https://labs.hackxpert.com/CSRF

CSRF Labs: Course Instructions with Solutions

Welcome to the CSRF Labs course! In this enhanced guide, we provide solutions to each lab alongside step-by-step instructions. Use these insights to understand the logic behind each CSRF vulnerability and how to exploit or secure against it.

CSRF 1: Basics of CSRF Token Validation

- **Objective**: Understand the importance of CSRF tokens and how to bypass basic implementation flaws.
- **Solution**: This form should have a CSRF token implemented. Check the form's source code to locate the token and identify if it is being validated properly.
- **Steps**:
    1. Locate the CSRF token in the form fields or request headers.
    2. Craft a malicious HTML page that includes the token in the request.
    3. Test whether the server validates the token correctly.
    - **Insight**: If the token is not validated server-side, it can be easily bypassed.

CSRF 2: Length-Only Validation

- **Objective**: Exploit weak validation of the CSRF token.
- **Solution**: This form only checks if the length of the CSRF token is correct. You can bypass it by providing a token of the expected length, even if the value is invalid.
- **Steps**:
    1. Inspect the token to determine its expected length.
    2. Craft a malicious payload that includes a placeholder token of the same length.
    3. Submit your exploit and verify the action.
    - **Pro Tip**: Tools like Burp Suite can help craft requests with arbitrary token values.

CSRF 3: UserID as Token

- **Objective**: Exploit a CSRF implementation that uses predictable values, such as the userID, as the token.
- **Solution**: The CSRF token is the userID. You can craft your exploit by including the victim's userID as the token value.
- **Steps**:
    1. Identify the victim's userID (this could be through observing the URL, cookies, or another source).
    2. Include the userID as the token value in your malicious payload.
    3. Test the exploit to confirm that the action is performed on behalf of the victim.
    - **Challenge**: Finding the userID requires some reconnaissance on the application.

CSRF 4: Empty Token Check

- **Objective**: Exploit a system that only checks whether the token is not empty.
- **Solution**: This system only ensures that a token value is present but does not validate its correctness. You can bypass it by including any non-empty value as the token.
- **Steps**:
    1. Include a placeholder token (e.g., "abc123") in your exploit payload.
    2. Submit the crafted payload to the system.
    3. Verify that the request is accepted, despite the token being invalid.
    - **Tip**: This is a significant design flaw that highlights poor server-side validation.

CSRF 5: Partial Token Validation

- **Objective**: Exploit a system that only checks if the submitted value is part of the actual CSRF token.
- **Solution**: The system validates the token by checking if the submitted value is part of the real token. Submitting partial values like "1" or "a" can randomly match valid tokens.
- **Steps**:
    1. Test the system with simple token values (e.g., "1", "a").
    2. Observe which values result in successful requests.
    3. Refine your attack to increase the chances of guessing a valid token.
    - **Note**: This is a critical flaw that exposes the system to brute-force attacks.

General Notes

- The solutions file contains specific examples and further explanations for each vulnerability.
- As you progress through the labs, you'll notice how the complexity of CSRF tokens and validation mechanisms impacts the security of the application.
- Use these insights to learn both how to exploit vulnerabilities and how to design systems resistant to CSRF attacks.