Sri Lanka Institute of Information Technology

File Upload Vulnerability - Report 03

IT23187214

Web Security - IE2062

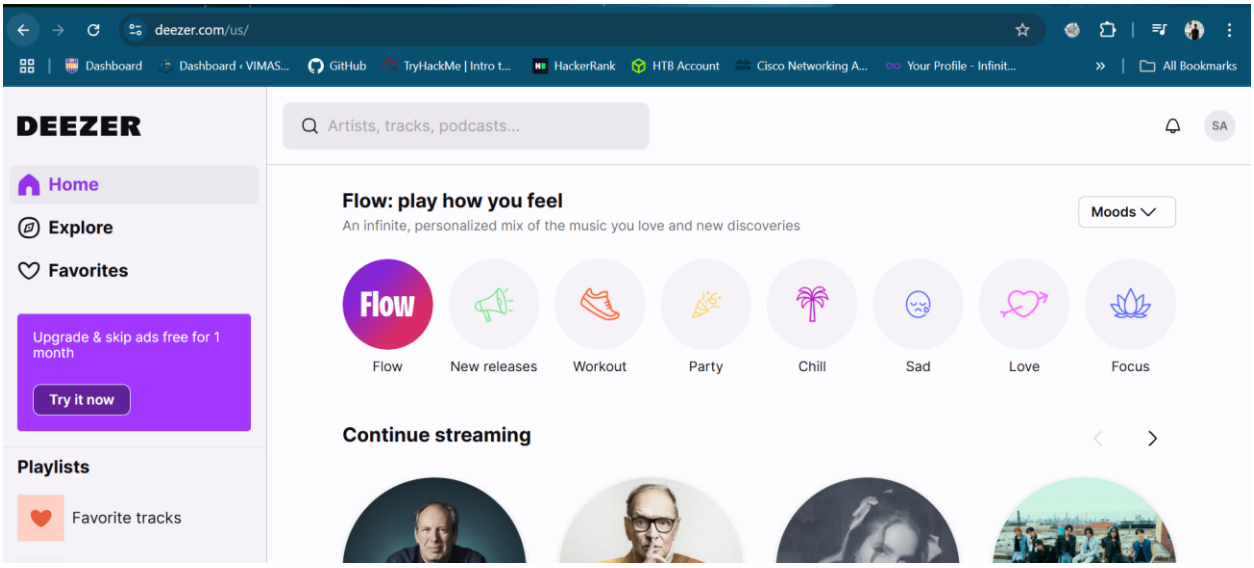
Vulnerability Title:

File Upload Vulnerability

Vulnerability Description:

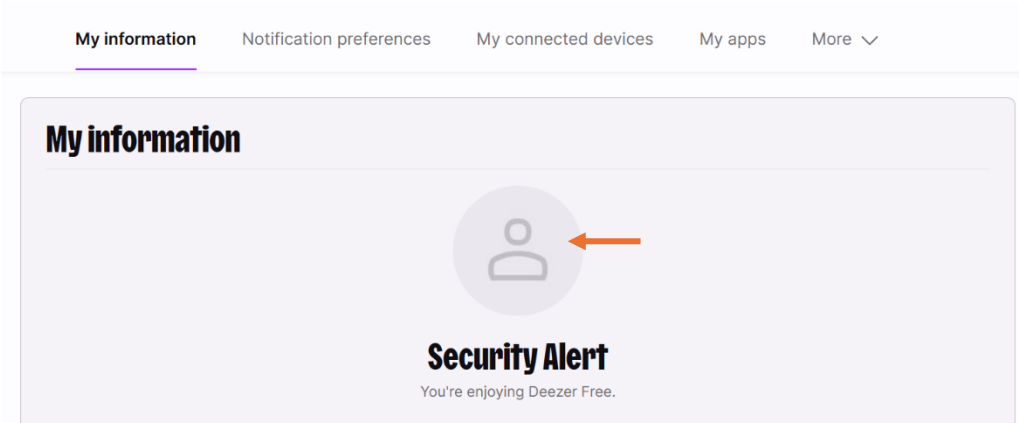
I found this program on the yeswehack Bug hunting website. The website hosted at https://www.deezer.com. Improper file upload validation can allow attackers to upload files that appear to be safe (e.g., .jpg, .png) but contain malicious code. If the server does not check the contents of the file and does not enable execution of the uploaded file, an attacker can run any code remote.

This test mimics a malicious php shell or image (shell.php.jpg) with a .php extension and embeds a file uploaded via the image-upload function of the image. The payload was designed to allow remote commands with CMD parameters.



Affected Components:

- File Upload Form / Endpoint



Impact Assessment:

- **Risk Level:** High

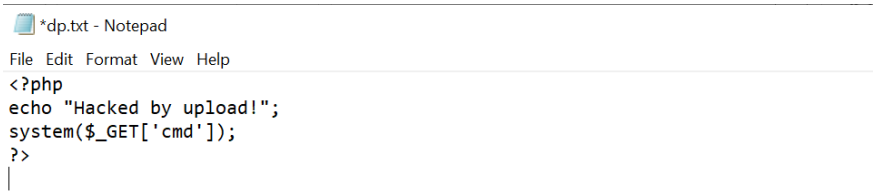
If successful, this attack would have allowed remote code execution on the server, giving full control to the attacker through the browser. This includes:

- Execute commands on the server (whoami, ls, etc.)
- Read or delete files
- Upload more malicious scripts
- Potentially compromise the entire system

However, in this case, the file upload was successfully mitigated. The server blocked the malicious file, renamed it, stripped the executable extension, or stored it safely preventing execution.

Steps to Reproduce:

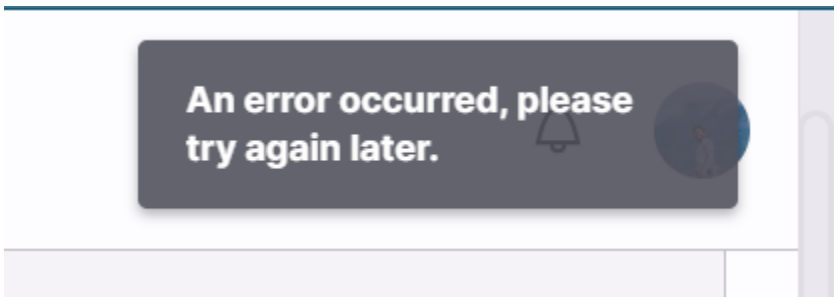
1. Created a file named dp.php.jpg containing the following code:



- I tried using both dp.php.jpg and dp.jpg.

2. Uploaded the file using the image upload feature on the website.

Proof of Concept (PoC):



1. **Filename Used**: dp.php.jpg/dp.jpg
2. **Payload**: Embedded PHP code with system($_GET['cmd'])
3. **Result**: The server **did not execute** the script,
   - The upload was blocked,
   - The file extension was sanitized, or
   - PHP execution was disabled in the upload directory

Proposed Mitigation or Fix:

Even though the server is already secure, here are the recommended practices that are likely in place and should be continued:

- Strict File Extension Checks - Disallowed .php, .asp, .js, etc.
- MIME Type Validation - Ensured only valid images like .jpg, .png are accepted
- Stored Files Outside Web Root - Prevented browser access to uploaded files
- Script Execution Disabled - Upload directory configured to block PHP execution
- Content Inspection or File Scanning - Detected PHP code inside an image wrapper
- Renaming Uploaded Files - Removed dangerous extensions or randomized names

Conclusion:

The application is well protected from file upload attacks. Even advanced payloads hidden in image files have been blocked or become harmless. This shows strong backend validation and a good server configuration based on one of the top 10 OWASPs.