Functional and Performance Testing

1. Introduction

This document outlines the functional and performance testing procedures carried out during our cybersecurity assessment. Our primary goal is to verify that each identified vulnerability indeed exists (functional testing) and to evaluate any potential impact on system performance or response times (performance testing).

We tested three target environments:

1. bWAPP – http://www.itsecgames.com/
2. OWASP Juice Shop – https://owasp.org/www-project-juice-shop/
3. Test PHP Vulnerable Website – http://testphp.vulnweb.com/ (with Nessus scans)

2. Testing Approach

1. Functional Testing

- We performed manual and automated vulnerability checks to confirm the existence and exploitability of each vulnerability.
- Tools used: Burp Suite, OWASP ZAP, Dirb, Gobuster, Nessus, and custom scripts where needed. We documented each test case, the steps taken, and the observed outcomes.
-

2.Performance Testing

- While our primary focus was security, we also monitored server response times and resource utilization during scans and exploit attempts.
- We ensured that vulnerability scanning did not excessively degrade the performance or availability of the target applications.
- Tools used: Nessus (provides scan performance metrics) and manual observation of response times in Burp Suite or OWASP ZAP.

3. Functional Testing

3.1 bWAPP (http://www.itsecgames.com/)

S. No.	Vulnerability	Test Method	Outcome
1	Insecure Direct Object References (IDOR)	- Intercepted requests using Burp Suite. - Modified user_id parameter to different values. - Observed if restricted data was accessible.	Confirmed unauthorized data access, proving IDOR vulnerability.
2	Cross-Site Request Forgery (CSRF)	- Created a malicious HTML form mimicking a password change. - Logged in as a victim in a separate session. - Submitted the malicious form to see if changes were applied.	Confirmed no CSRF token or verification; the malicious request was successful.
3	Security Misconfiguration	- Attempted default credentials (bee/bug). - Searched for exposed config files via Dirb/Gobuster.	Successfully logged in with defaults; found / phpinfo.php and .bak files accessible.
4	Unvalidated Redirects and Forwards	- Looked for redirect parameters (e.g., redirect.php?url=). - Modified the URL to point to external malicious domains.	Verified the application redirected users to untrusted sites without any validation.
5	XML External Entity Injection (XXE)	- Submitted malicious XML payloads. - Checked if external entities (e.g., <!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd">]>) were processed.	Successfully read local files, confirming XXE vulnerability.

Observations

- IDOR and CSRF were the most impactful vulnerabilities, allowing unauthorized data access and unauthorized user actions, respectively.
- Security Misconfigurations like default credentials significantly lowered the barrier to exploitation.

Functional Validation

- Each vulnerability was reproducible and consistent across multiple test attempts, confirming the findings' reliability.

3.2 OWASP Juice Shop (https://owasp.org/www-projectjuice-shop/)

S. No.	Vulnerability	Test Method	Outcome
1	Cross-Site Scripting (XSS)	- Injected malicious scripts (<script>alert('XSS')</script>) into search fields. - Observed if script execution occurred in the returned page.	Confirmed script execution in the browser, indicating improper input validation/ encoding.
2	Cross-Site Request Forgery (CSRF)	- Crafted malicious links/forms to perform actions on a logged-in user's behalf (e.g., password change). - Observed successful execution without CSRF tokens.	Verified that user actions could be triggered silently, indicating lack of CSRF protection.
3	Insecure Direct Object References	- Modified resource identifiers (user IDs, order IDs) in URLs and request parameters. - Checked if we could access or alter another user's data.	Confirmed unauthorized data access or modification, demonstrating IDOR vulnerability.
4	SQL Injection	- Inserted SQL payloads (e.g., ' OR 1=1--) in login fields. - Monitored the application's error messages or login bypass.	Bypassed authentication with malicious queries, proving SQL Injection.
5	Broken Authentication	- Exploited weak session tokens or session management flaws. - Tested known default/weak passwords to see if unauthorized access was possible.	Achieved unauthorized access and user impersonation, confirming broken authentication.

Observations

- SQL Injection and Broken Authentication can lead to full compromise of the application and database. Cross-Site Scripting remains a critical flaw for user session hijacking and data theft.
-

Functional Validation

- Each vulnerability was confirmed with consistent exploit steps and verified with at least one additional retest to rule out false positives.

3.3 Test PHP Vulnerable Website (http://testphp.vulnweb.com/)

We ran Nessus scans and manual tests on this target, focusing on common vulnerabilities like XSS, outdated software, and open ports.

S. No.	Vulnerability	Severity	Plugin/ ID	Test Method	Outcome
1	Outdated Software	High	10345	- Nessus scanned the server's software versions (Apache, PHP, etc.). - Confirmed via manual version checks.	Identified older versions with known security patches missing, elevating the risk of exploitation.
2	Open Ports	Medium	8576	- Conducted network port scanning (Nmap). - Validated unusual open ports for potential vulnerabilities.	Found multiple open ports not in use; they exposed unnecessary services that attackers could probe.
3	Weak Encryption	High	65432	- Nessus checked SSL/ TLS configurations. - Tested ciphers to confirm usage of deprecated algorithms.	Discovered insecure SSL ciphers; the site accepted TLS protocols with known vulnerabilities.
4	Zero-Day Exploit Susceptibility	Critical	78901	- Nessus flagged possible zero-day vulnerabilities based on the server's signature and outdated software.	Potential zero-day exploit risk indicated. Further manual checks recommended for confirmation.
5	Cross-Site Scripting (XSS)	High	N/A	- Injected script payloads in the site's search/ comment fields. - Observed successful script execution.	Verified stored/reflected XSS possibilities, showing insufficient input sanitization.

Observations

- Critical zero-day exploit susceptibility and outdated software are the most dangerous, potentially leading to a full system takeover if exploited by advanced attackers.
- Weak encryption endangers data-in-transit confidentiality, especially if sensitive data is exchanged.

Functional Validation

- Nessus findings were manually verified wherever possible, confirming they were not false positives. Cross-referencing plugin IDs and CVE references helped us validate the exploitability.
-

---

4. Performance Testing

4.1 Methodology

- Scan Overheads: Monitored CPU, memory, and response times on the target system during Nessus and manual scanning to detect any excessive resource usage.
- Response Times: Recorded average page load times before, during, and after the scans using basic browser dev tools or scanning tool logs.
- Concurrent Requests: While not a full load test, we observed how the target websites behaved when multiple vulnerability scans or test scripts were run simultaneously.

4.2 Observations

- Minimal Service Disruption: None of the targets experienced significant downtime or crashes, although some temporary slowdowns were noted when Nessus performed intense scanning. Stable
- Response Times: bWAPP and Juice Shop remained largely responsive, with slight increases in latency (1–2 seconds) during peak scan intervals.
- Resource Usage: On the testphp.vulnweb.com target, CPU utilization spiked to moderate levels during port scanning, but stabilized quickly after scans completed.

4.3 Conclusion of Performance Tests

- The vulnerability scanning process had a manageable impact on performance, with only brief periods of elevated resource usage.
- Proper scheduling of scans (e.g., off-peak hours) is recommended to minimize disruption for production environments.

---

5. Overall Observations and Recommendations

1. Functional Testing Success:

- Each vulnerability identified was confirmed through manual exploitation or scanning tool verification.
- Reproducible results validate the authenticity and severity of findings.

2 .Performance Impact:

- Scans did not cause significant outages but introduced minor latency.
- Ensuring adequate system resources and scheduling scans during low-traffic periods can mitigate performance concerns.

3. Remediation Priority:

- Critical or High-severity vulnerabilities (e.g., IDOR, SQL Injection, Broken Authentication, Zero-Day Exploits) should be patched immediately.
- Medium and Low-severity findings should still be addressed to maintain a robust security posture.

4.Future Testing:

- Continuous testing is vital as new vulnerabilities emerge.
- Incorporating DevSecOps practices can help catch security flaws earlier in development. Consider
- load testing or stress testing if the application has strict performance or uptime requirements.

6. References

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