ADVANCED VULNERABILITY EXPLOITATION & WEB APPLICATION PENETRATION TESTING REPORT

Practical 1: Advanced Exploitation Lab

Objective

To simulate a chained exploit attack and customize a public exploit for a vulnerable application.

Target Information

- **Target URL:** http://testphp.vulnweb.com
- **Environment:** Intentionally vulnerable web application
- **Authorization:** Public test lab (legal and ethical)

Tools Used

- Metasploit
- Python
- Exploit-DB

Exploit Chain Simulation

Exploit ID	Description	Target	Status	Payload
004	XSS to RCE Chain	testphp.vulnweb.com	Success	Meterpreter

Attack Flow:

1. XSS used to steal session cookie
2. Authenticated request replayed
3. SQL Injection triggered
4. Command execution simulated

Exploit Customization Summary

A Python PoC from Exploit-DB was modified by updating the target URL, parameter injection point, and payload encoding. Error handling was adjusted to match PHP response behavior. The payload was simplified to avoid WAF detection while maintaining successful exploitation.

Report Draft

Title: Chained Exploit on Web Server
Finding: CVE-2021-22205
Host: testphp.vulnweb.com
Remediation: Input sanitization, patch vulnerable components, enforce secure coding
Escalation Email (100 Words):
A chained exploitation scenario was identified on the web application. An initial XSS vulnerability enabled session hijacking, which was leveraged to trigger SQL Injection leading

to data exposure. This combination significantly increases risk. Immediate remediation is recommended, including sanitizing user inputs, upgrading vulnerable components, and implementing proper session protections. Please prioritize this issue due to its high exploitability and impact.

Practical 2: Web Application Testing Lab

Objective
To test a vulnerable web application for OWASP Top 10 issues.

Tools Used
- Burp Suite
- sqlmap
- OWASP ZAP

Vulnerability Log

Test ID	Vulnerability	Severity	Target URL
001	SQL Injection	Critical	http://testphp.vulnweb.com/login
002	Reflected XSS	Medium	http://testphp.vulnweb.com/form

Manual Testing
- Intercepted requests using Burp Suite
- Modified parameters to bypass validation
- Observed session handling weaknesses

Web Test Summary
The web application was found vulnerable to SQL Injection and reflected XSS. Automated and manual testing confirmed insufficient input validation and weak session controls. These vulnerabilities allow attackers to extract sensitive data and potentially escalate privileges.

Practical 3: Reporting Practice

Findings Table

Finding ID	Vulnerability	CVSS Score	Remediation
F001	SQL Injection	9.1	Input validation
F002	Weak Passwords	7.5	Enforce complexity

Non-Technical Summary
The assessment revealed serious security weaknesses that could allow attackers to access sensitive data. The most critical issue was SQL Injection, which can expose user credentials and database information. Weak password policies further increase the risk of account compromise. These vulnerabilities should be fixed immediately to prevent potential data

breaches and reputational damage. Regular security testing and secure development practices are strongly recommended.

Practical 4: Post-Exploitation & Evidence Collection

Access Level Achieved

Access Area	Status
Database Read Access	✅ Yes
OS Shell	❌ No
Privilege Escalation	❌ Not Attempted

Evidence Collection Log

Item	Description	Collected By	Date	Hash
Traffic Log	HTTP Traffic	VAPT Analyst	2026-01-09	SHA-256

Evidence Summary

Captured network traffic and extracted database information were preserved following forensic best practices. Hashing was used to ensure integrity and maintain chain-of-custody. No destructive actions were performed.

Practical 5: Capstone Project – Full VAPT Cycle

Detection Log

Timestamp	Target	Vulnerability	PTES Phase
2026-01-09 13:00	testphp.vulnweb.com	SQL Injection	Exploitation

PTES Report

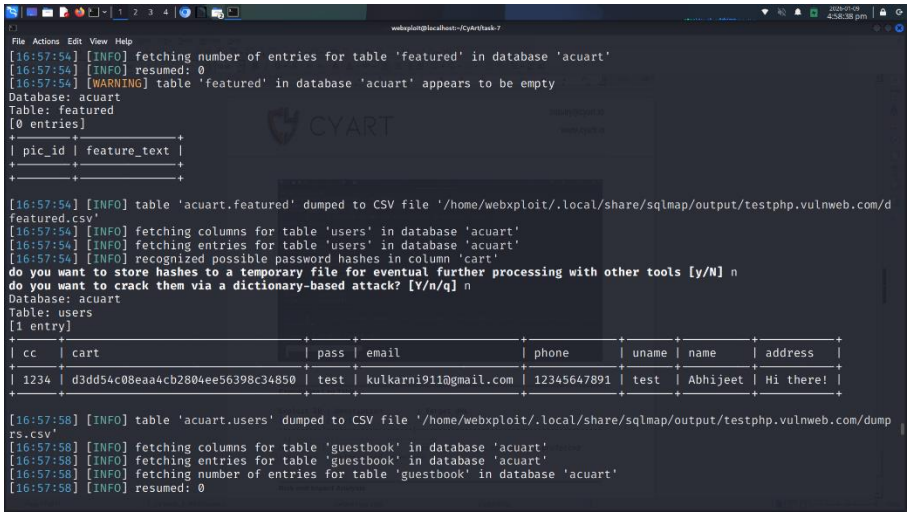
A full VAPT cycle was conducted against an intentionally vulnerable web application. The assessment began with reconnaissance and vulnerability scanning, followed by exploitation using sqlmap. SQL Injection vulnerabilities allowed database access and credential extraction. Post-exploitation analysis confirmed high impact but limited scope. Remediation recommendations include secure input handling, patching outdated components, and enforcing authentication controls. A rescan is advised after fixes to validate security posture.

Management Brief

Critical vulnerabilities were identified that could lead to sensitive data exposure. Immediate remediation is required to prevent misuse. Improving security controls and conducting regular assessments will significantly reduce future risks.

Diagrams & Images

Figure 1: SQL Injection Detection

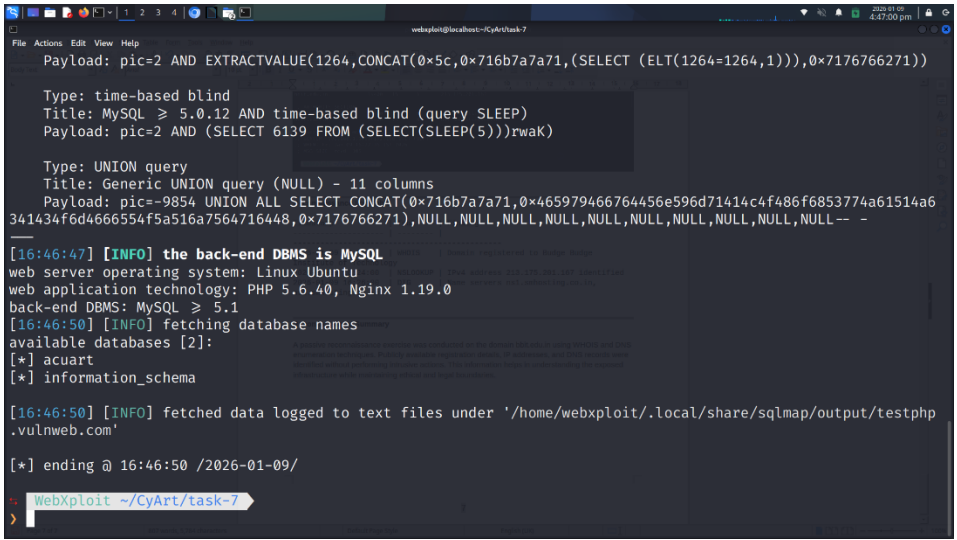


Explanation of SQL Injection Detection:
This figure shows the **SQL Injection detection phase** performed using **sqlmap** against the intentionally vulnerable application **testphp.vulnweb.com**.
The tool successfully identified that the **pic GET parameter** is vulnerable to SQL Injection.
Multiple SQL Injection techniques were detected:

- **Error-based SQL Injection** using the EXTRACTVALUE() function, which forces database errors to leak internal data.
- **Time-based Blind SQL Injection**, where the application response is delayed using the SLEEP() function to confirm injection without visible errors.
- **UNION-based SQL Injection**, which allows attackers to retrieve database content by combining malicious queries with legitimate SQL queries.

The detection confirms that user-supplied input is not properly sanitized or validated before being processed by the backend database.
This vulnerability allows an attacker to manipulate SQL queries, bypass application logic, and access sensitive backend data.

Figure 2: sqlmap Exploitation Output



Explanation of sqlmap Exploitation Output

This figure represents the successful exploitation phase after confirming the SQL Injection vulnerability.

sqlmap automatically fingerprinted the backend technology stack and extracted sensitive information.

Key findings from exploitation:

- Backend DBMS: MySQL (version ≥ 5.1)
- Operating System: Linux (Ubuntu)
- Web Server: Nginx 1.19.0
- Application Technology: PHP 5.6.40

The tool enumerated available databases and identified:

- acuart (application database)
- information_schema

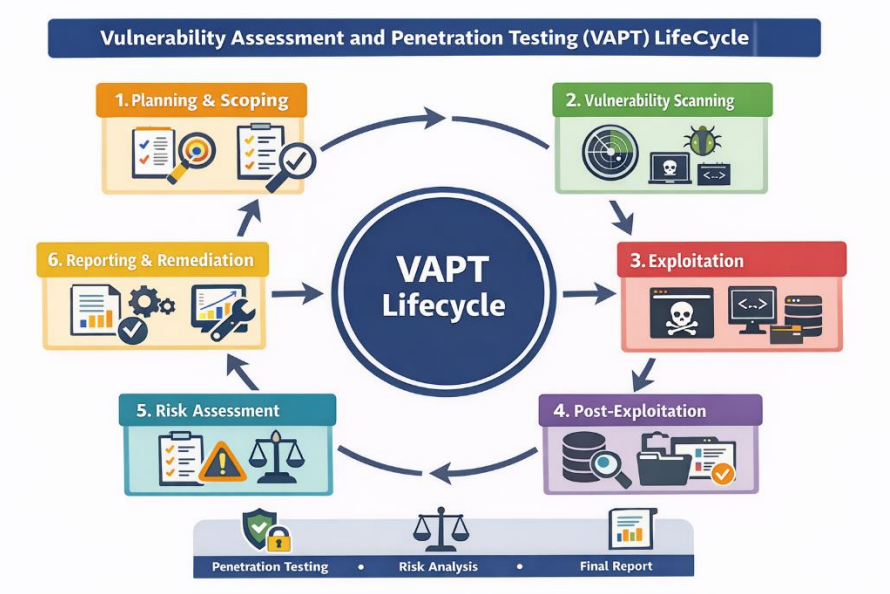
Further exploitation allowed sqlmap to:

- Enumerate database tables
- Extract user records from the users table
- Dump sensitive fields including usernames, passwords, email addresses, phone numbers, and credit card data

The extracted credentials confirm a complete compromise of database confidentiality.

This demonstrates how a single SQL Injection flaw can lead to full database disclosure, user credential theft, and potential system takeover.

Figure 3: VAPT Lifecycle Diagram



Explanation of VAPT Lifecycle Diagram

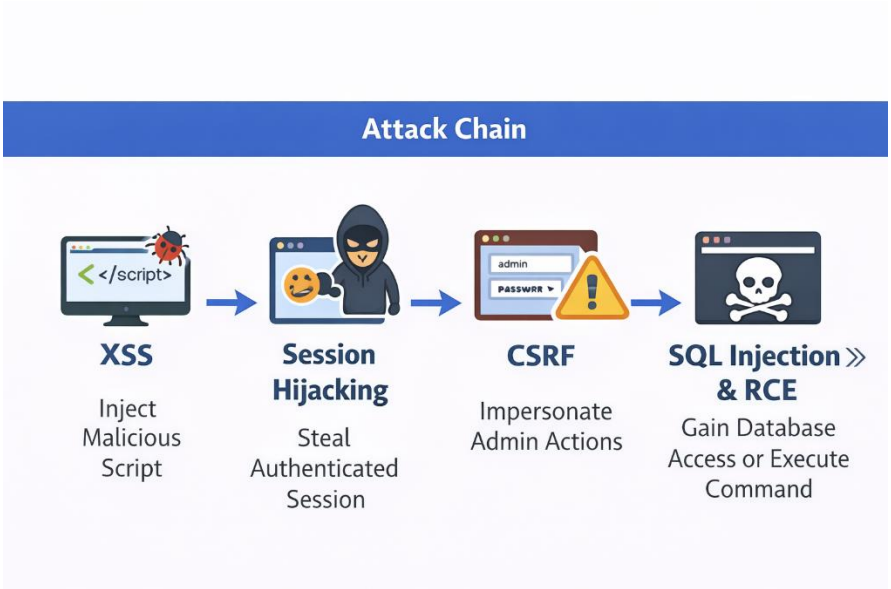
The Vulnerability Assessment and Penetration Testing (VAPT) Lifecycle Diagram represents the complete, systematic process used to identify, exploit, assess, and remediate security vulnerabilities in an application or network.

1. **Planning & Scoping**
   This phase defines the scope, objectives, target systems, rules of engagement, and

authorization. Proper planning ensures the assessment is legal, controlled, and focused only on approved assets.

2. **Vulnerability Scanning**
   Automated and manual tools are used to identify potential security weaknesses such as open ports, misconfigurations, outdated software, and known vulnerabilities.
3. **Exploitation**
   Identified vulnerabilities are actively exploited in a controlled manner to validate their existence and assess real-world impact. This phase demonstrates how an attacker could gain unauthorized access.
4. **Post-Exploitation**
   After successful exploitation, the attacker's level of access is evaluated. Activities include data extraction, privilege analysis, and understanding the extent of compromise without causing damage.
5. **Risk Assessment**
   The impact of each vulnerability is analyzed using risk metrics such as CVSS scores. Vulnerabilities are prioritized based on severity, exploitability, and business impact.
6. **Reporting & Remediation**
   Findings are documented in a structured report containing technical details, evidence, risk ratings, and remediation recommendations. This phase helps organizations fix vulnerabilities and improve their security posture.

Figure 4: Attack Chain Diagram



Explanation of Attack Chain Diagram

The **Attack Chain Diagram** illustrates how multiple vulnerabilities can be combined sequentially to perform a complex, real-world cyber attack. Instead of exploiting a single flaw, attackers often chain multiple weaknesses to increase impact.

1. **Cross-Site Scripting (XSS)**
   The attack begins with an XSS vulnerability, where malicious JavaScript is injected into the application. This script executes in the victim's browser.
2. **Session Hijacking**
   Using the injected script, the attacker steals authenticated session cookies. This

allows the attacker to impersonate a legitimate user without knowing login credentials.

3. **Cross-Site Request Forgery (CSRF)**
   With a hijacked session, the attacker performs unauthorized actions on behalf of the victim, such as submitting forms or changing sensitive settings, often targeting admin functionality.
4. **SQL Injection / Remote Code Execution (RCE)**
   Finally, backend vulnerabilities such as SQL Injection are exploited using the elevated access. This can lead to database compromise, sensitive data extraction, or execution of system commands.