Cybersecurity Functional and Performance Testing Report

1. Introduction

This report details the functional and performance testing procedures conducted during a cybersecurity assessment of three vulnerable web applications. The objective of the assessment was twofold: to verify the existence of identified vulnerabilities (functional testing) and assess the potential performance impact of exploiting these vulnerabilities (performance testing). The three target applications were:

1. **bWAPP** - www.itsecgames.com
2. **OWASP Juice Shop** - www.owasp.org
3. **Vulnerable Web Application** - testphp.vulnweb.com

---

2. Testing Methodology

2.1 Functional Testing

For functional testing, we employed a combination of manual techniques and automated scanning tools to confirm the presence of each vulnerability. The goal was to verify that each identified issue could be reliably exploited.

Tools used for functional testing:

- **Burp Suite**
- **OWASP ZAP**
- **Dirb/Gobuster** (for directory fuzzing)
- **Nessus** (for network scanning)
- **Custom Scripts** (for specific vulnerabilities)

Test cases were documented with precise steps for reproducing each vulnerability, ensuring consistency across multiple attempts. Each vulnerability's exploitability was verified, and we confirmed whether the risks posed were severe enough to justify remediation.

2.2 Performance Testing

Performance testing was focused on understanding the impact of vulnerability scanning and exploitation on the system's responsiveness and overall performance. While security was the main focus, we ensured that scans did not cause significant disruptions or system crashes.

Key performance metrics tracked:

- **CPU and memory usage** during scanning operations.
- **Response times** of web pages before, during, and after scans.

- **Concurrent scans and exploits** to observe how the application handles multiple simultaneous requests.

Testing tools included **Nessus** for scanning metrics and **Burp Suite**/**OWASP ZAP** for measuring web application performance.

---

3. Vulnerability Testing Results

3.1 bWAPP Testing Outcomes

Vulnerability	Test Method	Outcome
Insecure Direct Object References (IDOR)	Intercepted HTTP requests using Burp Suite and modified user IDs	Confirmed unauthorized data access, exploiting the IDOR vulnerability.
Cross-Site Request Forgery (CSRF)	Crafted a malicious HTML form and submitted it to a logged-in user	Malicious request executed without CSRF token validation, confirming vulnerability.
Security Misconfiguration	Attempted login using default credentials and searched for exposed configuration files	Successfully logged in with default credentials and found sensitive files.
Unvalidated Redirects	Modified redirect URLs to external domains	Application redirected users to malicious external sites, confirming vulnerability.
XML External Entity Injection (XXE)	Submitted specially crafted XML payloads to test for entity resolution	Successfully read files on the server, indicating an XXE vulnerability.

Key Observations:

- **IDOR** and **CSRF** were particularly critical, allowing attackers to steal data or perform unauthorized actions.
- **Security misconfigurations**, such as default credentials, made the system easy to exploit.

Each vulnerability was verified through multiple testing attempts to ensure reliability.

3.2 OWASP Juice Shop Testing Outcomes

Vulnerability	Test Method	Outcome
Cross-Site Scripting (XSS)	Injected JavaScript payloads into input fields and observed page responses	Successful script execution, confirming improper input sanitization.
Cross-Site Request Forgery (CSRF)	Created a malicious link to perform unauthorized actions on behalf of the user	Action executed successfully without CSRF token protection.
Insecure Direct Object References (IDOR)	Manipulated user IDs in request parameters to test unauthorized access	Unauthorized data access was achieved, proving IDOR vulnerability.
SQL Injection	Inserted SQL payloads (e.g., ' OR 1=1--) into login fields	Authentication bypassed, confirming SQL Injection vulnerability.
Broken Authentication	Attempted to log in with weak credentials and poor session management	Successful login with weak credentials, confirming broken authentication.

Key Observations:

- **SQL Injection** and **Broken Authentication** vulnerabilities are critical, potentially leading to full system compromise.
- **XSS** remains a significant concern, allowing attackers to steal sensitive information or hijack user sessions.

3.3 Vulnerable Web Application Testing Outcomes

Vulnerability	Test Method	Outcome
Outdated Software	Conducted a Nessus scan to identify outdated PHP and Apache versions	Found outdated versions with known security issues.
Open Ports	Performed a network scan using Nmap	Identified unused open ports, potentially exposing the system to attacks.
Weak Encryption	Tested SSL/TLS settings using Nessus	Discovered weak encryption algorithms and outdated protocols.

Vulnerability	Test Method	Outcome
Zero-Day Exploit Susceptibility	Flagged potential zero-day vulnerabilities via Nessus	Identified outdated software with the possibility of zero-day exploitation.
Cross-Site Scripting (XSS)	Injected payloads into input fields	XSS vulnerability confirmed, allowing for script execution.

Key Observations:

- **Outdated software** and **zero-day vulnerability risks** were the most concerning, as they could lead to a total system compromise.
- **Weak encryption** leaves data vulnerable to interception, particularly during transactions.

Each of these vulnerabilities was manually verified to ensure their validity.

---

4. Performance Testing Results

4.1 Observations

Scan Impact:

- **Minimal System Impact**: The vulnerability scans caused only slight performance degradation. For example, slight latency (1-2 seconds) was observed during high-intensity scans on **bWAPP** and **OWASP Juice Shop**.
- **Moderate CPU Usage**: During scans on the **Vulnerable Web Application**, CPU usage spiked temporarily but quickly normalized.

Recommendations for Performance Optimization:

- Conduct vulnerability scans during off-peak hours to avoid disrupting service.
- Ensure systems have adequate resources (CPU, RAM) to handle the additional load from security scans.

4.2 Performance Considerations

Testing confirmed that the resource usage during scanning was manageable. There were no significant outages or crashes, but there were minor slowdowns during periods of peak scanning. This suggests that while the performance impact is minimal, it’s important to schedule scans strategically to minimize any user experience degradation.

---

5. Conclusions

5.1 Key Findings

- **Critical Vulnerabilities**: Vulnerabilities like **SQL Injection**, **Broken Authentication**, and **Zero-Day Exploit Susceptibility** require immediate remediation due to their potential for complete system compromise.
- **Moderate Vulnerabilities**: Issues like **XSS**, **Outdated Software**, and **Weak Encryption** should be addressed promptly to prevent security breaches and data loss.

5.2 Performance Impact

- Scans did not cause significant performance degradation, but scheduling during low-traffic periods is recommended.

5.3 Future Recommendations

- **Patching and Remediation**: Prioritize patching high-severity vulnerabilities immediately, as they pose the most significant risks.
- **Continuous Security Testing**: Regular security assessments are essential to keep up with emerging threats.
- **DevSecOps Integration**: Incorporating security into the development lifecycle will help identify vulnerabilities earlier in the development process.
- **Load Testing**: For systems with strict performance requirements, additional load testing and stress testing are recommended to ensure that performance under heavy load remains stable.

---

6. References

- **OWASP Testing Guide** - OWASP Official Website
- **Nessus Documentation** - Tenable
- **Burp Suite Official Documentation** - Burp Suite
- **OWASP Juice Shop Official Site** - OWASP Juice Shop
- **bWAPP Official Site** - bWAPP
