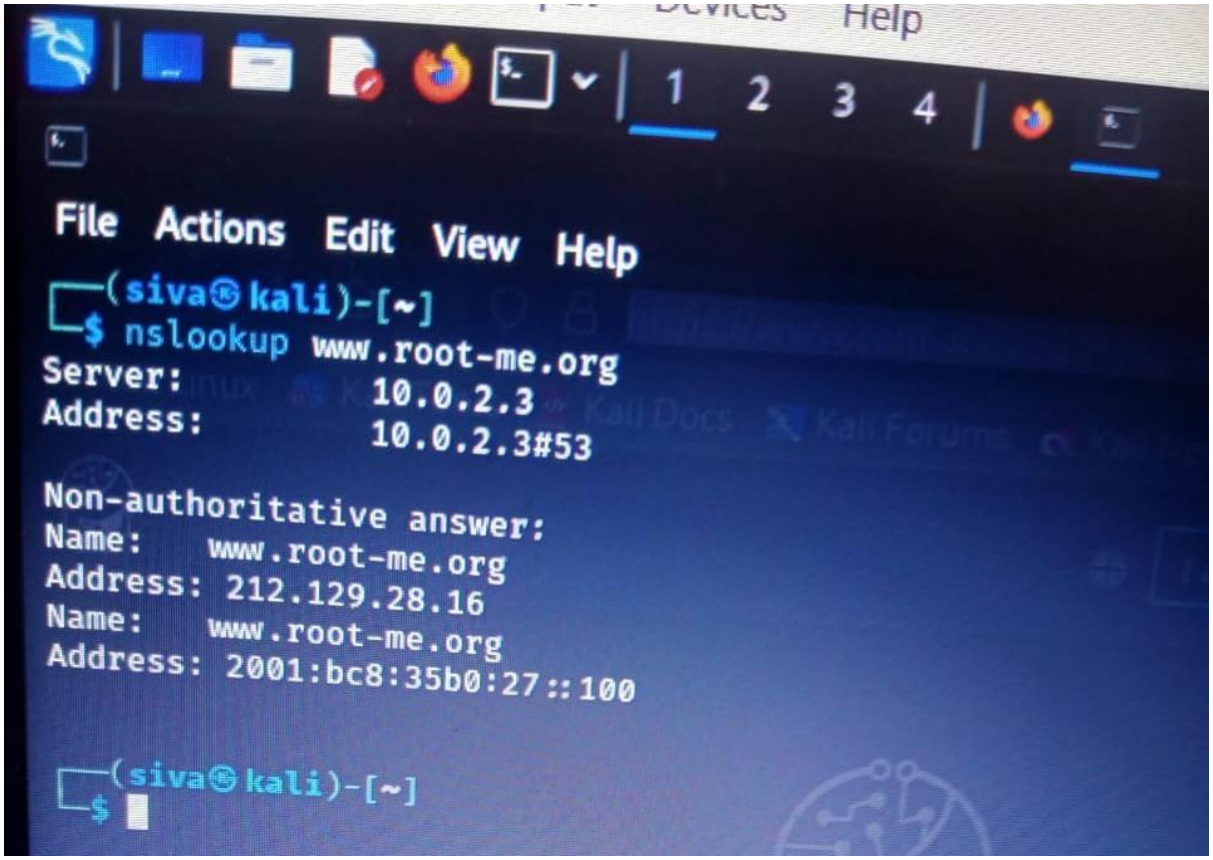
FUNCTIONAL & PERFORMANCE TESTING :

Finding Vulnerabilities for the targeted wedsite :

Targeted Website :root-me

IP Address :212.129.28.16

Software Used : Burp Suite



S.No	Vulnerability Name	CWE-No.
1.	Insecure File Upload	CWE-434
2.	Cross-Site Request Forgery (CSRF)	CWE-352
3.	Cross-Site Scripting (XSS)	CWE-79

To find the vulnerabilities mentioned in your report (Insecure File Upload, Cross-Site Request Forgery (CSRF), and Cross-Site Scripting (XSS)) using Burp Suite, you can follow these general steps. Burp Suite is a powerful web application security testing tool that helps identify and exploit vulnerabilities such as these.

Here's how you can find and identify these vulnerabilities using Burp Suite:

1. Insecure File Upload (CWE-434)

Goal: Detect file upload vulnerabilities that allow malicious files to be uploaded.

- Step 1: Spider the Target
    - Open Burp Suite, go to the Target tab, and set up your proxy (if not already done). Ensure that your browser is configured to route traffic through Burp's proxy.
    - Browse the application to find any file upload functionality (e.g., profile picture upload, document uploads).
    - Use Burp's Spider to crawl through the website and find the file upload forms and endpoints.
- Step 2: Manual Testing of File Upload
    - In Burp Suite, go to the Proxy tab and capture a request for a file upload.
    - Check if the server is performing proper file validation (e.g., checking file extensions, MIME types, or content validation).
    - Use Burp's Intruder tool to test the upload functionality by sending requests with malicious file payloads, such as PHP web shells or other executable file types disguised as images or documents.
- Step 3: Analyzing the Response
    - Check the response from the server. If it allows the upload of malicious files or fails to sanitize the file's metadata (filename, path, etc.), then the system is vulnerable to Insecure File Upload.
    - You can attempt to access the uploaded file if it's saved on the server and check if it gets executed.

Burp Suite Tools to Use:

- Proxy for manual interception of file upload requests.
- Intruder for sending payloads to test file upload handling.
- Repeater to replay requests and observe server responses.

2. Cross-Site Request Forgery (CSRF) (CWE-352)

Goal: Detect CSRF vulnerabilities where malicious actions can be triggered without the user's consent.

- Step 1: Spider the Target
    - In Burp Suite, use the Spider tool to automatically crawl through the website and capture all forms that use GET/POST methods.

    - Specifically look for sensitive actions like changing account details, transferring funds, or submitting forms that modify data.
- Step 2: Check for CSRF Tokens
    - CSRF protection usually relies on tokens that are included in requests (such as in hidden fields within forms or in request headers).
    - Inspect the requests using the Proxy tab to see if CSRF tokens are present. If there's no token or if tokens are not properly validated by the server, the application may be vulnerable to CSRF.
- Step 3: Test for CSRF Vulnerabilities
    - To test for CSRF, use Burp's Intruder tool to manipulate requests that perform state-changing actions (e.g., account settings change, password reset) by crafting malicious requests that don't include the CSRF token.
    - If the server processes the action without validating the CSRF token, it is vulnerable to CSRF.

Burp Suite Tools to Use:

- Spider for crawling and identifying potential CSRF-sensitive endpoints.
- Intruder to automate testing of CSRF vulnerabilities.
- Repeater to manually send requests and observe if CSRF protections are in place.

3. Cross-Site Scripting (XSS) (CWE-79)

Goal: Detect XSS vulnerabilities where attackers can inject malicious scripts into the web pages.

- Step 1: Identify User Input Fields
    - Use Burp Suite's Spider or Scanner (available in Burp Suite Pro) to crawl through the website and identify input fields, such as search bars, contact forms, or comment sections, where user input is reflected back in the browser.
    - Capture requests using the Proxy tab to examine parameters that are vulnerable to XSS.
- Step 2: Test for Reflected XSS
    - In Burp Suite, use Intruder or Repeater to inject common XSS payloads (e.g., <script>alert('XSS')</script>) into the input fields and observe if the script gets executed when the page is returned.

    - Check the response to see if the injected script is reflected back in the browser without proper sanitization or escaping.
- Step 3: Test for Stored XSS
    - For stored XSS, look for places where user input is stored (e.g., profile details, blog comments).
    - Inject payloads into input fields and see if the injected JavaScript is stored and later executed when others view the page.
- Step 4: Check for DOM-based XSS
    - Inspect the JavaScript on the page using Burp's DOM Invader (if you have Burp Suite Pro) or other manual techniques to see if user input is passed directly into the DOM without proper validation.

Burp Suite Tools to Use:

- Spider for automatic crawling and identifying input points.
- Scanner (in Burp Suite Pro) for automated detection of XSS vulnerabilities.
- Intruder to test input fields with various XSS payloads.
- Repeater to manually test and observe script execution.
- DOM Invader (in Burp Suite Pro) for identifying DOM-based XSS.

Summary of Burp Suite Features for Testing:

- Proxy: Intercepts and analyzes HTTP requests and responses to identify vulnerabilities.
- Spider: Crawls through the application to find forms and user input fields.
- Intruder: Sends automated attack payloads to test for file upload, CSRF, and XSS vulnerabilities.
- Repeater: Manually edits and resends HTTP requests to test for vulnerabilities.
- Scanner (Pro version): Automates the process of identifying vulnerabilities like XSS, CSRF, and insecure file upload.
- DOM Invader (Pro version): Helps identify DOM-based XSS vulnerabilities.

By following these steps and using the Burp Suite tools, you can identify and assess the vulnerabilities (Insecure File Upload, CSRF, XSS) present in a web application.