

Lab 2: OWASP Top 10 Web Application Testing for GRC Professionals

REG NO:2025/GRC/11032

To: CISO & Application Development Management

From: GRC Audit Team

Date: 21-09-2025

Executive Summary

An assessment of the OWASP BWA server multiple high-risk vulnerabilities that align with the OWASP Top 10. These issues indicate gaps in application security and pose a risk to both customer and company data.

Key Findings:

- Broken Access Control (A01:2021): By changing the typeid parameter from 6—2 in Bodgelt's product .jsp, allowed access to product categories without proper authorization. This confirms that unauthorized users could view or manipulate data they shouldn't have access to.
- SQL Injection (A03:2021): The search field is vulnerable and could allow attackers to retrieve or modify the database.
- Cryptographic Failures (A02:2021): Credentials are transmitted in cleartext, which could be intercepted.
- Other WordPress Scan Findings: Automated scanning of the WordPress site revealed SSL Injection, HTTP Request Smuggling, and LDAP Injection vulnerabilities. These were not manually exploited, but highlight additional potential risks.

Recommended Actions:

1. **Immediate Mitigation:** Restrict access to affected systems until proper access controls are implemented.
2. **Remediation:** Apply server-side authorization checks, enforce ACLs, and use indirect object references to prevent unauthorized data access.
3. **Process Improvement:** Provide security training to developers focused on the OWASP Top 10. Integrate SAST/DAST tools like Burp Suite into CI/CD pipelines.
4. **Penetration Testing:** Conduct a full third-party pentest before deploying similar applications in production.

OWASP Top 10 category	Vulnerability Example	Affected Application	Inherent Risk {L/M/H}	Compliance violation	Business impact
Ao1: Broken Access Control	Unprotected Admin Page	Bodgelt store	H	PCI DSS 7.2.1	Unauthorized access, privilege escalation
A02: Cryptography failures	Cleartext password transmission	WebGoat	H	PCI DSS 4.1	Credential theft, account takeover
A03: injection	SQL injection in search	Bodgelt store	H	PCI DSS 6.5.1	Full database compromise data loss
A07: Identification Failures	Weak Password Policy	Bodgelt Store	H	NIST CSF PR.AC-	Unauthorized access, initial breach vector

Figure 1: Burp Suite scan results for the WordPress instance showing identified vulnerabilities including SQL Injection, Cross Site Scripting, and Broken Access Controls.

Burp Suite Professional v2025.7.4 - 2025-09-20 - licensed to chika

Tasks

- 4. Exploring Backup file on 192.168.37.131 (Backup file, Error)
- 3. Crawl and audit of 192.168.37.131 (Crawl and Audit - Lightweight, Finished, Issues: 3, 19, 18, 63)
- 2. Live audit from Proxy (all traffic) (Audit checks - passive, Capturing, Issues: 0, 0, 22, 109)
- 1. Live passive crawl from Proxy (all traffic) (Add links, Add item itself, same domain and URLs in suite scope, Capturing, Issues: 0, 0, 22, 109)

3. Crawl and audit of 192.168.37.131 Rate this scan

Summary **Audit items** **Issues** **Event log** **Logger** **Audit log** **Live crawl view**

Most serious vulnerabilities found (live) View all

Issue type	Host	Time
Flash cross-domain policy	https://192.168....	17:18:29 20 Sept...
Flash cross-domain policy	http://192.168.3....	17:18:36 20 Sept...
Cross-site scripting (DOM-bas...	https://192.168....	17:26:59 20 Sept...
TLS certificate	https://192.168....	17:18:29 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:46 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:46 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:48 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:50 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:51 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:53 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:54 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:55 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:58 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:59 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:28:59 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:29:00 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:29:01 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:29:02 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:29:02 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:29:03 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:29:03 20 Sept...
TLS cookie without secure flag...	https://192.168....	17:29:04 20 Sept...
Strict transport security not en...	https://192.168....	17:28:42 20 Sept...

Task configuration View configural...

Task type:	Crawl & audit
Scope:	192.168.37.131
Configuration:	Crawl and Audit - Lightweight

Task progress

Total audit items:	74	Discovered locations:	4
Audit items in progress:	0	Discovery actions pending:	0
Audit items completed:	74	Requests:	7
Network errors:	3	Requests per second:	0

Task log

```

> Auditing "https://192.168.37.131/AppSensorDemo/login.jsp" for Web Cache
sizing
> Auditing "https://192.168.37.131/ESAPI-Java-SwingSet-Interactive/main" for
ackup Files Append Extension
> Auditing "https://192.168.37.131/ESAPI-Java-SwingSet-Interactive/main" for
ackup Files Append Filename
> Auditing "https://192.168.37.131/ESAPI-Java-SwingSet-Interactive/main" for
ackup Files Replace Extension
> Auditing "https://192.168.37.131/ESAPI-Java-SwingSet-Interactive/main" fo

```

Event log (9) All issues (234) Memory: 303.9MB

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

site map Scope Issues

[Open in browser](#)

Issue definitions

This listing contains the definitions of all issues that can be detected by Burp Scanner.

Name	Typical severity	Type index
OS command injection	High	0x00100100
SQL injection	High	0x00100200
SQL injection (second order)	High	0x00100210
ASP.NET tracing enabled	High	0x00100280
File path traversal	High	0x00100300
XML external entity injection	High	0x00100400
LDAP injection	High	0x00100500
XPath injection	High	0x00100600
XML injection	Medium	0x00100700
ASP.NET debugging enabled	Medium	0x00100800
Broken access control	Information	0x00100850
HTTP PUT method is enabled	High	0x00100900
Out-of-band resource load (HTTP)	High	0x00100a00
File path manipulation	High	0x00100b00
PHP code injection	High	0x00100c00
Server-side JavaScript code injection	High	0x00100d00
Perl code injection	High	0x00100e00
Ruby code injection	High	0x00100f00
Python code injection	High	0x00100f10
Expression Language injection	High	0x00100f20
Unidentified code injection	High	0x00101000
Server-side template injection	High	0x00101080
Ssi injection	High	0x00101100
Cross-site scripting (stored)	High	0x00200100
HTTP request smuggling	High	0x00200140
Client-side desync	High	0x00200141
Web cache poisoning	High	0x00200180
HTTP response header injection	High	0x00200200
Cross-site scripting (reflected)	High	0x00200300
Client-side template injection	High	0x00200308
Cross-site scripting (DOM-based)	High	0x00200310
Cross-site scripting (reflected DOM-based)	High	0x00200311
Cross-site scripting (stored DOM-based)	High	0x00200312
Client-side prototype pollution	Information	0x00200316
JavaScript injection (DOM-based)	High	0x00200320
JavaScript injection (reflected DOM-based)	High	0x00200321
JavaScript injection (stored DOM-based)	High	0x00200322
Path relativization attack (absolute)	Information	0x00300228

OS command injection

Description

Operating system command injection vulnerabilities arise when an application incorporates user-controllable data into a command that is processed by a shell command interpreter. If the user data is not strictly validated, an attacker can use shell metacharacters to modify the command that is executed, and inject arbitrary further commands that will be executed by the server.

Remediation

If possible, applications should avoid incorporating user-controllable data into operating system commands. In almost every situation, there are safer alternative methods of performing server-level tasks, which cannot be manipulated to perform additional commands than the one intended.

If it is considered unavoidable to incorporate user-supplied data into operating system commands, the following two layers of defense should be used to prevent attacks:

- The user data should be strictly validated. Ideally, a whitelist of specific accepted values should be used. Otherwise, only short alphanumeric strings should be accepted. Input containing any other data, including any conceivable shell metacharacter or whitespace, should be rejected.
- The application should use command APIs that launch a specific process via its name and command-line parameters, rather than passing a command string to a shell interpreter that supports command chaining and redirection. For example, the Java API Runtime.exec and the ASP.NET API Process.Start do not support shell metacharacters. This defense can mitigate the impact of an attack even in the event that an attacker circumvents the input validation defenses.

References

- [Web Security Academy: OS command injection](#)

Vulnerability classifications

Event log (9) • All issues (234) •

Memory: 313.4MB

Figure 2: Successful Broken Access Control test using Burp Repeater. Changing the typeid parameter allowed access to restricted product information.

burp Suite Professional v2025.1.4 - 2025-09-20 - licensed to cnika

Repeater tab selected.

Request:

```
GET /bodgeit/product.jsp?typeid=2 HTTP/1.1
Host: 192.168.37.131
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.37.131/bodgeit/register.jsp
Cookie: JSESSIONID=181F8A5EBF16B6A52F8865FE7816AFER;
icgroupwithpersist=nada
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Response:

```
<td align="left" valign="top" width="25%>
Doodahs</a><br/>
Gizmos</a><br/>
Thingamajigs</a><br/>
Thingies</a><br/>
Whatchamacallits</a><br/>
Whatsits</a><br/>
Widgets</a><br/>
<br/><br/><br/><br/><br/><br/><br/><br/>
</td>
<td align="top" width="70%">
<h3>Products</h3><center><table border="1" width="80%" class="border">
<tr><th>Product</th><th>Type</th><th>Price</th></tr>
<tr>
<td><a href="product.jsp?prodid=4">Thingie 1</a><td>Thingies
</td></td><td align="right">\\\\\\\$3.00</td>
</tr>
<tr>
<td><a href="product.jsp?prodid=5">Thingie 2</a><td>Thingies
</td></td><td align="right">\\\\\\\$3.20</td>
</tr>
<tr>
<td><a href="product.jsp?prodid=6">Thingie 3</a><td>Thingies
</td></td><td align="right">\\\\\\\$3.30</td>
</tr>
<tr>
<td><a href="product.jsp?prodid=7">Thingie 4</a><td>Thingies
</td></td><td align="right">\\\\\\\$3.50</td>
</tr>
</table></center>
```

Inspector panel showing Request attributes, Request query parameters, Request body parameters, Request cookies, Request headers, and Response headers.

Bottom status bar: 3,284 bytes | 1,005 mil | Memory: 335.9 MB

Conclusion:

This assessment demonstrates that even intentionally vulnerable or training applications can present high risk vulnerabilities. For GRC purposes, understanding the QWASP Top 10 is essential for validating security controls, assessing risk, and improving development practice.