Session Hijacking Attack Using XSS and Cookie Editor: A Step-by-Step Guide

Introduction

This document provides a detailed guide on how a session hijacking attack was performed on `testphp.vulnweb. using` a Cross-Site Scripting (XSS) vulnerability to capture the session cookie, followed by manually injecting the cookie using a cookie editor to hijack the session. The documentation explains the process, the associated vulnerabilities, and the necessary steps to mitigate such risks.

Prerequisites

- **Cookie Editor Extension** installed in your browser (e.g., "EditThisCookie" for Chrome or "Cookie-Editor" for Firefox).
- **Firefox or Chrome Browser**.
- **Basic knowledge** of XSS, HTTP requests, and cookie handling.

Step 1: Identifying the XSS Vulnerability

Explanation: Cross-site scripting (XSS) is a common vulnerability that occurs when an application allows users to inject malicious scripts into web pages viewed by other users. These scripts can capture sensitive information, such as session cookies.

Instructions:

1. Navigate to `testphp.vulnweb.com` in your browser.
2. Locate the search option or any input field reflecting user input on the page.
3. To test for XSS, enter the following script into the search field:
   <script>alert(document.cookie)</script>
4. Submit the search form.

Outcome:

- If the application is vulnerable, you should see an alert box displaying the current session's cookie (e.g., `PHPSESSID=abc123xyz;`).

Step 2: Capturing the Session Cookie

Explanation: The script `<script>alert(document.cookie)</script>` is a simple XSS payload. When executed, it triggers an alert box that displays the current session cookie. This script exploits the fact that the application does not properly sanitize user inputs, allowing the script to run in the browser.

Instructions:

1. After submitting the XSS payload, an alert box should appear displaying the session cookie.
2. Copy the session cookie from the alert box for use in the hijacking attempt.

Vulnerability Explanation:

- **XSS Vulnerability**: The application's failure to properly sanitize user input allows malicious scripts to be executed. This can lead to various attacks, such as session hijacking, where the attacker steals the session cookie and gains unauthorized access to the user's session.

Step 3: Open a New Browser Instance or Incognito Mode

Explanation: To perform the hijacking attack, it's essential to use a separate browser session where you will inject the stolen session cookie.

Instructions:

1. Open a new incognito/private window in your browser or a different browser to ensure no existing session cookies interfere with the attack.
2. Navigate to `testphp.vulnweb.com` but do not log in.

Step 4: Manually Add the Captured Cookie Using the Cookie Editor

Explanation: A cookie editor allows you to manually modify, add, or delete cookies for a particular site. By manually adding the stolen session cookie, you can take over the user's session who originally owned that cookie.

Instructions:

1. Open the cookie editor extension from your browser's toolbar.
2. Click the + icon to add a new cookie.
3. Manually enter the following details:
   - **Name**: `login`
   - **Value**: Paste the stolen session ID.
   - **Domain**: Set this to `testphp.vulnweb.com`.

    - **Path**: Set this to `/`.
4. Save the changes in the cookie editor.

Step 5: Reload the Page to Hijack the Session

Explanation: After manually adding the session cookie, reloading the page will send the modified cookie to the server, allowing you to assume the identity of the user whose session you've hijacked.

Instructions:

1. Reload the `testphp.vulnweb.com` page in the browser where you manually added the cookie.
2. If the session hijacking was successful, you should now be logged in as the user associated with the session ID you used.
3. Verify access by navigating through the site and performing actions that would typically require authentication.

Step 6: Verify and Document the Hijacked Session

Explanation: Documenting the process and verifying the successful hijack is crucial for understanding the implications of such vulnerabilities and how to prevent them.

Instructions:

1. Navigate through `testphp.vulnweb.com` to see if you have access to user-specific areas like account settings or personal information.
2. Take screenshots of the steps, including the XSS alert, the cookie editor before and after modification, and the results after the page reload.
3. Document any actions you were able to perform that confirm the session hijacking was successful.

Step 7: Mitigation Recommendations

Explanation: Understanding how to exploit a vulnerability is the first step toward securing systems against such attacks. The final step involves recommending best practices to prevent session hijacking and XSS attacks.

Instructions:

1. **Sanitize User Inputs**: Ensure that all user inputs are properly sanitized and encoded before being rendered on the page to prevent XSS.

2. **Use Secure Cookies**: Set the `HttpOnly` and `Secure` flags on session cookies to prevent client-side scripts from accessing them and to ensure they are only transmitted over HTTPS.
3. **Implement Content Security Policy (CSP)**: A CSP can help mitigate the impact of XSS by restricting the sources from which scripts can be loaded.
4. **Session Expiration and Regeneration**: Ensure that sessions expire after a period of inactivity and that session IDs are regenerated upon login to reduce the risk of session fixation and hijacking.

Conclusion

This documentation outlines the process of performing a session hijacking attack using an XSS vulnerability to capture the session cookie, followed by manually injecting the cookie using a cookie editor. By understanding the method and its implications, organizations can take steps to secure their web applications against such attacks. Always conduct these activities in ethical and legal environments to ensure responsible security practices.