Vulnerability Assessment & Penetration Testing on a Web Application

Jayant Singh

TABLE OF CONTENTS

COVER PAGE

TITLE

Vulnerability Assessment & Penetration Testing on a Web Application

NAME

Jayant Singh

COURSE

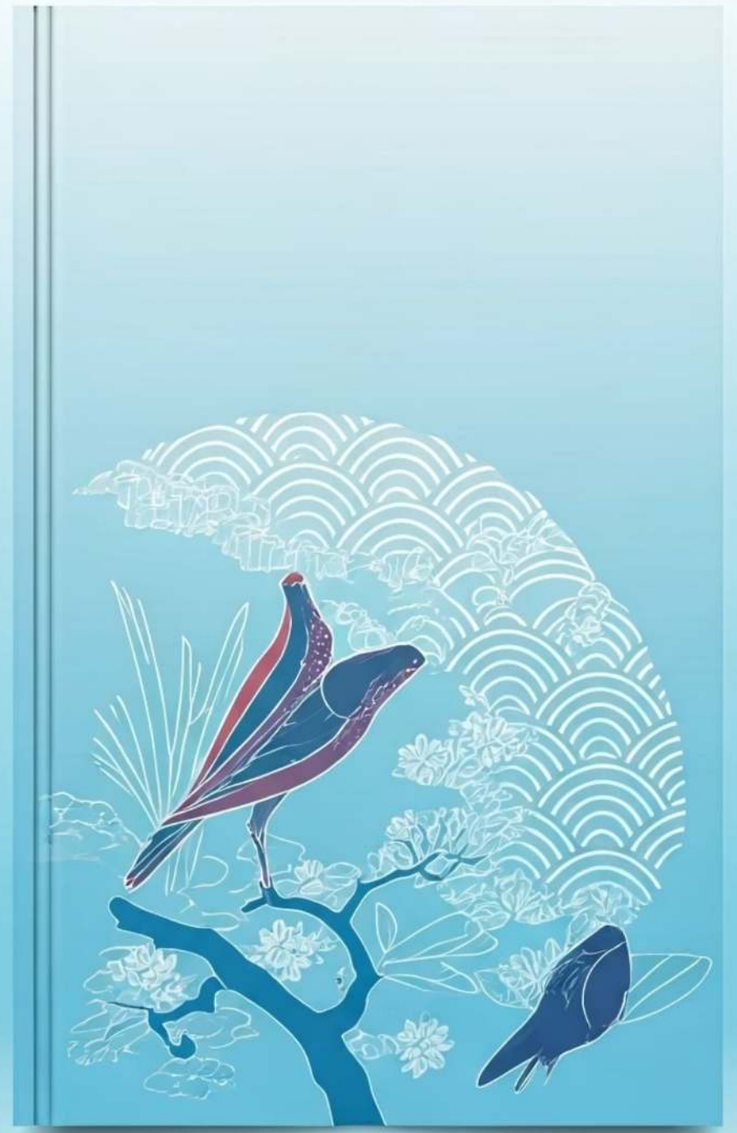
Electronics & Communication Engineering

INSTITUTION

JSS , Noida , Gautam Buddha Nagar, UP

SUPERVISOR'S NAME

MR. Nikhil Pandey



ABSTRACT

This project report details a Vulnerability Assessment and Penetration Testing (VAPT) performed on a sample web application. The goal was to identify security flaws using tools such as Nmap, Burp Suite, OWASP ZAP, and SQLMap.

Multiple vulnerabilities were detected, analyzed, and documented. The project simulates a practical attack scenario while adhering to ethical hacking standards, providing insights into common web application weaknesses.

INTRODUCTION TO VAPT

In an era where digital landscapes define our lives, Every line of code, every configuration, every connected device holds a potential doorway for malicious actors. But what if you could see these hidden vulnerabilities before they become catastrophic breaches? What if you could simulate the very attacks designed to compromise your data, and then build an impenetrable defense?

This is the essence of **Vulnerability Assessment and Penetration Testing (VAPT)**.

WHAT IS VAPT? A DUAL-LENS APPROACH TO CYBERSECURITY

VAPT is not merely a security check; it's a strategic, proactive defense mechanism that combines two powerful methodologies to provide an unparalleled view into your system's resilience:

1. **Vulnerability Assessment (VA): The Broad Scan**

What it is: A systematic process of identifying known security weaknesses and flaws within your applications and infrastructure.

How it works: Think of it as a comprehensive health check. Automated tools meticulously scan your systems, comparing configurations and software versions against vast databases of known vulnerabilities.

The Outcome: A broad, categorized list of potential security issues, highlighting where your defenses *might* be weak. It tells you *what* weaknesses exist.

2. **Penetration Testing (PT): The Targeted Attack Simulation**

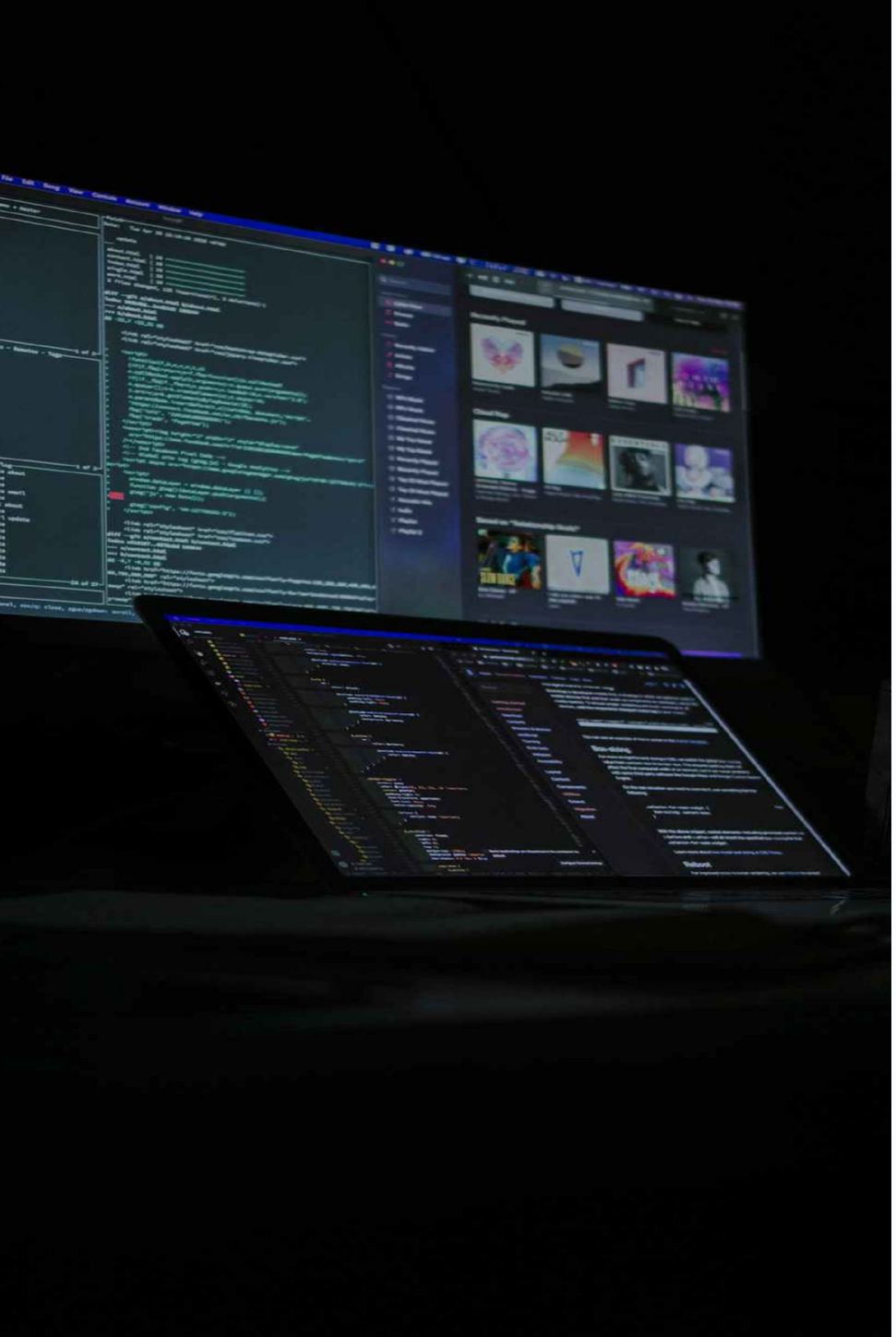
What it is: A hands-on, adversarial simulation where security experts actively attempt to exploit the identified weaknesses.

How it works: This is where the "ethical hacker" mindset comes into play. Combining automated tools with deep manual techniques, our experts simulate real-world attack scenarios to determine if a vulnerability is truly exploitable and what its real-world impact would be.

The Outcome: Concrete proof of exploitability, demonstrating the actual risk and potential business impact. It tells you *how* a weakness can be exploited and *what damage* it could cause

THIS REPORT

Delve into the findings of such an assessment, revealing the vulnerabilities discovered and outlining the precise steps to transform potential weaknesses into unyielding strengths. Prepare to see your security landscape through the eyes of an adversary, and then, to build a defense that truly stands guard.



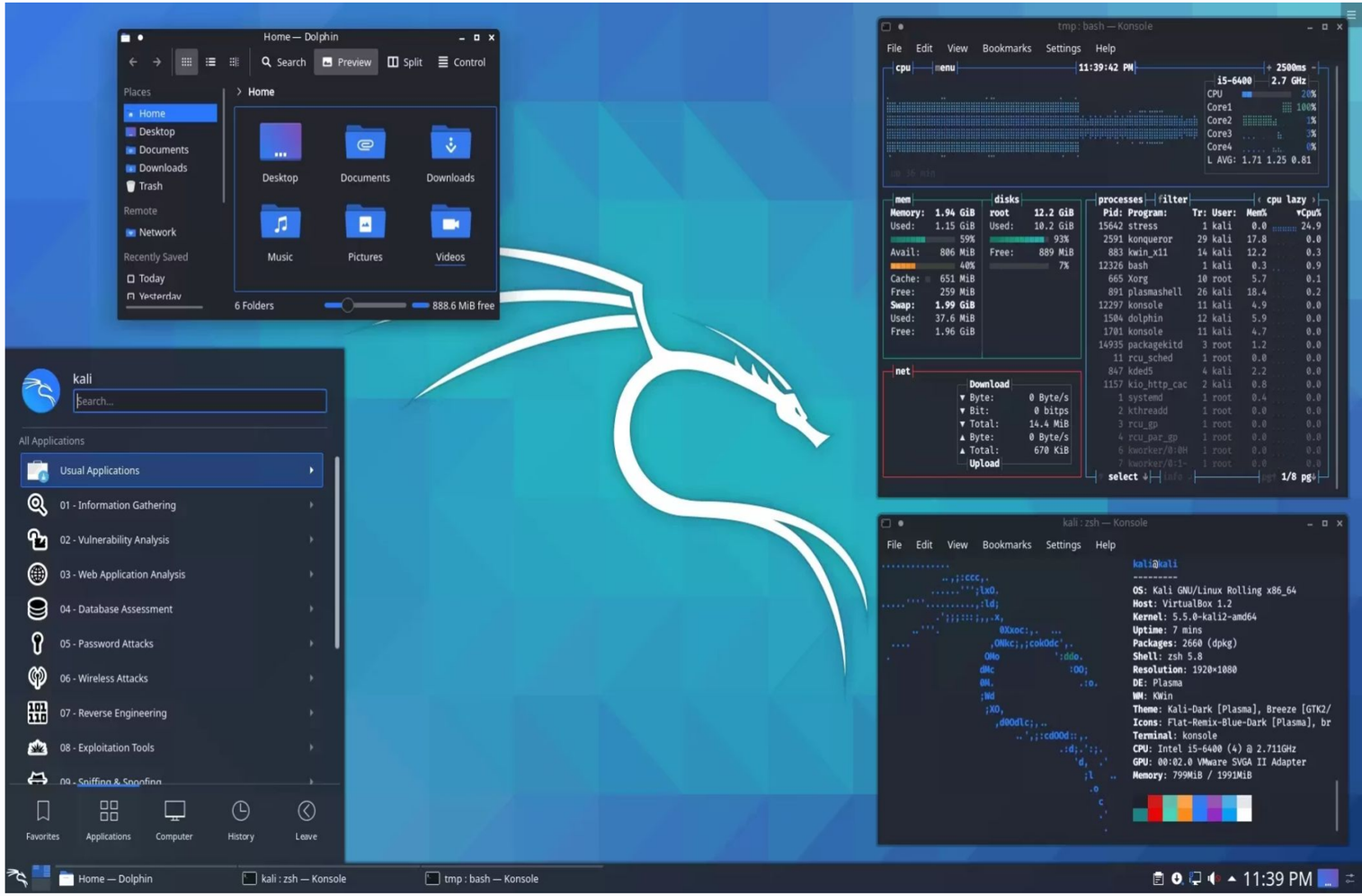
TOOLS AND SETUP

OPERATING SYSTEM

Kali Linux / Parrot OS (Virtual Machine)

WEB APPLICATION FOR TESTING: OWASP DVWA (Damn Vulnerable Web App) or BWAPP

TOOLS USED : BURPE SUITE, NMAP, SQL MAP, Mozilla Firefox Plugins



METHODOLOGY: STEP-BY-STEP VAPT

SETUP ENVIRONMENT

Set up the vulnerable environment (DVWA/BWAPP on XAMPP or Docker).

CONFIGURE BROWSER

Configure the browser to work with Burp/ZAP for traffic interception.

SCAN WITH NMAP

Scan the target web application with Nmap for open ports and OS detection.

INTERCEPT TRAFFIC

Intercept web traffic using Burp Suite or OWASP ZAP to identify potential vulnerabilities.

EXPLOIT VULNERABILITIES

Exploit identified vulnerabilities, such as SQL Injection, using SQLMap.

DOCUMENT RESULTS

Document all findings, including detected vulnerabilities and exploitation details.

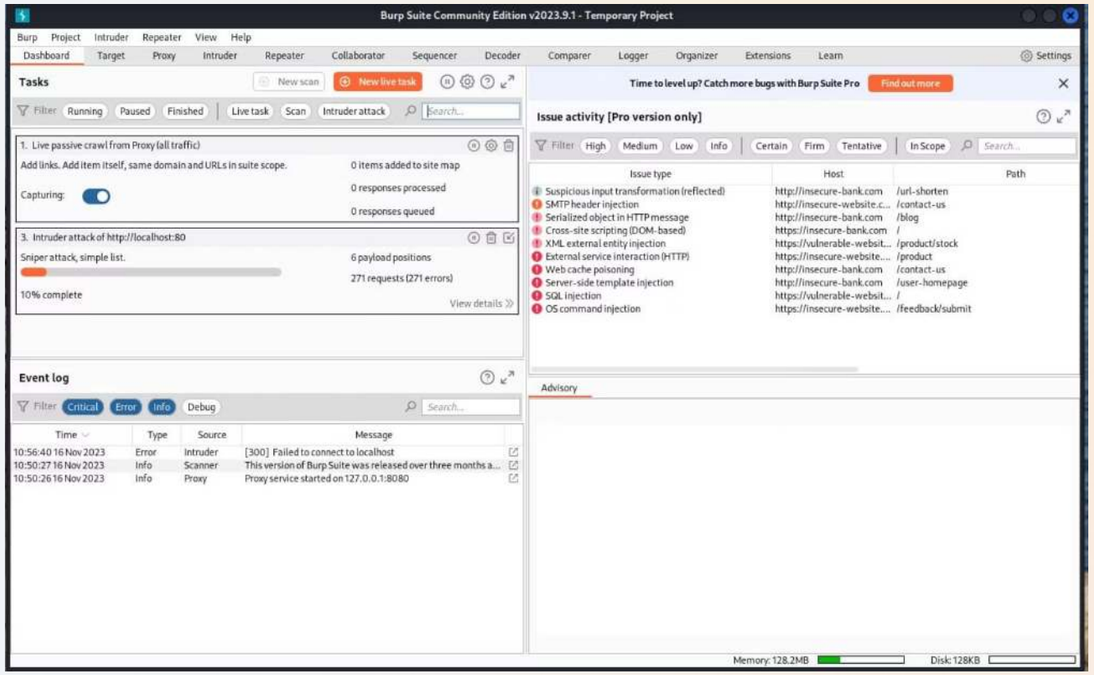
BURP SUITE / ZAP RESULTS: INTERCEPTION & ANALYSIS

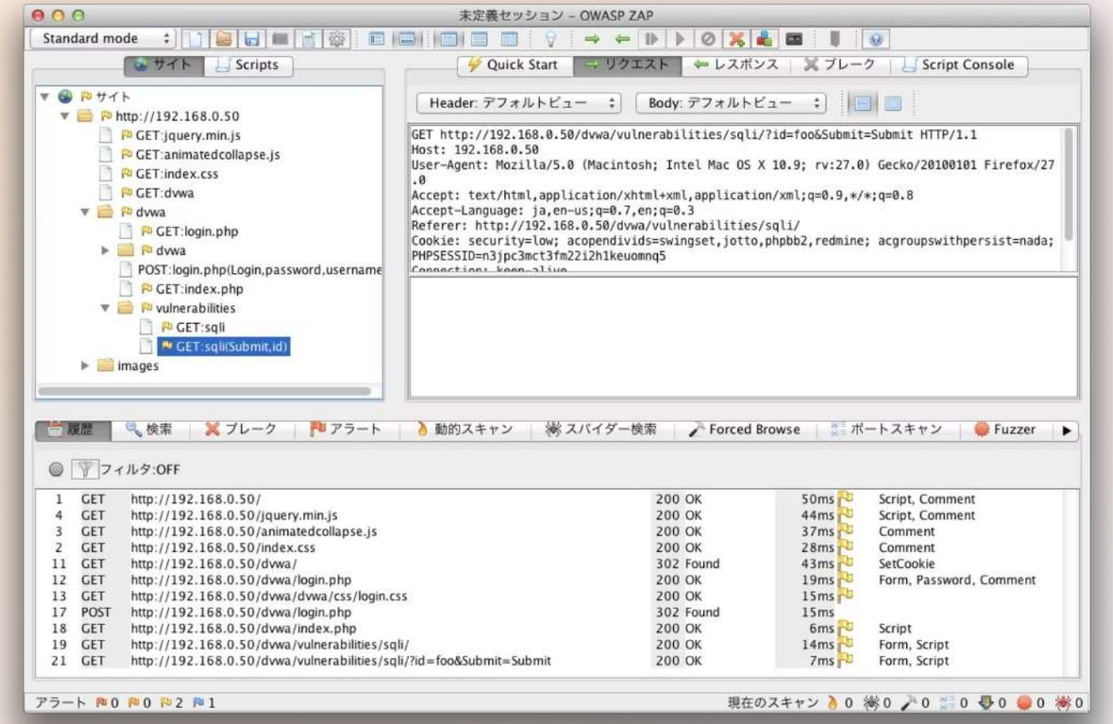
Captured Login Request: Successfully intercepted authentication requests.

Identified Input Fields for Injection: Pinpointed areas susceptible to injection attacks.

Analyzed Responses and Vulnerabilities: Examined server responses to uncover potential security flaws.

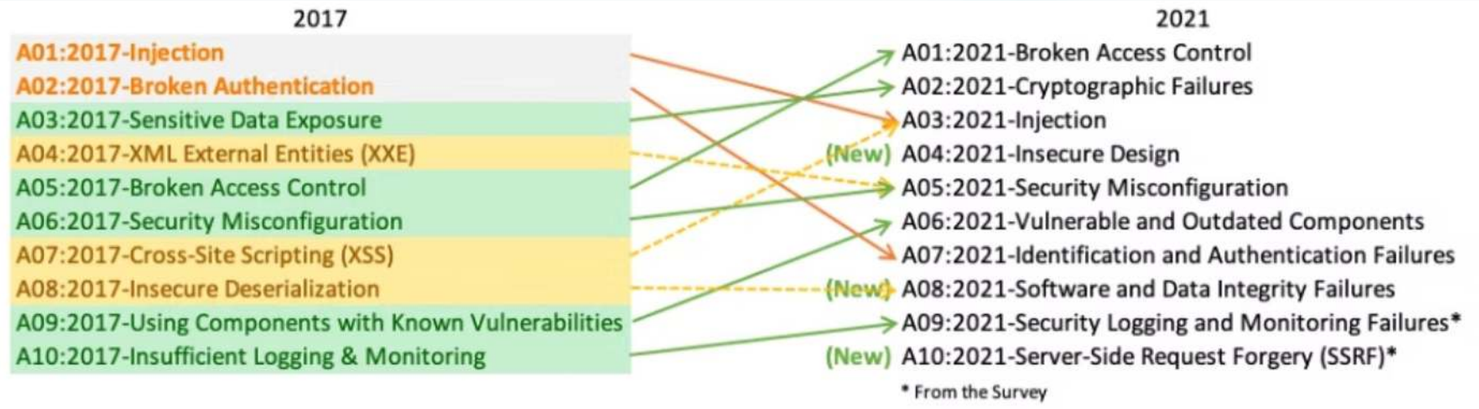
Burp Suite and OWASP ZAP were instrumental in analyzing web traffic, identifying sensitive data, and discovering vulnerabilities through active and passive scanning.





OWASP TOP TEN: CRITICAL WEB APPLICATION SECURITY RISKS

The **OWASP Top Ten** is a standard awareness document for developers and web application security professionals. It represents a broad consensus about the most critical security risks to web applications



OWASP Top 10 Indicators

	System	Vulnerabilities	Exploitable
A1 – Injection	2	10	60 %
A2 – Broken Authentication and Session Management	5	34	21 %
A3 – Cross-Site Scripting (XSS)	2	6	60 %
A4 – Insecure Direct Object Reference	5	49	45 %
A5 – Security Misconfiguration	7	54	44 %
A6 – Sensitive Data Exposure	5	30	6 %
A7 – Missing Functio n Level Access Control	4	13	92 %
A8 – Cross-Site Request Forgery (CSRF)	0	0	%

BROKEN AUTHENTICATION

Flawed authentication or session management allows attackers to assume user identities (e.g., weak passwords, exposed session IDs).

INJECTION

Commands or queries injected into input fields, leading to data theft or system compromise (e.g., SQL Injection).

SECURITY MISCONFIGURATION

Default settings, unnecessary features, overly permissive roles, or verbose error messages create easy entry points for attackers

CRYPTOGRAPHIC FAILURES

outdated, or mis-configured encryption and hardcoded secrets expose sensitive data at rest or in transit to unauthorized disclosure

INSECURE DESIGN

Fundamental flaws in architecture and business logic introduce risks that cannot be fixed by simple code patches or configuration tweaks

VULNERABILITY OUTDATED COMPONENTS

Reliance on third-party libraries and frameworks with known flaws leaves applications at risk until components are patched or replaced

IDENTIFICATION AND AUTHENTICATION FAILURES

Weak or missing controls around login, session management, and credential validation enable account takeover and impersonation.

SOFTWARE AND DATA INTEGRITY FAILURE

Insecure CI/CD pipelines, unverified dependencies, or unsigned updates allow attackers to inject malicious code into software builds.

SECURITY LOGGING AND MONITORING FAILURES

Poor or missing audit trails and alerts prevent timely detection and investigation of breaches, prolonging an attacker’s dwell time.

SERVER-SIDE REQUEST FORGERY (SSRF)

Unsanitized URL inputs let attackers abuse server-side functionality to make unauthorized requests to internal or external

NMAP RESULTS: INITIAL RECONNAISSANCE

COMMAND USED:

```
nmap -A <target IP>
```

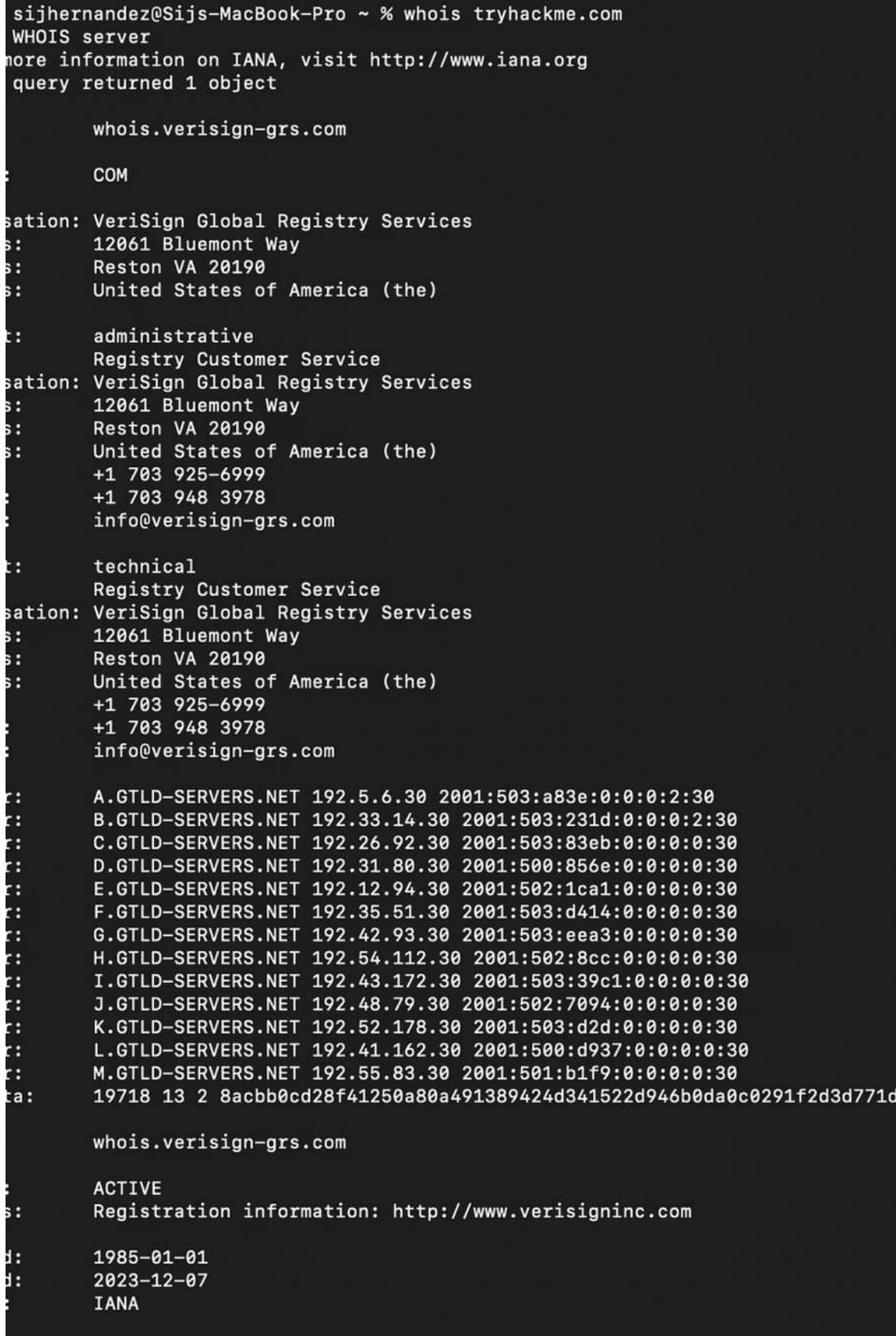
SUMMARY OF FINDINGS:

Port 80 open – Apache (Web Server)

Port 22 open – SSH (Secure Shell)

OS Detected: Linux-based operating system

Nmap provided crucial initial insights into the target's network services and operating system, laying the groundwork for further penetration testing.



SQLMAP RESULTS: DATABASE EXPLOITATION

COMMAND EXAMPLE:

```
sqlmap -u "http://<target-ip-or-domain>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=<your-session-id>" --dbs
```

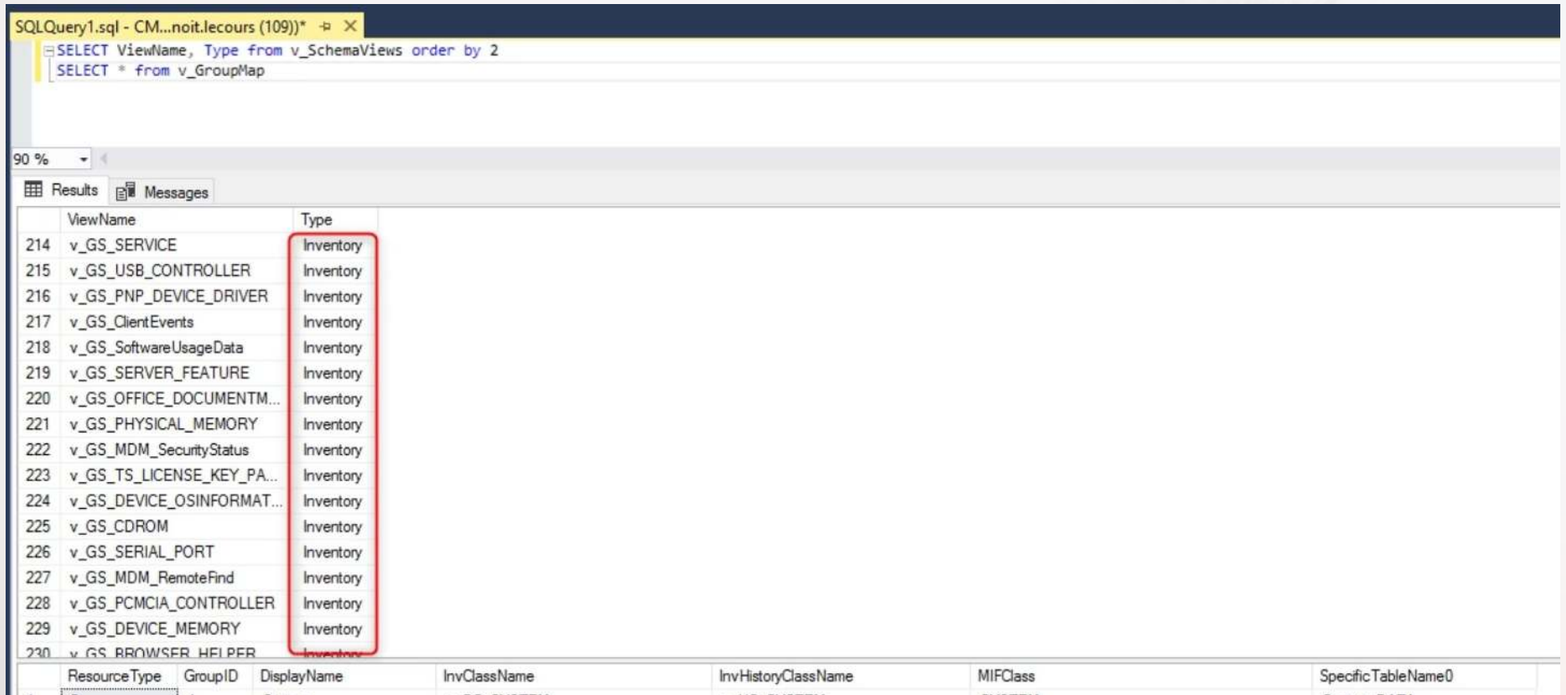
Key Findings :

Database: dvwa

Tables: users, guestbook

Exploited: SQL Injection (Error-based)

SQLMap successfully exploited SQL Injection vulnerabilities, allowing the extraction of database names and table structures, demonstrating the critical impact of such flaws.



VAPT FOR REGULATORY COMPLIANCE

Vulnerability Assessment and Penetration Testing (VAPT) is essential for organizations to meet strict industry and government regulations.

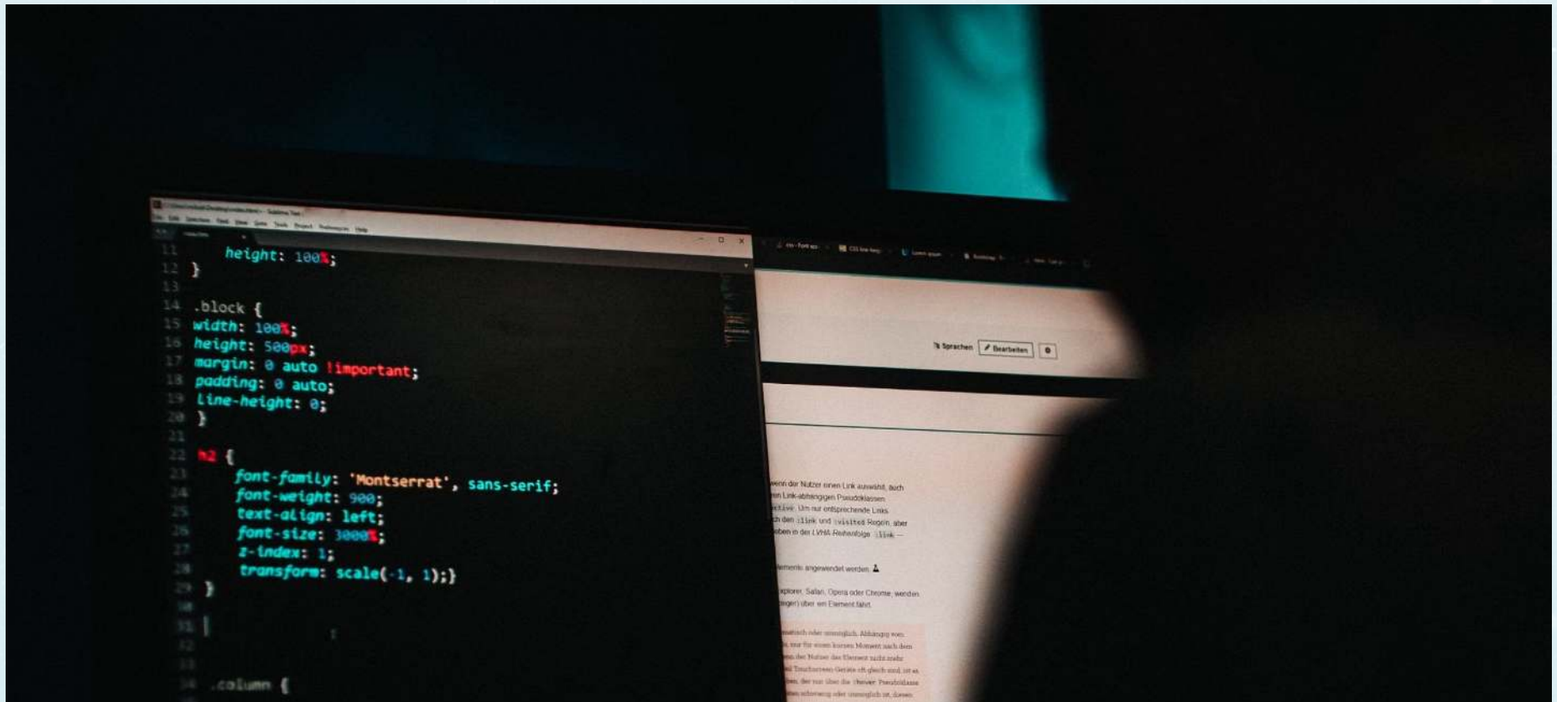
PCL DDS

Identifies flaws in cardholder data environments, ensuring compliance with payment security standards.

GDPR & HIPAA

Supports data breach preventing & reporting mandates.

ISO/IEC 27018	PCI DSS
VAPT verifies secure data processing & leak preventions	Identifies flaws in cardholder environments



Public Cloud Service

Businesses Seeking

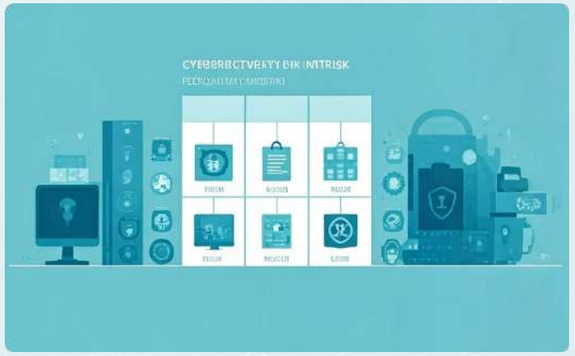
REPORTING BEST PRACTICES

CLEAR & CONCISE

Ensure the report is easy to understand, avoiding overly technical jargon where possible for key stakeholders.

ACTIONABLE RECOMMENDATIONS

Provide specific, implementable steps for remediation, not just identifying vulnerabilities.



PRIORITIZATION & RISK ASSESSMENT

Rank vulnerabilities by severity and potential impact to help prioritize remediation efforts effectively.

EVIDENCE & SCREENSHOTS

Include supporting evidence like screenshots or log excerpts to validate findings and aid in replication.

PROFESSIONAL FORMATTING & REVIEW

Proofread for clarity, correct terminology and ensure compliance with any organizational templates/ standards.

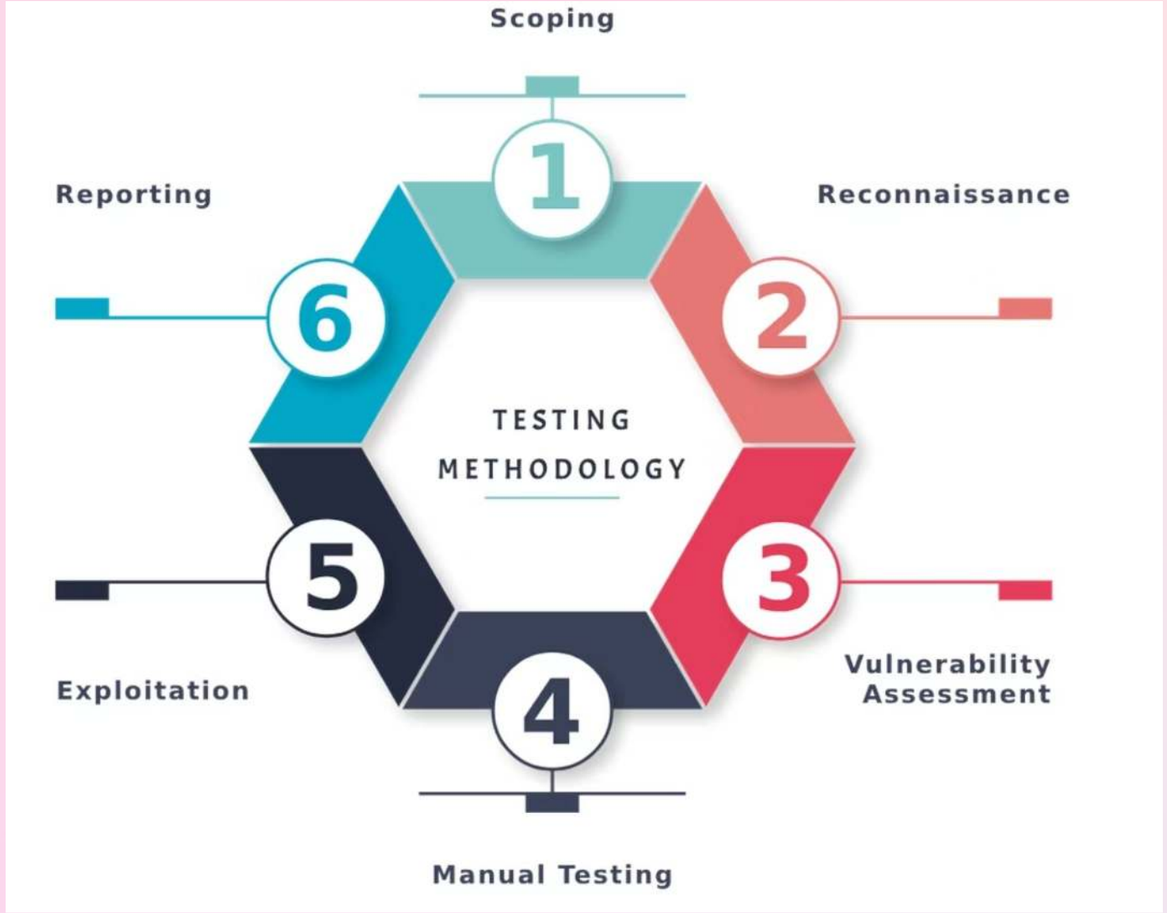
STRUCTURED EXECUTIVE SUMMARY

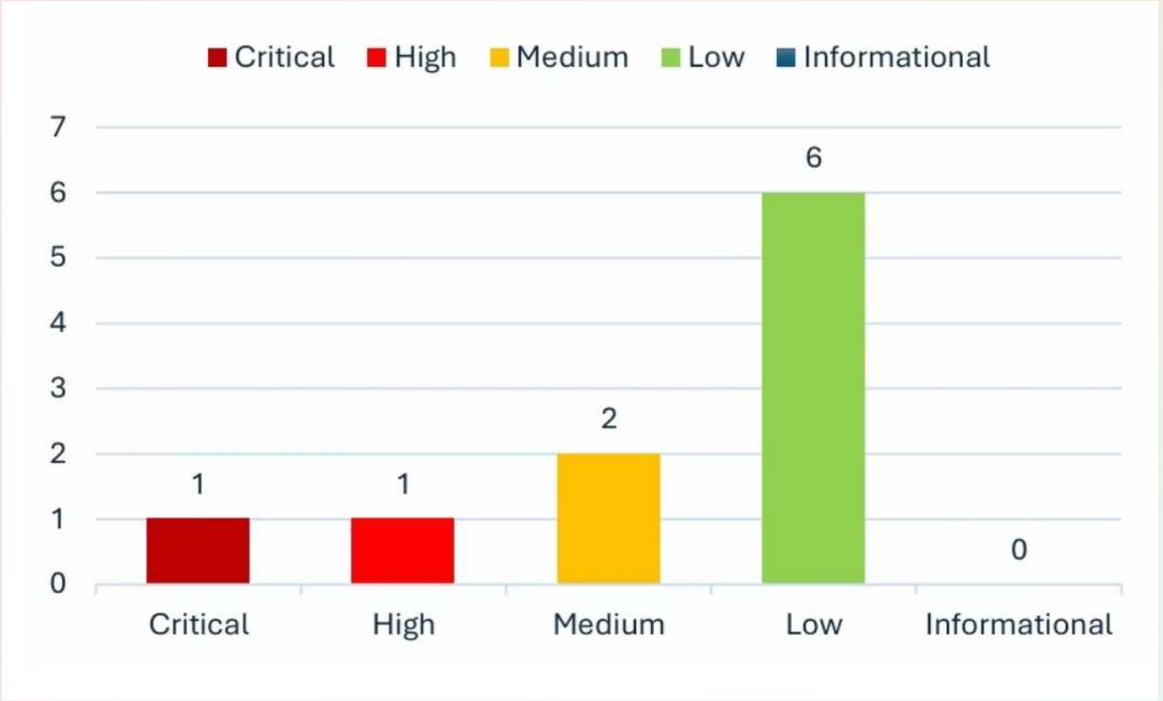
overview of objectives, scope, key findings, and high-level recommendations so stakeholders grasp the essence without diving into technical details

VULNERABILITY TABLE

This table contain the key vulnerabilities identified during the VAPT, their severity, the affected component and reference of the OWASP

ID	**Vulnerability Description**	Severity	Affected Component	OWASP/CWE Reference
1	**Admin account with weak credentials**	High	WordPress	OWASP A2 / CWE-521
2	**Multiple weak SoftwareName admin passwords**	High	SoftwareName	OWASP A2 / CWE-521
3	**MySQL password found in cleartext**	Medium	Server file system	OWASP A6 / CWE-285
4	**Clickjacking vulnerability**	Medium	Web server configuration	OWASP A5 / CWE-693
5	**Sensitive logs exposed via HTTP**	Medium	Server file system	OWASP A5 / CWE-200
6	**Backup files world-readable**	Medium	Server file system	OWASP A5 / CWE-200
7	**Username enumeration via URL**	Medium	WordPress	OWASP A5 / CWE-200
8	**Outdated WordPress version**	Low	WordPress	OWASP A9 / CWE-200
9	**Outdated SoftwareNam version**	Low	SoftwareName	OWASP A9 / CWE-200
10	**Missing HSTS header**	Low	HTTPS server	OWASP A5
11	**World writable directories**	Low	Web application directory	OWASP A5 / CWE-693
12	**Server discloses software version in headers**	Low	Web server configuration	OWASP A5 / CWE-200
13	**Vulnerable jQuery libraries**	Low	Client-side scripting	OWASP A9 / CWE-200





Severity of Vulnerabilities

S. No.	Name	Severity	Risk Score	Status
1.	CSRF Leads to account takeover	Critical	9.1	Unresolved
2.	CSRF on the claim Create/ Update page	High	8.8	Unresolved
3.	Weak Password Policy (Password in plaintext)	Medium	6.5	Unresolved
4.	No Rate Limit	Medium	5.0	Unresolved
5.	Token doesn't implement properly (Token Misconfigured)	Low	3.1	Unresolved

COMPARISON TABLE: VULNERABILITY TESTING VS PENETRATION TESTING

Aspect	Vulnerability Testing	Penetration Testing
Goal	Find weaknesses	Exploit weaknesses
Method	Automated scanning	Manual + automated techniques
Risk	Low	Medium to high (controlled test)
Output	List of potential issues	Proof of exploitability
Tools	Nessus, OpenVAS	Metasploit, Burp Suite
Skill Required	Moderate	High (experience in offensive security

CHALLENGES FACED DURING VAPT

1. **Limited Visibility in Blackbox Testing**
   - With no prior knowledge of internal architecture, discovering vulnerabilities was akin to searching blindfolded.
   - Many services and endpoints were not directly exposed or documented, making enumeration difficult.
2. **Risk of Account Lockouts During Brute Force Attempts**
   - Aggressive credential guessing risked triggering lockout mechanisms or alerting security monitors.
   - Balancing thoroughness with discretion was critical to not disrupt live services.
3. **Difficulty Identifying Legacy and Unpatched Components**
   - Several systems were outdated but obscured version information deliberately through minimal banners or obfuscation.
   - Lack of proper documentation or CMS clues hindered passive reconnaissance.
4. **Overwhelming Volume of False Positives from Scanners**
   - Automated tools generated redundant or inaccurate alerts, cluttering analysis.
   - Distinguishing critical vulnerabilities from informational noise consumed valuable testing time.
5. **Security Header Misconfigurations Hidden by Dynamic Routing**
   - Some pages applied HTTP security headers, while others did not—making consistent detection challenging.
   - Dynamic content delivery systems occasionally bypassed core configuration checks.
6. **File System Permissions Inconsistencies**
   - Identifying and verifying world-writable directories without full access rights required creative command chaining and limited shell execution.

SOLUTIONS

1. **Extended Enumeration with Hybrid Recon Tools**

Combined automated directory bruteforcing (`DirBuster`, `WPScan`) with OSINT techniques to uncover hidden login panels, outdated plugins, and sensitive files.

2. **Smart Brute Force Tactics**
   - Used throttled brute-force logic and curated password lists to minimize risk.

Integrated `wpscan` with time-based attempts, respecting lockout thresholds.

3. **Fingerprinting via Static Artifacts**

Analyzed exposed files like `readme.html`, JS/CSS libraries, and headers to deduce software versions.

   - Mapped discovered data to known CVEs for exploit correlation.

4. **Manual Validation of High-Impact Findings**
   - Each critical alert from Nessus or OpenVAS was manually tested using Burp Suite and command injection probes.
   - This eliminated false positives and refined reporting accuracy.
5. **Universal Header Policy Creation**

Drafted Nginx configuration templates to enforce headers like `X-Frame-Options`, `Strict-Transport-Security`, and `Content-Security-Policy` uniformly.

6. **Safe Shell Usage for Permission Auditing**

Executed non-intrusive shell scripts to scan for `777` directories and assess exposure without modifying server integrity.

CONCLUSION

The vulnerability assessment and penetration testing (VAPT) engagement conducted on the target web application provided a clear view of its security posture across authentication, authorization, data handling, and infrastructure configurations. Leveraging automated tools—Nmap for network enumeration, Burp Suite and OWASP ZAP for web analysis—and manual exploitation techniques, we validated potential attack vectors aligned with the OWASP Top Ten and relevant CWE categories.

Key findings from this engagement include:

- Critical vulnerabilities such as weak administrative credentials and broken access controls (IDOR) that allow unauthorized data retrieval and privilege escalation.
- Medium-severity issues involving server misconfigurations, publicly accessible backup files, and storage of sensitive credentials in cleartext.
- Insufficient logging and monitoring mechanisms, impeding timely detection of intrusion attempts and forensic analysis.

WHAT I LEARNED

- Combining automated tools with manual testing is essential to uncover complex, business-logic vulnerabilities.
- Efficient filter tuning in Burp Suite and OWASP ZAP drastically reduces false positives without missing true positives.
- Translating technical findings into stakeholder-friendly language ensures buy-in and faster remediation.
- Thorough documentation of replication steps and evidence collection builds credibility and reproducibility.
- Iterative retesting after each remediation cycle catches residual misconfigurations that initial scans may overlook.

RECOMMENDED NEXT STEPS

- Enforce strong password policies and implement multi-factor authentication for all sensitive interfaces.
- Patch and upgrade all software dependencies and frameworks to their latest secure versions.
- Harden system configurations by disabling unnecessary services, applying least-privilege permissions, and enforcing secure HTTP headers.
- Integrate automated security testing and secure code reviews into the CI/CD pipeline to catch regressions early.
- Enhance logging, monitoring, and alerting via centralized SIEM solutions and regular audit reviews.

Addressing these recommendations will bolster the organization’s defense-in-depth strategy, align with OWASP best practices, and meet compliance requirements. Continued security assessments and team awareness training will further strengthen resilience against evolving threats and foster a culture of ongoing security improvement

WORKS CITED

CWE 521 Weak Password Requirements - CVE Details, accessed August 1, 2025, https://www.cvedetails.com/cwe-details/521/Weak-Password-Requirements.html

CWE-521 - Security Database, accessed August 1, 2025, https://www.security-database.com/cwe.php?name=CWE-521

OWASP Top Ten | OWASP Foundation, https://owasp.org/www-project-top-ten/

CWE-200: Exposure of Sensitive Information to an ... - CWE, https://cwe.mitre.org/data/definitions/200.html

OWASP TOP 10 CWE Coverage - Mend.io Documentation,https://docs.mend.io/platform/latest/owasp-top-10-cwe-coverage

CWE 693 Protection Mechanism Failure - CVE Details, , https://www.cvedetails.com/cwe-details/693/Protection-Mechanism-Failure.html

CWE-693: Protection Mechanism Failure (4.17) - CWE, https://cwe.mitre.org/data/definitions/693.html