**Title: [Authenticated File Upload → Remote Code Execution] in [Pet-Grooming-Management-System] <= [v1.0]**

**BUG Author: Kuldeep Sahu**

**Product Information:-**

**Vendor Homepage**: https://www.sourcecodester.com/

**Software Link**: [https://www.sourcecodester.com/php/xxxx/pet-grooming-management-system.html]

**Affected Version**: <= v1.0

**Vulnerability Details**

**Type**: Authenticated File Upload → Remote Code Execution (RCE)

**Affected URL**: http://localhost/petgrooming\_erp/pet\_grooming/admin/profile.php

**Vulnerable Parameter**: File Upload Field (Profile Image)

**Vulnerable Files**:[admin/profile.php]

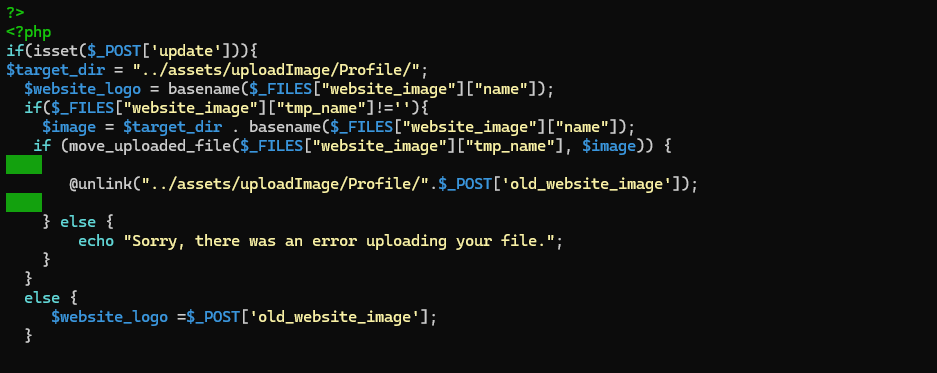
**Web accessible upload directory** http://localhost/petgrooming\_erp/pet\_grooming/assets/uploadImage/Profile/

**Vulnerability Type**: CWE-434 (Unrestricted Upload of File with Dangerous Type)

**Severity Level**: CRITICAL (CVSS: 9.8)

**Root Cause**

The application fails to validate or sanitize uploaded files. Attackers can upload PHP webshells (e.g., shell.php), which are stored in a publicly accessible folder. Direct access to the file allows remote code execution.



**Impact**

Remote Code Execution on server

Full compromise of web application

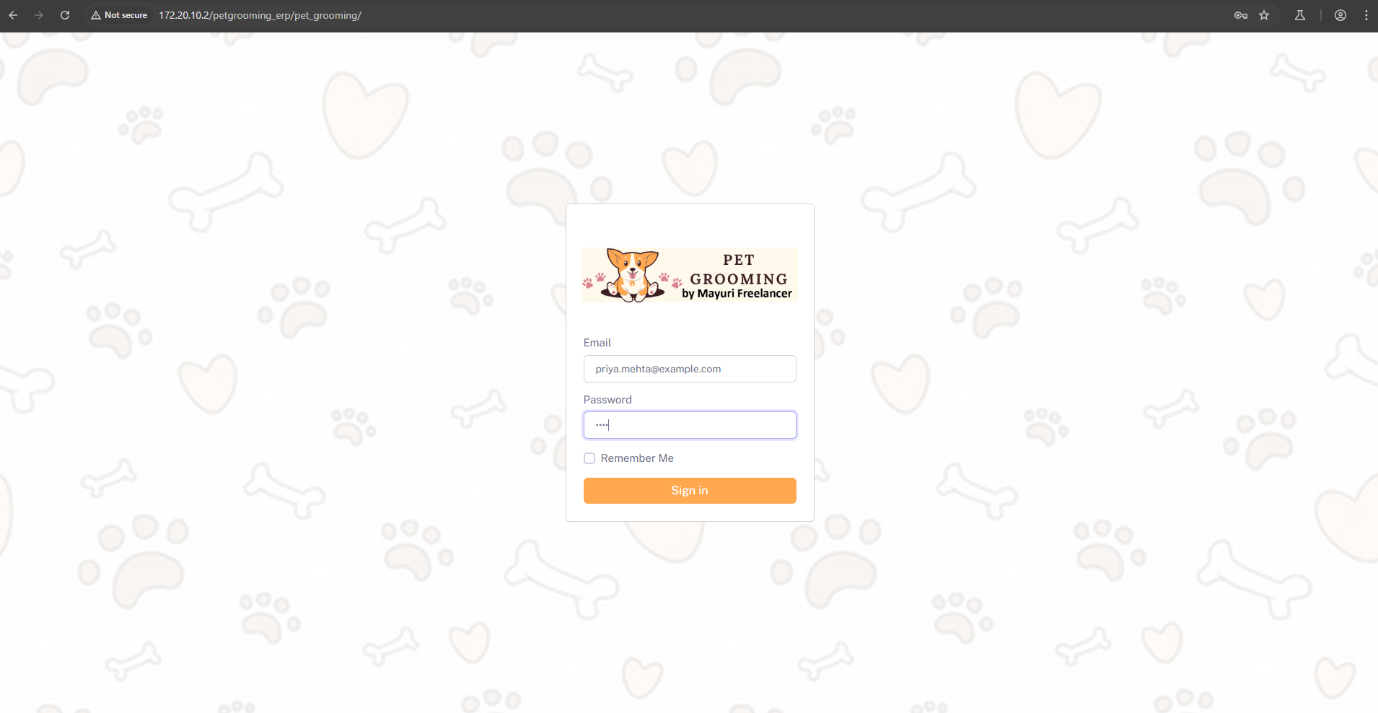
Possible privilege escalation and lateral movement

Database dump and credential theft

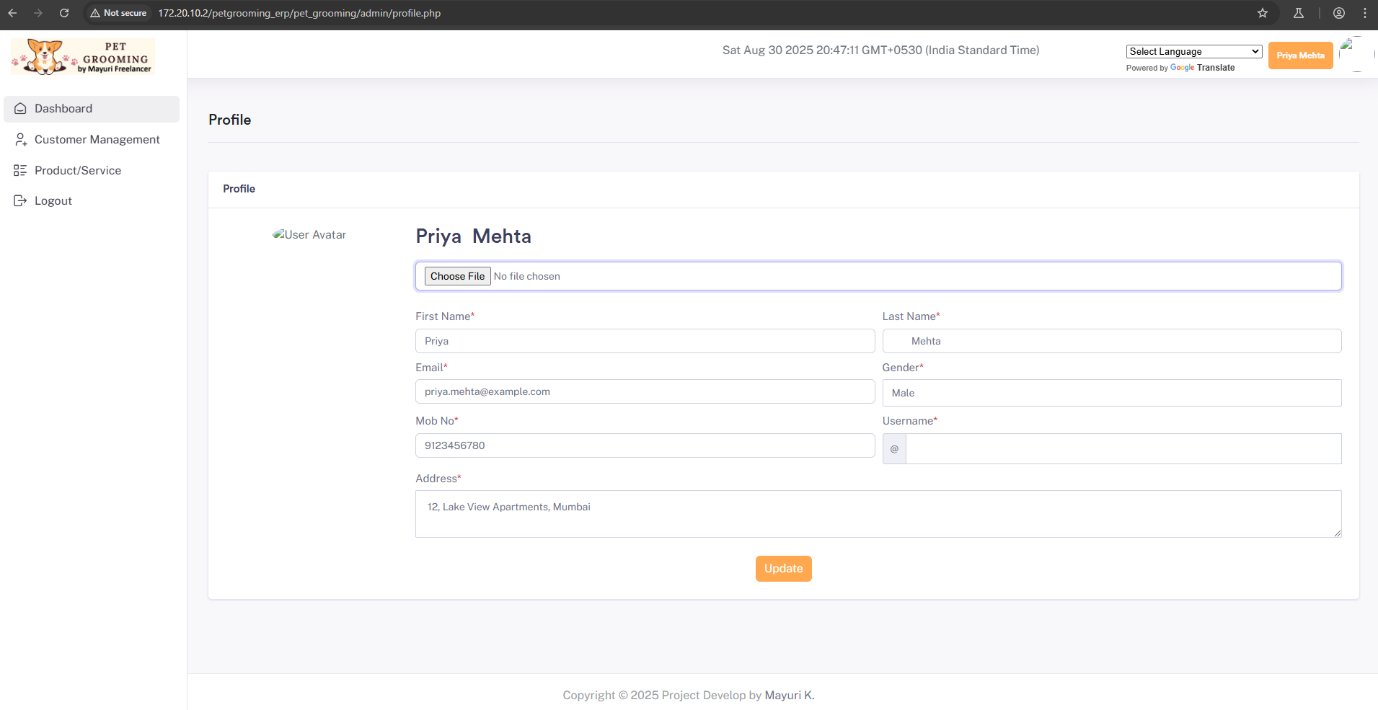
Server takeover

**Proof of Concept (PoC)**

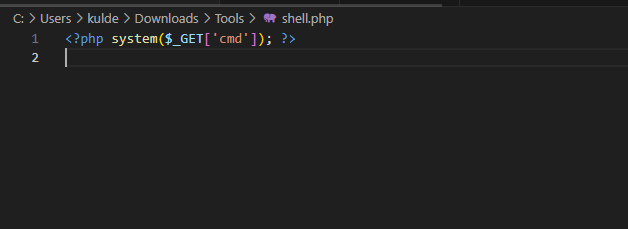
**Step1**:- Login with valid user credentials.



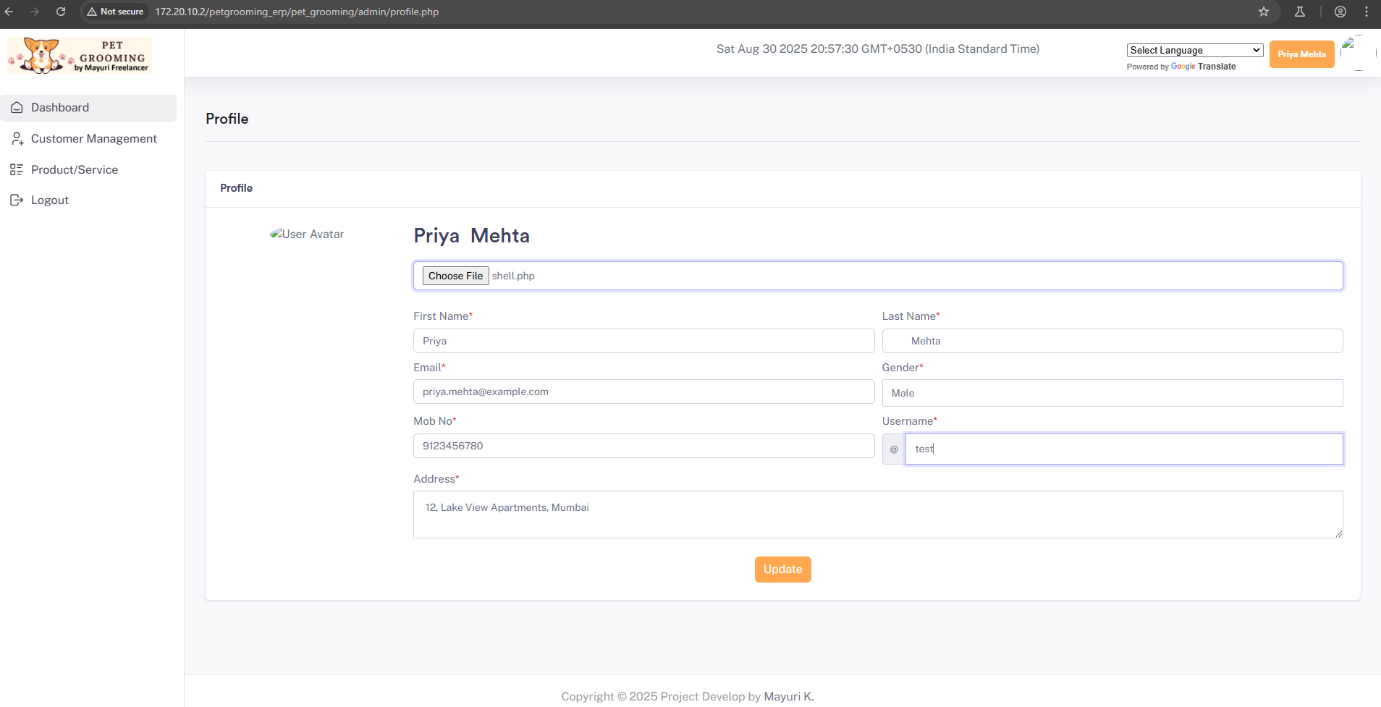
**Step2**:- Navigate to Profile Page section → Upload Image



**Step3**:- Create a single liner php webshell and save it as shell.php

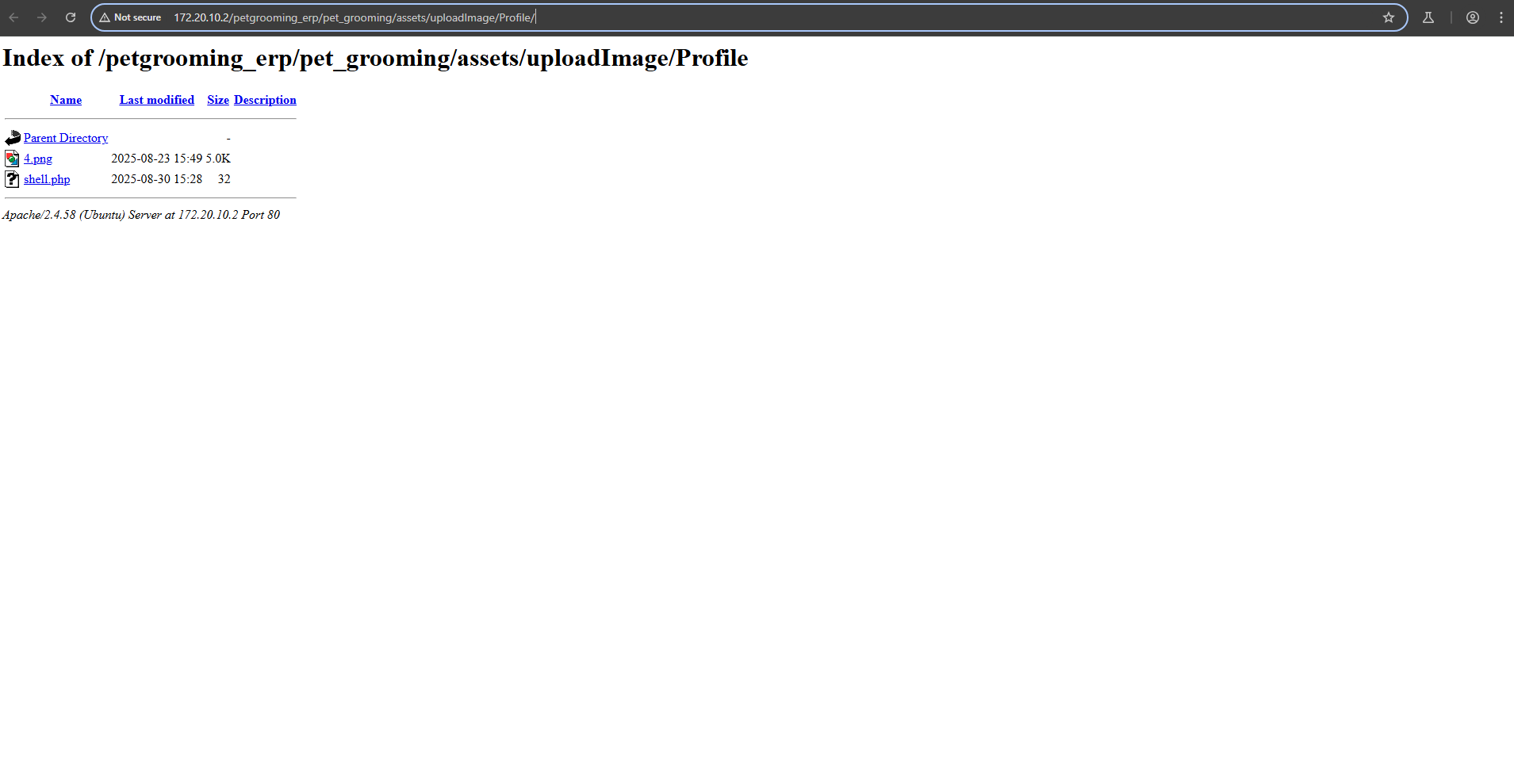


**Step4**:- Upload the wbeshell on the upload image section



fter uploading the **shell.php** file successfully, navigate back to the profile section.  
**(Note:** If the page turns blank after upload, don’t worry — this indicates that shell.php has been successfully uploaded.)

**Step5**:- Right click over the profile image section and open image in new tab



Then you can see the shell.php

**Step6**:- Then click over the shell.php and use ?cmd=id to execute command 

As you can see we can successfully execute the command .

**Remediation**

**To mitigate this vulnerability:**

**Restrict allowed file types**: Ensure the upload functionality only accepts specific safe formats (e.g., .png, .jpg, .jpeg).

**Implement server-side validation**: Do not rely solely on client-side checks. Validate MIME types and file extensions on the server.

**Sanitize and rename uploaded files**: Prevent execution of arbitrary code by renaming files on upload and storing them outside the webroot.