Web Application Penetration Testing Report

Prepared from OpenVAS Scan Results

Executive Summary

This penetration testing report summarizes the security vulnerabilities identified during a network and web application assessment. The scan, performed using OpenVAS, revealed multiple high, medium, and low severity issues that could potentially compromise the confidentiality, integrity, and availability of the target system. The objective of this report is to present these findings in a concise and actionable manner, along with recommended mitigations.

Scope

Target Host: 192.168.152.129 Assessment Type: Automated Vulnerability Scan & Manual Analysis Tool Used: OpenVAS Scan Date: 5 June 2025

Methodology

1. Reconnaissance and service identification. 2. Automated scanning using OpenVAS for known vulnerabilities. 3. Manual validation and classification of findings. 4. Categorization by severity based on CVSS scores. 5. Recommendations for remediation.

Vulnerability Summary

Severity	Count
High	24
Medium	40
Low	6

Key Findings by Severity

High Severity Issues

- Outdated Ubuntu Linux OS (EOL) – No security updates since 2013.
- Java RMI Server RCE (CVE-2011-3556).
- Multiple backdoors (Ingreslock, UnrealIRCd, vsftpd).
- VNC with weak/default credentials.
- Apache Tomcat Ghostcat vulnerability (CVE-2020-1938).
- Multiple outdated and vulnerable PHP versions with RCE potential.
- Misconfigured FTP allowing brute-force/default credentials.
- PostgreSQL with default password 'postgres'.
- Dangerous HTTP methods (PUT/DELETE) enabled.

Medium Severity Issues

- Weak SSH configuration.
- SMB service exposure.
- Outdated PostgreSQL service.
- Exposed SMTP, Telnet services.

Low Severity Issues

- ICMP timestamp responses enabled.
- General information disclosures.

Recommendations

1. Upgrade all outdated operating systems and applications immediately. 2. Disable unnecessary and insecure services (rexec, rsh, rlogin, Telnet). 3. Enforce strong authentication (disable default credentials). 4. Patch Apache Tomcat, PHP, and PostgreSQL to the latest supported versions. 5. Restrict access to management interfaces and sensitive ports. 6. Disable dangerous HTTP methods (PUT, DELETE) unless required. 7. Implement network segmentation and firewall rules to limit exposure. 8. Conduct periodic vulnerability scans and penetration tests.