Target: Damn Vulnerable Web Application (DVWA)

1. Findings Log

Test ID	Vulnerability	Severity	Target URL
DVWA-SQLi-001	SQL Injection (Blind)	Critical	`http://localhost:42001/vulnerabilities/sqli/`
DVWA-XSS-002	Cross-Site Scripting (Reflected)	High	`http://localhost:42001/vulnerabilities/xss_r/`

Vulnerability evidence is provided in the screenshots referenced below.

---

2. Testing Checklist

- **Test for SQL injection (sqlmap)**: **Completed.** The `id` parameter on the SQL Injection page was found to be vulnerable to automated exploitation. The `sqlmap` tool successfully enumerated databases and dumped the dvwa user table credentials. (See: `10_sqlmap_Database_Enum.png`, `11_sqlmap_User_Credentials_Dump.png`)
- **Check for XSS (manual payloads)**: **Completed.** The `name` parameter on the Reflected XSS page was found to be vulnerable. The payload `<script>alert('XSS')</script>` was successfully injected and executed using Burp Suite to intercept and manipulate the HTTP request. (See: `12_Burp_Intercepted_Request.png`, `13_Burp_Modified_XSS_Request.png`, `14_XSS_Alert_Triggered.png`)

---

3. Testing Summary

Assessment of the DVWA application identified critical security flaws, including a SQL Injection vulnerability allowing full database disclosure and a Reflected Cross-Site Scripting flaw. These findings, part of the OWASP Top 10, demonstrate a lack of input validation and output encoding. Immediate remediation through the use of parameterized queries and context-aware output encoding is critically advised.

Screenshot Evidence References:

1. **`10_sqlmap_Database_Enum.png`:** `sqlmap` successfully identifying the backend DBMS and enumerating available databases.
2. **`11_sqlmap_User_Credentials_Dump.png`:** `sqlmap` dumping and cracking the hashes from the `users` table, revealing weak passwords.
3. **`12_Burp_Intercepted_Request.png`:** Burp Suite Proxy intercepting the initial GET request to the Reflected XSS page.
4. **`13_Burp_Modified_XSS_Request.png`:** The intercepted request after modification, showing the XSS payload inserted into the `name` parameter.
5. **`14_XSS_Alert_Triggered.png`:** The victim's browser displaying the JavaScript alert pop-up, proving successful execution of the injected script.