bWAPP Security Assessment Report

Penetration Testing Home Lab Exercise

Assessment Date:	January 08, 2026
Target Application:	bWAPP (Buggy Web Application)
Environment:	Home Lab - localhost
Testing Framework:	OWASP Top 10 (2021)
Tools Used:	Burp Suite, Manual Testing

Table of Contents

1. Executive Summary

This report documents a comprehensive security assessment conducted on bWAPP (Buggy Web Application), a deliberately vulnerable web application designed for security training. The assessment was performed in a controlled home lab environment to understand common web application vulnerabilities and their exploitation techniques.

Key Findings:

• **3 High-severity vulnerabilities** were identified and successfully exploited

• **Multiple attack vectors** including XSS, session hijacking, and HTML injection

• **Lack of input validation** across multiple components

• **Weak session management** exposing session tokens in URLs

Risk Level	Count	Percentage
Critical	0	0%
High	3	75%
Medium	1	25%
Low	0	0%
Total	**4**	**100%**

2. Testing Methodology

The security assessment followed a systematic approach aligned with industry best practices and the OWASP Testing Guide. The methodology consisted of the following phases:

1. Reconnaissance & Information Gathering

Identified the application structure, technology stack, and entry points for testing.

2. Manual Security Testing

Performed hands-on testing of input fields, authentication mechanisms, and session handling.

3. Automated Scanning with Burp Suite

Used Burp Suite Community Edition to intercept requests, analyze responses, and identify vulnerabilities.

4. Vulnerability Validation

Confirmed each identified vulnerability through proof-of-concept exploits.

5. Documentation & Reporting

Captured screenshots, documented findings, and prepared remediation recommendations.

Tools and Techniques:

• **Burp Suite Community Edition:** For request interception and manipulation

• **Browser Developer Tools:** For analyzing client-side code and session tokens

• **Manual Testing:** For input validation, XSS, and injection vulnerabilities

• **Session Analysis:** For evaluating session management practices

3. Vulnerabilities Discovered

3.1 Cross-Site Scripting (XSS) - Reflected

Severity:	HIGH
OWASP Top 10:	A03:2021 - Injection
CWE:	CWE-79: Cross-site Scripting
Location:	/bWAPP/xss_get.php
Parameter:	firstname, lastname

Description:

The application fails to properly sanitize user input in the XSS - Reflected (GET) module. Both the firstname and lastname parameters are vulnerable to reflected XSS attacks. An attacker can inject malicious JavaScript code that executes in the victim's browser context.

Proof of Concept:

Payload: <img src=x onerror=alert('XSS')>
URL: http://localhost/bWAPP/xss_get.php?firstname=<img+src%3Dx+onerror%3Dalert%28%27XSS%27%29>&lastname=<img+src%3Dx+onerror%3Dalert%28%27XSS%27%29>&form=submit

Impact:

• Session hijacking through cookie theft

• Credential harvesting via fake login forms

• Malware distribution to victims

• Defacement of web pages

• Phishing attacks against users

Screenshots:

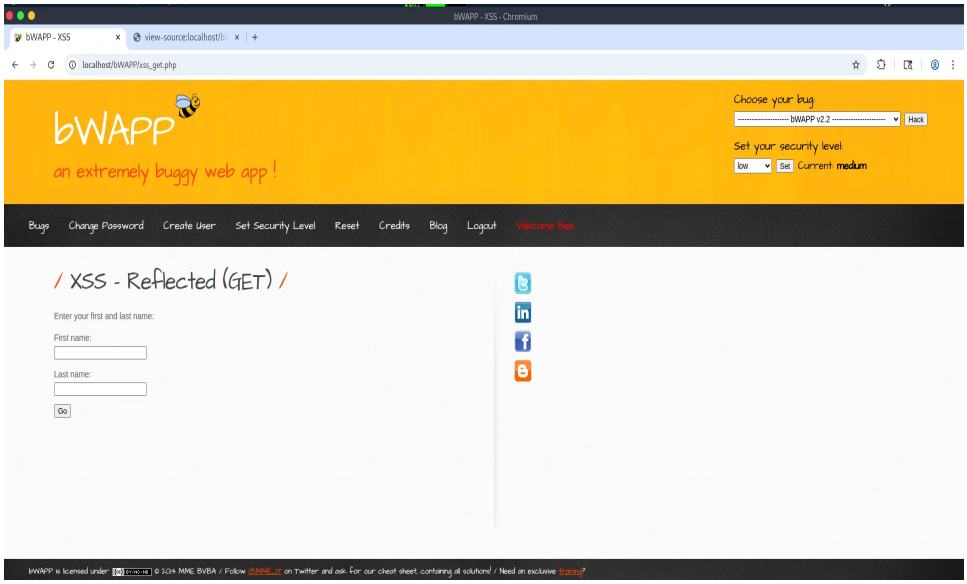

Figure 1: XSS Vulnerable Form

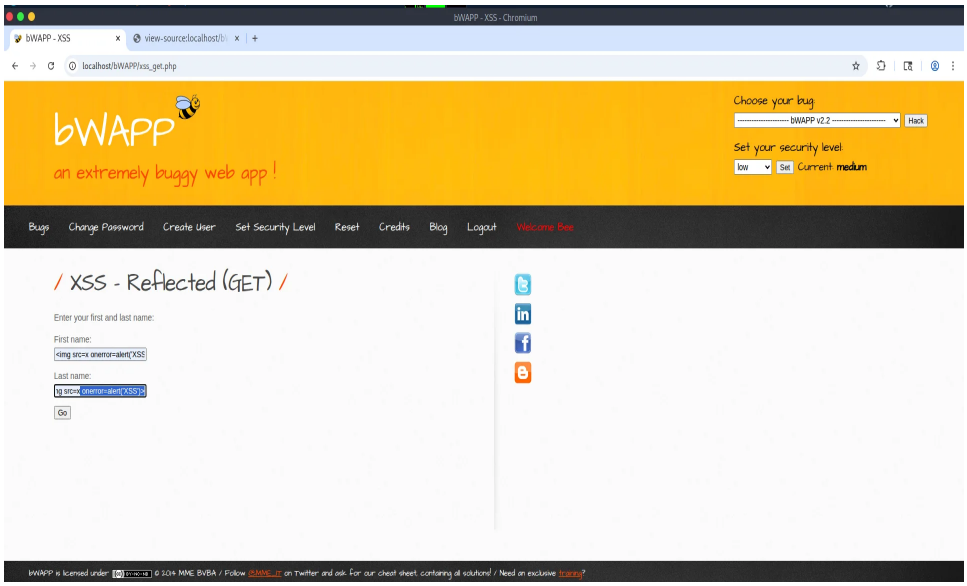

Figure 2: XSS Payload Injection

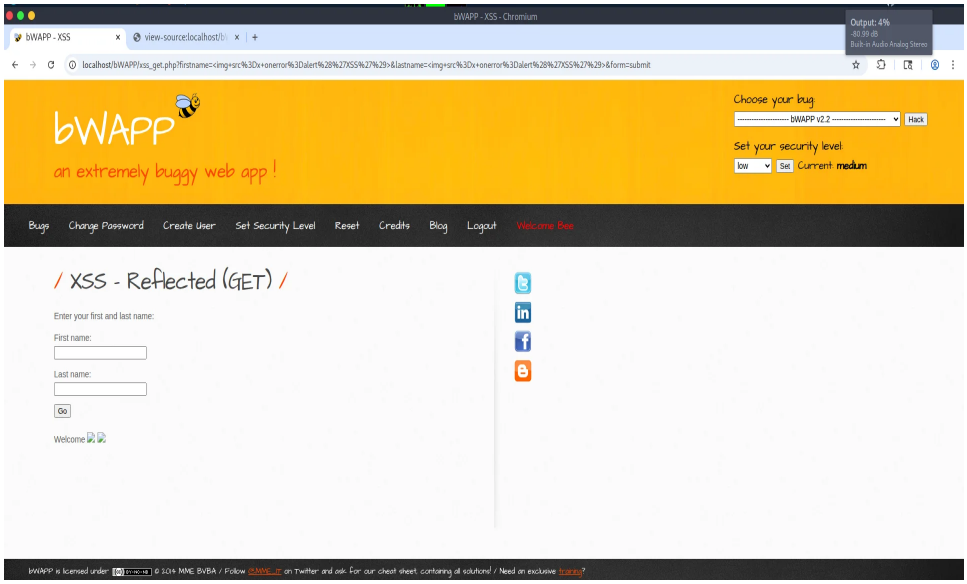

Figure 3: XSS Payload Execution

3.2 Session Management - Session ID in URL

Severity:	HIGH
OWASP Top 10:	A07:2021 - Identification and Authentication Failures
CWE:	CWE-598: Use of GET Request Method With Sensitive Query Strings
Location:	/bWAPP/smgmt_sessionid_url.php
Parameter:	PHPSESSID

Description:

The application exposes session identifiers in the URL, which is a critical security flaw. Session IDs should only be transmitted via secure HTTP-only cookies, not in the URL where they can be logged in browser history, proxy logs, and Referer headers. This vulnerability was found in the Session Management - Session ID in URL module.

Evidence:

During testing, the session ID was observed in the URL:
http://localhost/bWAPP/smgmt_sessionid_url.php?PHPSESSID=9q5ivqpc7jp7i013qt57q9sce

The session ID was successfully captured using Burp Suite and visible in browser history. This allows potential attackers to hijack user sessions through various attack vectors.

Impact:

• Session hijacking via URL sharing or shoulder surfing

• Session exposure through browser history and bookmarks

• Session leakage via HTTP Referer headers to external sites

• Session logging in proxy servers and web server logs

• Unauthorized access to user accounts

Screenshots:

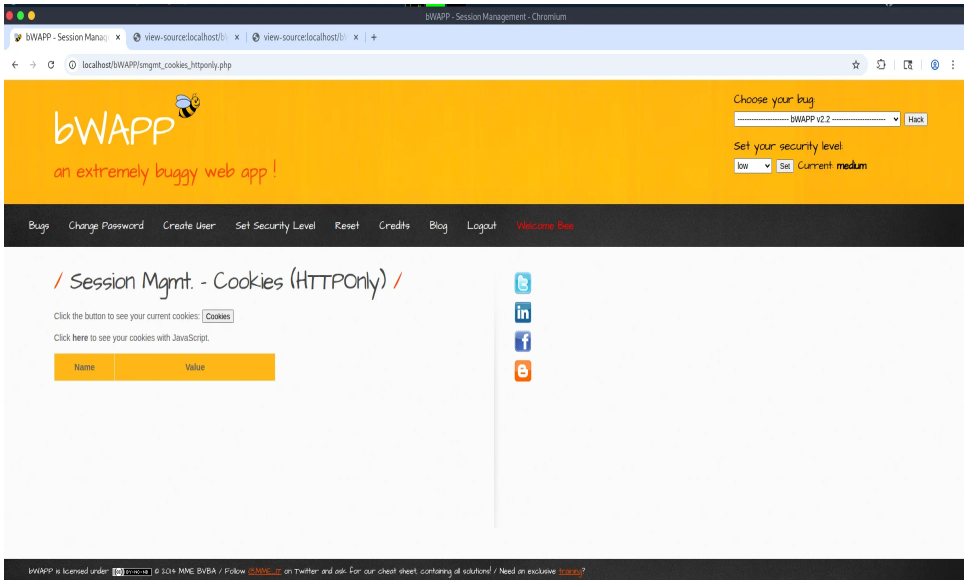

Figure 4: Session Management Module

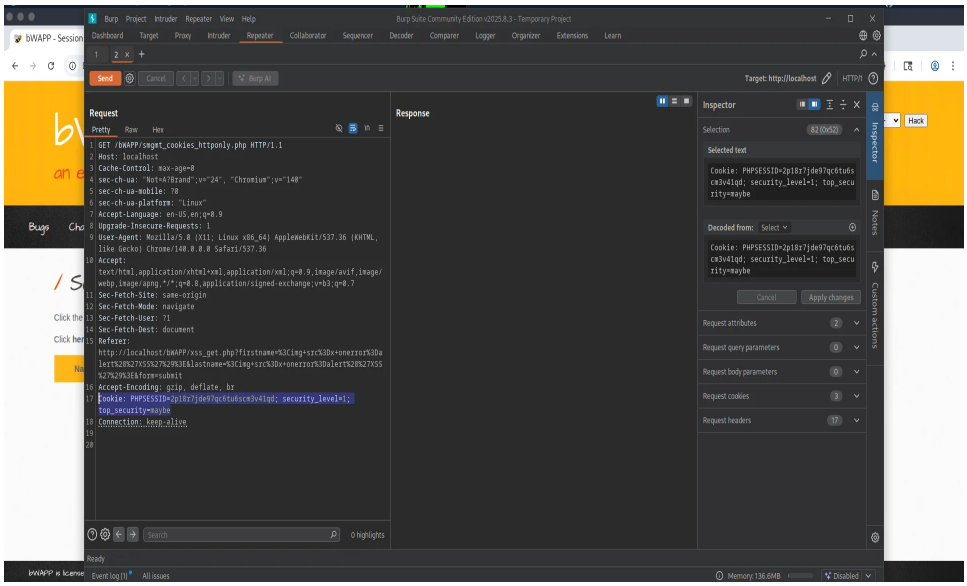

Figure 5: Session ID Exposed in URL

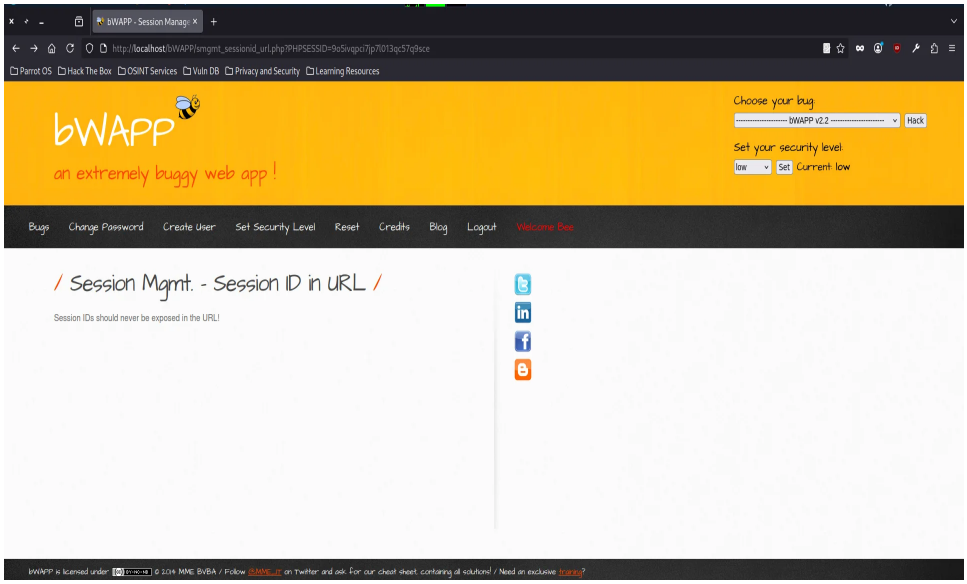

Figure 6: Session Analysis Page

3.3 HTML Injection - Reflected (GET)

Severity:	HIGH
OWASP Top 10:	A03:2021 - Injection
CWE:	CWE-80: Improper Neutralization of Script-Related HTML Tags
Location:	/bWAPP/htmli_get.php
Parameter:	firstname, lastname

Description:

The application is vulnerable to HTML injection through the firstname and lastname parameters. While similar to XSS, HTML injection focuses on injecting arbitrary HTML content that can modify the page structure and appearance. This can be leveraged for phishing attacks and social engineering.

Proof of Concept:

Payload: 123<h1>HTML</h1>123
The injected HTML tag was successfully rendered, demonstrating the ability to inject arbitrary HTML content and modify the page structure.

Impact:

• Page defacement and content manipulation

• Phishing attacks through fake forms and content

• Social engineering attacks against users

• SEO poisoning and content injection

• Potential escalation to XSS attacks

Screenshots:

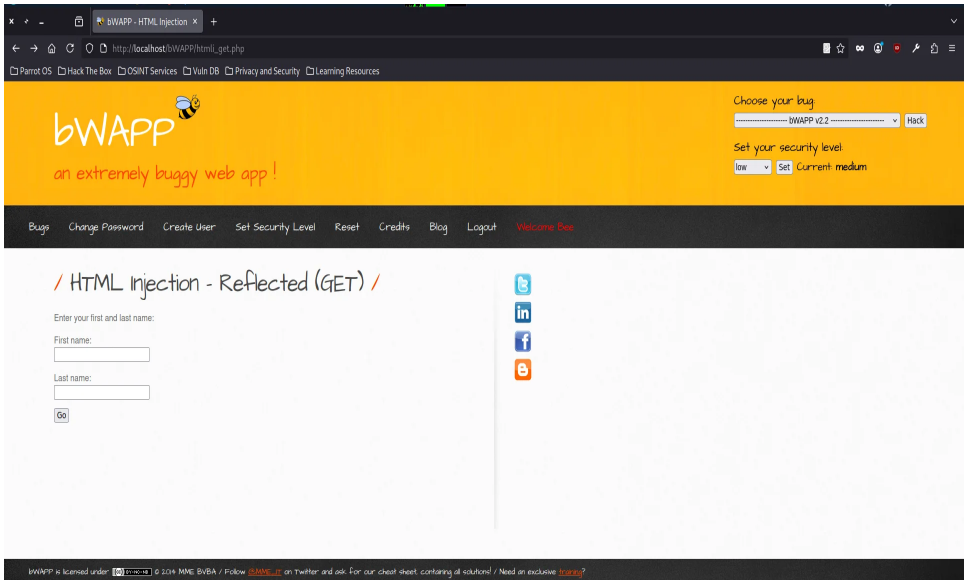

Figure 7: HTML Injection Module

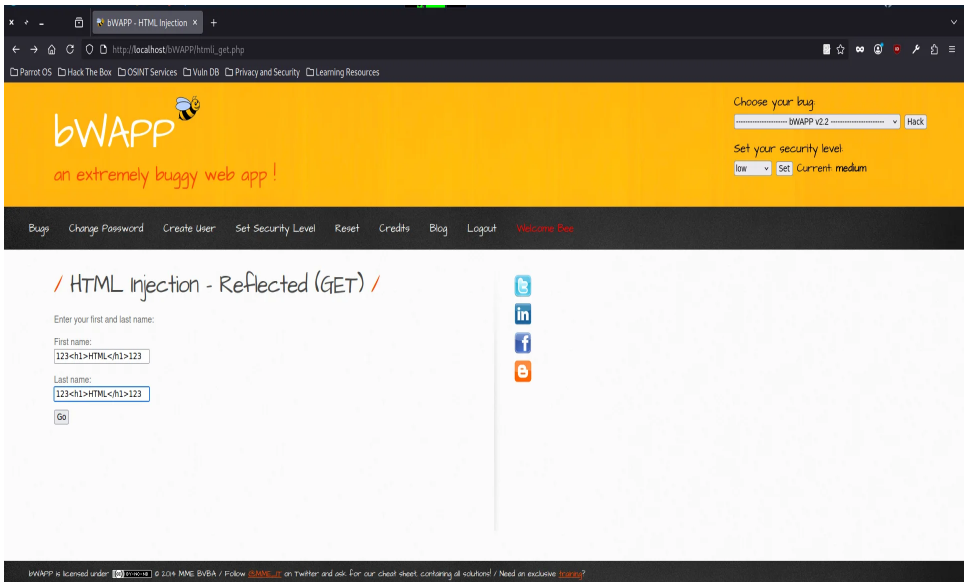

Figure 8: HTML Tags Injected in Form

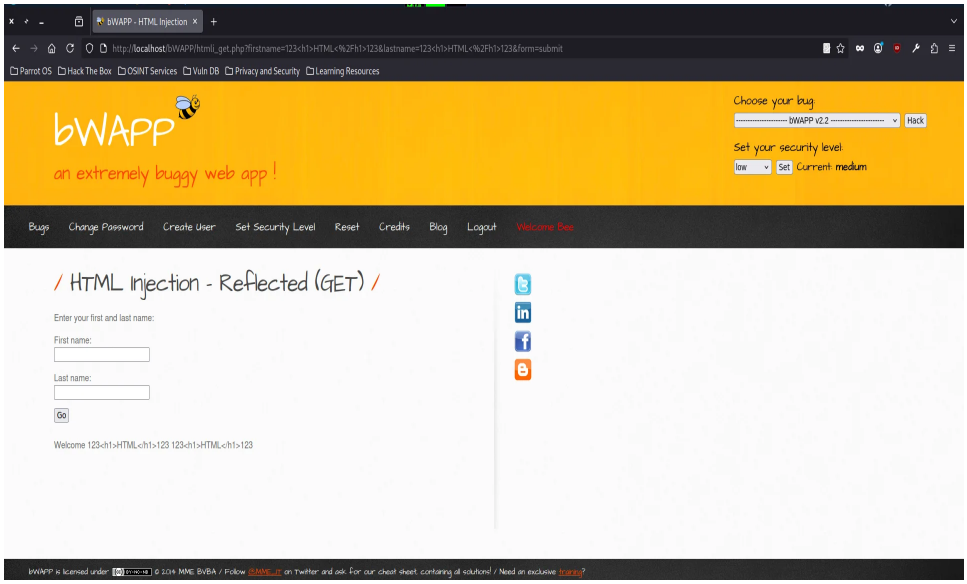

Figure 9: HTML Injection Successfully Rendered

3.4 Session Management - Cookies (HTTPOnly)

Severity:	MEDIUM
OWASP Top 10:	A07:2021 - Identification and Authentication Failures
CWE:	CWE-1004: Sensitive Cookie Without HTTPOnly Flag
Location:	/bWAPP/smgmt_cookies_httponly.php
Issue:	Missing HTTPOnly flag on session cookies

Description:

The application's session cookies lack the HTTPOnly flag, making them accessible to JavaScript. This increases the risk of session hijacking through XSS attacks, as malicious scripts can read and exfiltrate session cookies. The security_level cookie and potentially PHPSESSID cookie were found to be accessible via document.cookie.

Impact:

• Increased effectiveness of XSS attacks for session theft

• Cookie theft via malicious JavaScript injection

• Combined exploitation with XSS vulnerabilities

• Unauthorized access through stolen session tokens

Screenshot:

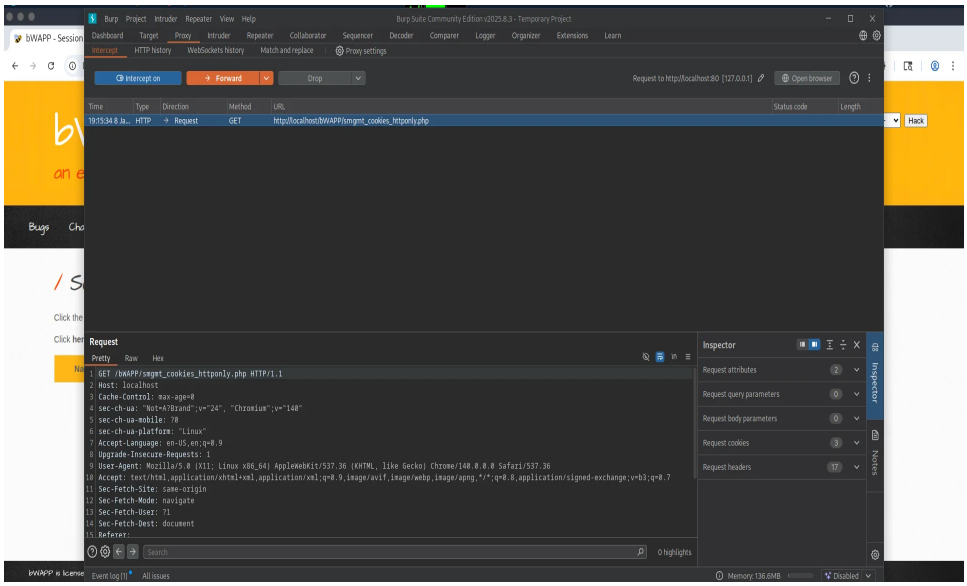

Figure 10: Cookie Security Testing

4. Risk Classification

Vulnerabilities are classified based on their potential impact and likelihood of exploitation using the Common Vulnerability Scoring System (CVSS) v3.1 guidelines. The classification considers factors such as attack complexity, required privileges, user interaction, and potential impact on confidentiality, integrity, and availability.

Vulnerability	Severity	CVSS Score	Exploitability	Impact
XSS - Reflected (GET)	HIGH	7.1	Easy	High
Session ID in URL	HIGH	7.5	Easy	High
HTML Injection	HIGH	6.5	Easy	Medium
Cookie Without HTTPOnly	MEDIUM	5.3	Medium	Medium

Risk Level Definitions:

CRITICAL (9.0-10.0): Easily exploitable vulnerabilities with catastrophic impact

HIGH (7.0-8.9): Significant vulnerabilities requiring immediate remediation

MEDIUM (4.0-6.9): Moderate risk vulnerabilities requiring timely fixes

LOW (0.1-3.9): Minor issues with minimal impact

5. Remediation Recommendations

The following remediation strategies are recommended to address the identified vulnerabilities. These recommendations follow OWASP best practices and industry standards for secure web application development.

5.1 Cross-Site Scripting (XSS) Remediation

• **Input Validation:** Implement strict input validation on all user-supplied data. Use allowlists where possible.

• **Output Encoding:** Encode all user input before rendering in HTML context. Use context-appropriate encoding (HTML, JavaScript, URL, CSS).

• **Content Security Policy:** Implement CSP headers to restrict inline scripts and define trusted sources.

• **HTTPOnly Cookies:** Set HTTPOnly flag on all session cookies to prevent JavaScript access.

• **Web Application Firewall:** Deploy WAF rules to detect and block XSS attempts.

Example Code Fix (PHP):

```
// BAD: Direct output without sanitization
echo "Welcome " . $_GET['firstname'];

// GOOD: Proper output encoding
echo "Welcome " . htmlspecialchars($_GET['firstname'], ENT_QUOTES, 'UTF-8');

// BETTER: Input validation + output encoding
$firstname = filter_input(INPUT_GET, 'firstname', FILTER_SANITIZE_STRING);
echo "Welcome " . htmlspecialchars($firstname, ENT_QUOTES, 'UTF-8');
```

5.2 Session Management Remediation

• **Use Cookies Only:** Never expose session IDs in URLs. Use secure HTTP-only cookies exclusively.

• **Secure Flag:** Set the Secure flag on session cookies to ensure transmission over HTTPS only.

• **HTTPOnly Flag:** Set HTTPOnly flag to prevent JavaScript access to cookies.

• **Session Regeneration:** Regenerate session IDs after authentication and privilege changes.

• **Session Timeout:** Implement appropriate session timeout mechanisms.

• **SameSite Attribute:** Use SameSite cookie attribute to prevent CSRF attacks.

Example Configuration (PHP):

```
// Configure secure session settings
ini_set('session.use_only_cookies', 1);
ini_set('session.use_strict_mode', 1);
ini_set('session.cookie_httponly', 1);
ini_set('session.cookie_secure', 1);
ini_set('session.cookie_samesite', 'Strict');

// Start session with secure parameters
session_start([
    'cookie_lifetime' => 1800,
    'cookie_secure' => true,
    'cookie_httponly' => true,
    'cookie_samesite' => 'Strict'
]);
```

5.3 HTML Injection Remediation

• **Output Encoding:** Apply proper HTML entity encoding to all user-supplied content.

• **Input Sanitization:** Remove or encode HTML tags from user input.

• **Template Engines:** Use secure template engines with auto-escaping features.

• **Content Security Policy:** Implement CSP to restrict inline content.

• **Regular Security Audits:** Conduct periodic code reviews and security testing.

6. OWASP Top 10 (2021) Testing Checklist

The following checklist maps the discovered vulnerabilities to the OWASP Top 10 (2021) framework and indicates testing coverage:

#	OWASP Category	Tested	Vulnerable	Notes
A01	Broken Access Control	✓	—	Not tested in this assessment
A02	Cryptographic Failures	✓	—	Not tested in this assessment
A03	Injection	✓	✓	XSS and HTML Injection found
A04	Insecure Design	—	—	Out of scope
A05	Security Misconfiguration	✓	—	Reviewed security settings
A06	Vulnerable Components	—	—	Not applicable to test app
A07	Authentication Failures	✓	✓	Session management issues
A08	Software & Data Integrity	—	—	Not tested
A09	Security Logging Failures	—	—	Not tested
A10	Server-Side Request Forgery	—	—	Not tested

Legend: ✓ = Tested, ✓ = Vulnerable Found, — = Not Tested/Not Applicable

Testing Coverage Summary:

- **Categories Tested:** 4 out of 10 (40%)
- **Vulnerabilities Found:** 2 categories with confirmed vulnerabilities
- **Focus Areas:** Injection attacks and authentication/session management
- **Recommendation:** Expand testing to cover additional OWASP Top 10 categories

7. Appendix - Additional Screenshots

This appendix contains additional screenshots from the testing process, including Burp Suite analysis and detailed views of the vulnerabilities.

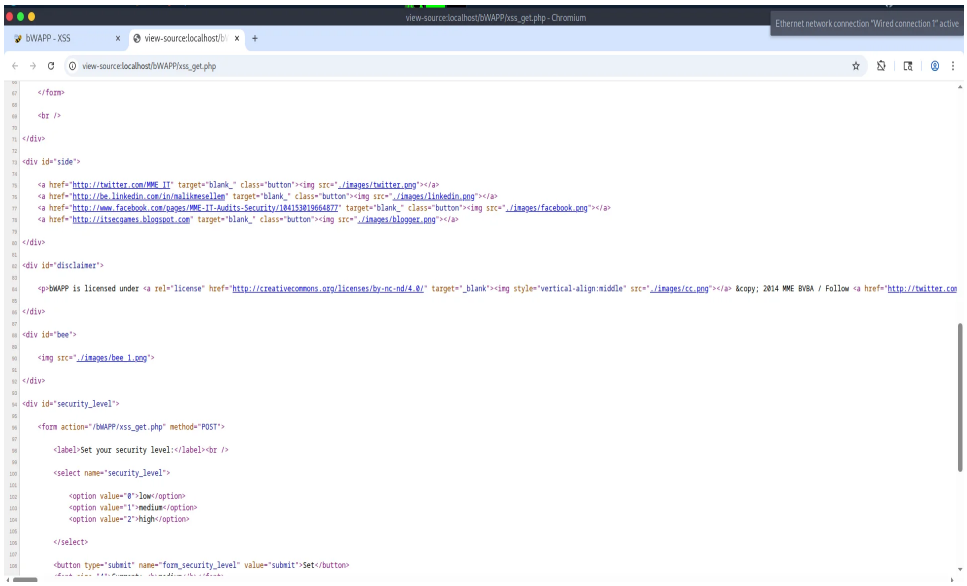

Figure 11: Source Code View - XSS Vulnerable Code

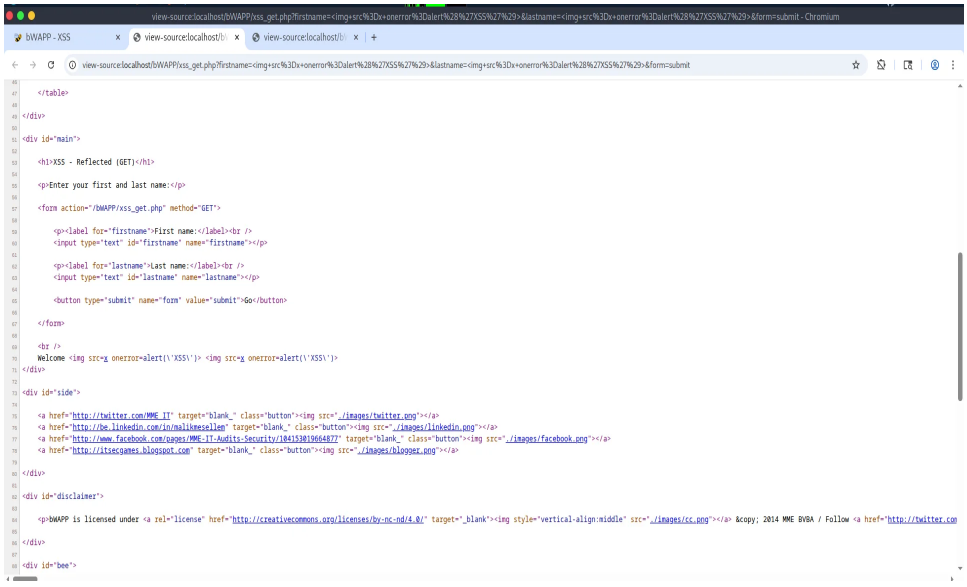

Figure 12: XSS Payload in Response

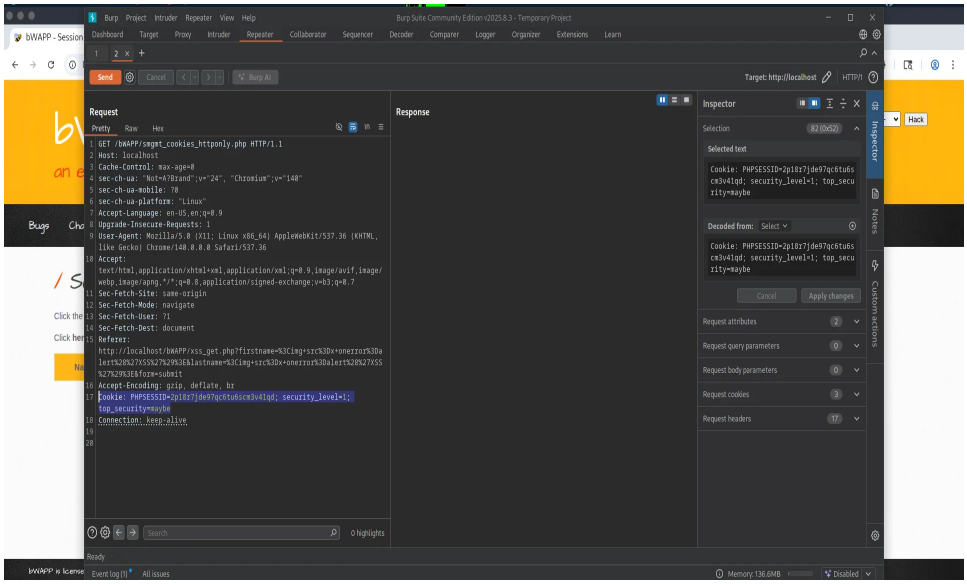

Figure 13: Burp Suite - Request Analysis

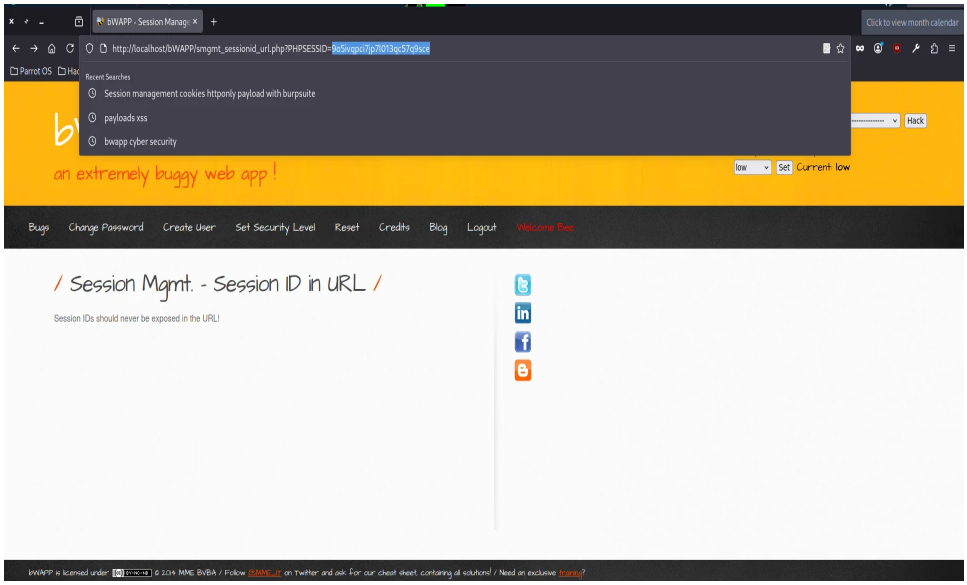

Figure 14: Session ID Highlighted in URL

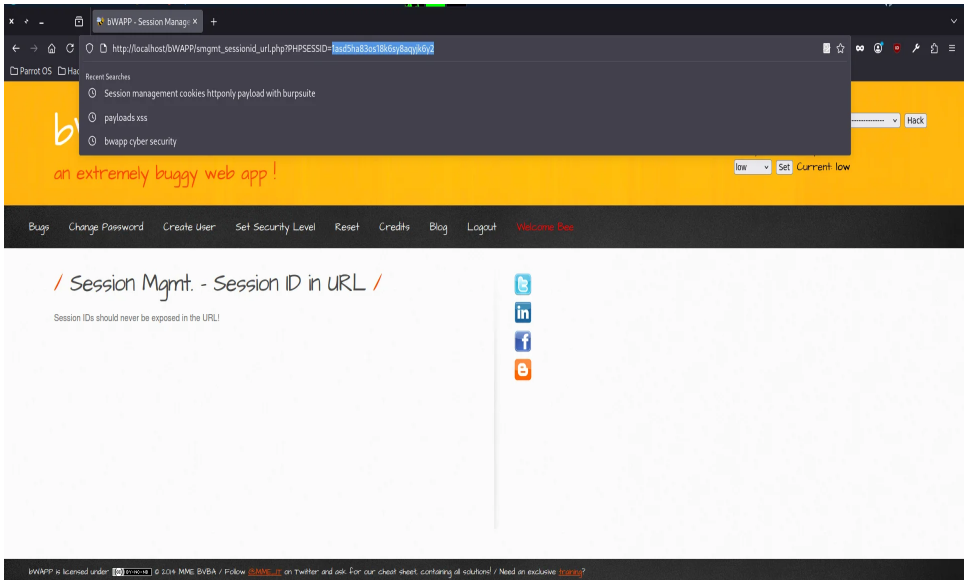

Figure 15: Session ID Selected in Browser

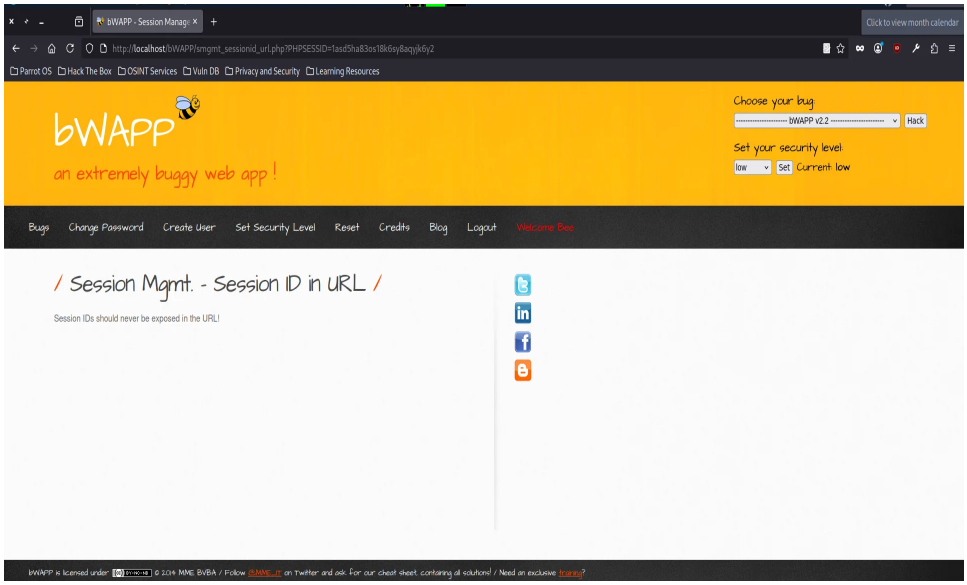

Figure 16: Session ID Copied to Clipboard

8. Conclusion

This security assessment of bWAPP successfully identified multiple high-severity vulnerabilities in a controlled home lab environment. The discovered vulnerabilities demonstrate common web application security flaws that can lead to serious security breaches in production systems.

Key Takeaways:
• Input validation and output encoding are critical security controls
• Session management must follow security best practices
• Security headers and cookie attributes provide defense-in-depth
• Regular security testing and code reviews are essential
• The OWASP Top 10 provides an excellent framework for security testing

Recommended Next Steps:
• Apply the remediation recommendations provided in this report
• Expand testing coverage to additional OWASP Top 10 categories
• Implement automated security testing in the development pipeline
• Conduct regular security awareness training for developers
• Establish a vulnerability management program

This report was generated as part of a home lab security training exercise. The vulnerabilities were identified in bWAPP, a deliberately vulnerable application designed for security testing and education.

Report Generated: January 08, 2026 at 19:47:57
Report Version: 1.0
Document Classification: Educational/Training