presumably to be displayed or processed by the application. However, the lack of validation on the uploaded files means that the server does not check the file type, size, or content before saving it to the filesystem. For example, if the application expects only image files (e.g., .jpg, .png), but does not enforce this expectation, an attacker could upload a file with a malicious payload disguised as an image file or with a dangerous file extension (e.g., .php, .exe). Once stored on the server, the malicious file could be executed, leading to unauthorized code execution, data exfiltration, or further compromise of the system.

**Steps to Reproduce:**

1. Go to Lab URL: <https://portswigger.net/web-security/file-upload/lab-file-upload-remote-code-execution-via-web-shell-upload> and click access lab

2. log in using these credential wiener:peter

3. you will find and upload feature to upload an avatar as below



4. make this php web shell by creating a php file with any name you want (i named exploit.php) with the content

<?php

echo file\_get\_contents("/home/carlos/secret");

?>



5. upload the php web shell via upload feature and access it via /files/avatars/exploit.php ot by right cilck on the avatar and open image in new tap



**Proof of Concept (PoC):**



**Impact:** Remote Code Execution: If an attacker uploads a malicious script (e.g., PHP or shell script) and is able to execute it, they could gain control of the server, leading to full system compromise**.**

**3. Recommendations**

1. **File Type Validation:** Implement strict validation to ensure that only specific file types (e.g., .jpg, .png, .gif) are allowed to be uploaded. Use a whitelist approach to enforce this restriction**.**
2. **Content Validation:** Analyze the content of the file to ensure it matches the expected format for the allowed file types. This can include checking the MIME type and file signature to verify that the file is indeed an image and not a disguised executable or script.
3. **File Size Restrictions:** Limit the size of uploaded files to prevent potential Denial of Service (DoS) attacks caused by uploading excessively large files.
4. **Rename Uploaded Files:** Rename uploaded files to a safe format and store them in a non-executable directory. This reduces the risk of uploaded files being executed.
5. **Store Files Outside Webroot:** Store uploaded files in a directory outside of the web root to prevent direct access to them via a URL.
6. **Set Proper Permissions**: Ensure that uploaded files have the least privilege required, preventing them from being executed or modified by unauthorized users.
7. **Use Secure File Upload Libraries:** Leverage secure file upload libraries or frameworks that provide built-in validation and sanitization mechanisms**.**
8. **Monitor and Audit File Uploads:** Implement logging and monitoring to detect and respond to suspicious file upload activities.

**4. Conclusion**

**Summary:** The presence of an unrestricted file upload vulnerability in the image upload function poses a significant threat to the security of the application. This vulnerability can be exploited by attackers to upload and execute malicious files, potentially leading to remote code execution, data breaches, and full system compromise. Fixing this vulnerability is crucial for maintaining the security posture of the application. By enforcing strict validation on file uploads and implementing secure coding practices, the organization can prevent unauthorized access and execution of malicious files, safeguarding both the system and its users. Addressing this issue is essential not only for protecting sensitive data and system integrity but also for ensuring compliance with security standards and maintaining customer trust.

**5. Appendices**

**Tools Used:**  Browser.

**References:**

* <https://portswigger.net/web-security/file-upload>
* <https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload>
* <https://learn.snyk.io/lesson/unrestricted-file-upload/>