Web Application Security Testing Report

Application Tested: OWASP Juice Shop

Tool Used: OWASP ZAP

Date of Testing: 3 Novmber2025

Test Conducted By: P.Nagesh

Introduction

In the rapidly evolving digital ecosystem, web applications have become integral to every business and individual activity — from online banking and e-commerce to education and healthcare. However, this increasing reliance has simultaneously expanded the attack surface, making web applications one of the most frequent targets for cyberattacks. Threat actors continuously exploit vulnerabilities such as SQL injection, cross-site scripting (XSS), authentication flaws, and insecure configurations to gain unauthorized access, steal sensitive data, or disrupt services.

To address these challenges, cybersecurity professionals must develop a deep understanding of web application vulnerabilities and the corresponding defense mechanisms. This project focuses on the security assessment of the OWASP Juice Shop, a purposely vulnerable web application designed for learning and practicing ethical hacking techniques in a controlled environment

Objective of the Assessment

- To replicate realistic cyberattack **scenarios** through the application of professional ethical hacking practices and controlled penetration testing techniques.
- To discover and analyze existing vulnerabilities within the target web application that could potentially be exploited by malicious actors.
- To categorize and prioritize the identified issues based on their severity and alignment with the OWASP Top 10 Web Application Security Risks.
- To develop and recommend actionable security measures aimed at mitigating these vulnerabilities and enhancing the overall robustness of the application's security posture.

Scope of the Assessment

Target Application: OWASP Juice Shop (Local Setup)

Tools Used: OWASP ZAP (Zed Attack Proxy)

Vulnerabilities Tested: Injection flaws, security misconfigurations, information disclosures,etc.

Methodology

The security assessment was carried out using a structured and systematic penetration testing methodology to ensure accurate, repeatable, and verifiable results. Each phase was designed to simulate real-world testing practices while maintaining an ethical and controlled testing environment.

- **Setup & Environment Preparation**
  - Deployed the **OWASP Juice Shop** application locally using a secure test environment.
  - Configured the browser and network settings to route all HTTP/S traffic through the **OWASP ZAP** proxy for inspection.
- **Reconnaissance Phase**
  - Conducted an initial exploration of the application to identify available endpoints, forms, and input vectors.
  - Collected preliminary information about server responses, technologies used, and exposed resources.
- **Passive Scanning**
  - Monitored and analyzed network traffic through OWASP ZAP without sending any active payloads.
  - Identified informational findings such as missing HTTP headers, information disclosure, and caching misconfigurations.
- **Active Scanning & Manual Verification**
  - Executed targeted active scans to emulate attack patterns like **Cross-Site Scripting (XSS)**, **Cross-Site Request Forgery (CSRF)**, and **Header Injection** vulnerabilities.
  - Verified high-confidence results manually to confirm exploitability and eliminate false positives.
- **Documentation & Analysis**
  - Captured relevant evidence including screenshots, scan logs, and tool outputs for each identified issue.
  - Classified each vulnerability according to its **risk level** and mapped it to the **OWASP Top 10** categories.
  - Compiled findings into a detailed report along with appropriate mitigation recommendations.

Tools Used

OWASP Juice Shop – A vulnerable web app for practicing penetration testing.

OWASP ZAP – An open-source security tool for finding vulnerabilities in web applications.

(Optionally) Kali Linux – Preloaded with hacking tools, if used in your setup.

Risk Rating Criteria

Each finding is categorized as:

High – Immediate security threat. Likely to be exploited.

Medium – Could be exploited, requires moderate effort

Low – Information leakage or minor misconfiguration.

Informational – Does not pose direct risk but may assist an attacker.

Conclusion

This assessment uncovered multiple medium- and low-risk vulnerabilities within the OWASP Juice Shop, demonstrating how real-world applications can be exposed to similar threats if not properly secured. The findings emphasize the importance of implementing robust security measures such as proper HTTP headers, strong authentication controls, and secure input validation. Applying the recommended remediation steps will help prevent data leaks, unauthorized access, and other potential attacks. This project reinforced the significance of proactive vulnerability assessment and secure coding practices in building resilient web applications

Detailed Findings

Content Security Policy (CSP) Header Not Set

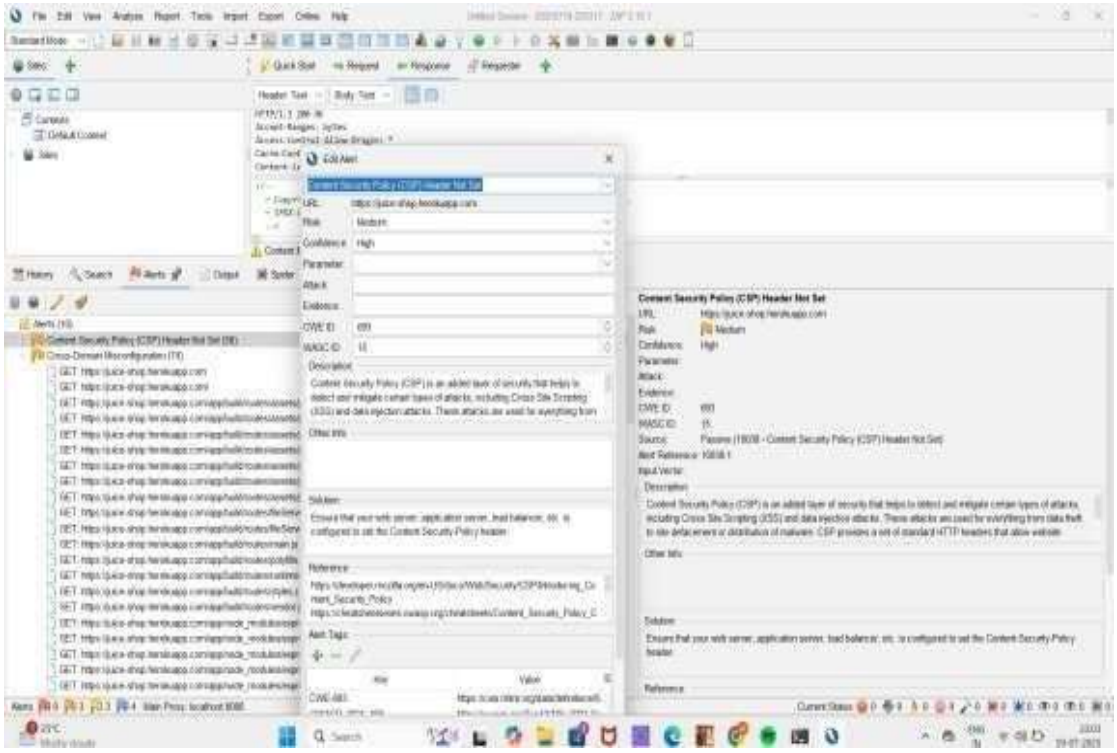
Risk Level: Medium

Description: The Content Security Policy header is missing. This makes the application more vulnerable to Cross-Site Scripting (XSS) and data injection attacks.

Explanation:
The absence of a CSP allows the browser to load resources from any origin, which significantly increases the risk of Cross-Site Scripting (XSS) and data injection attacks. Malicious actors can exploit this misconfiguration to inject harmful JavaScript, manipulate content, or exfiltrate sensitive information such as cookies and session tokens

Content-Security-Policy: default-src 'self'; script-src 'self'; style-src 'self';

Impact:
→ Increases exposure to **XSS** attacks and malicious script execution.
→ Compromises user session integrity and potentially exposes sensitive data.
→Reduces the overall trustworthiness and resilience of the application.
OWASP Mapping: A5 – Security Misconfiguration



Cross-Domain Misconfiguration

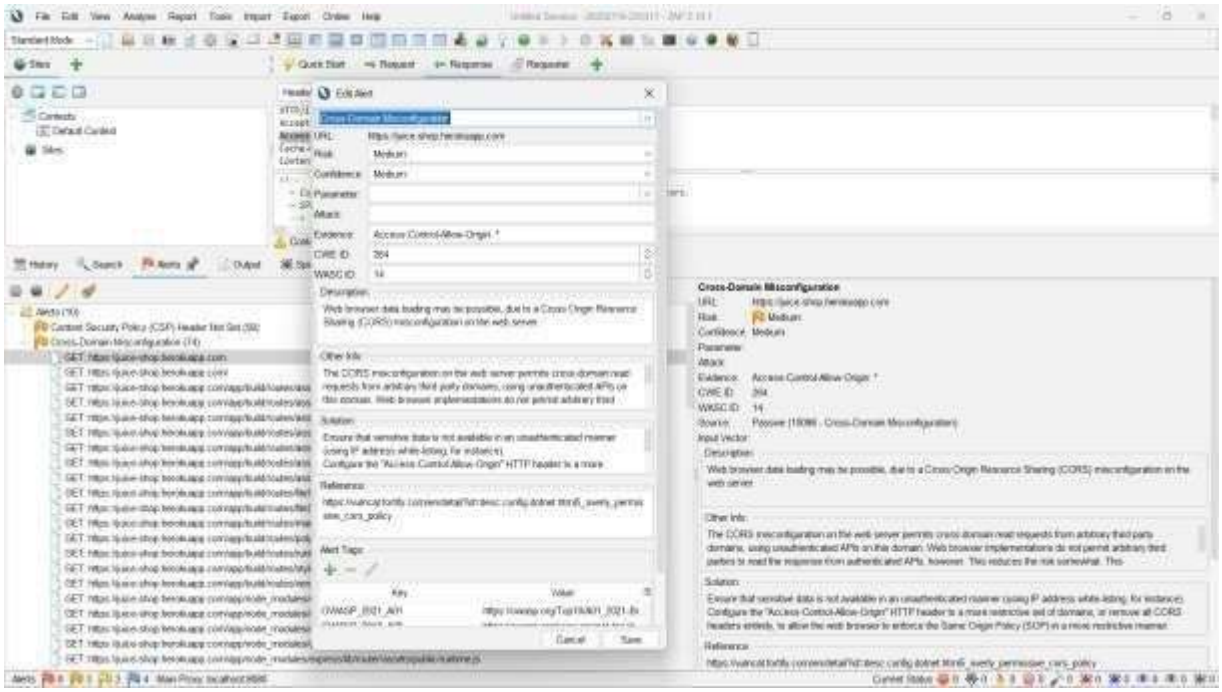
Risk Level: Medium

Description: The application allows resources to be shared across different domains, which may lead to Cross-Site Request Forgery (CSRF) or data leakage issues.

Explanation:
The application allows resources to be accessed from other domains without proper control, which is a common misconfiguration.

Impact:
Enables attackers to perform **Cross-Site Request Forgery (CSRF)** or data leakage.

OWASP Mapping: A8 – Cross-Site Request Forgery (CSRF)



Cross-Domain JavaScript Source File Inclusion

Risk Level: Medium

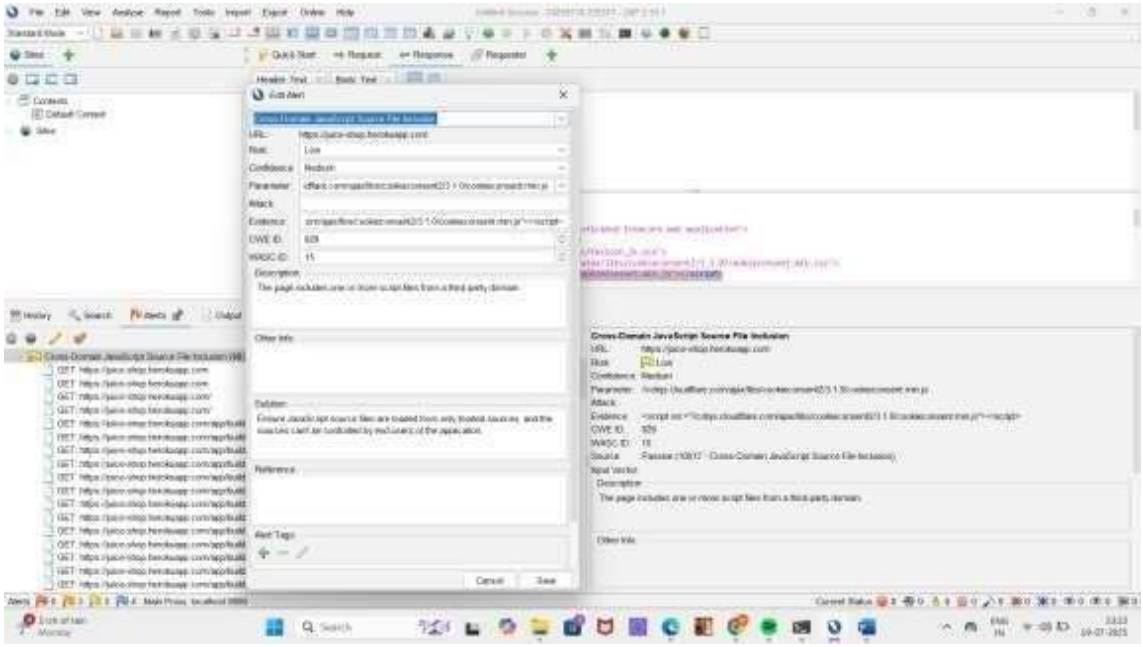
Description: JavaScript files are included from different domains. This can lead to injection of malicious scripts and potentially compromise application security.

Explanation:
JavaScript files are included from untrusted or unknown external sources.

Impact:

Could result in the injection of malicious code and compromise the application.

OWASP Mapping: A1 – Injection



Strict-Transport-Security Header Not Set

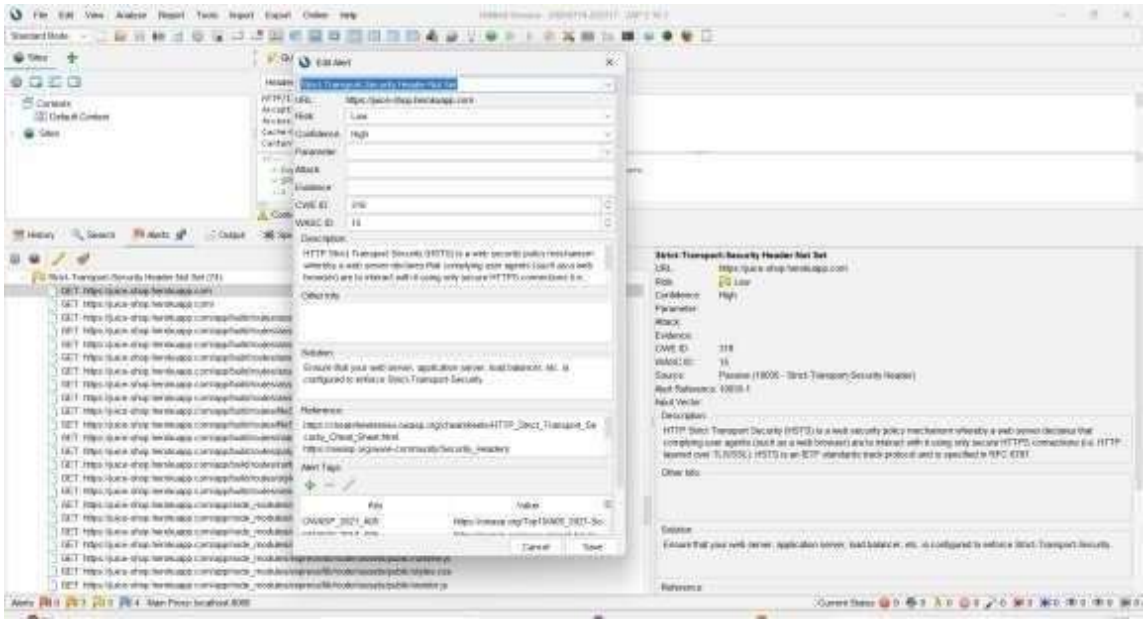
Risk Level: Medium

Description: Missing HSTS header allows connections over insecure HTTP. This exposes users to man-in-the-middle (MITM) attacks.

Explanation:
The application doesn't enforce HTTPS using the Strict-Transport-Security (HSTS) header.

Impact:
Leaves users open to **Man-in-the-Middle (MITM)** attacks over unsecured networks.

OWASP Mapping: A5 – Security Misconfiguration



Timestamp Disclosure - Unix

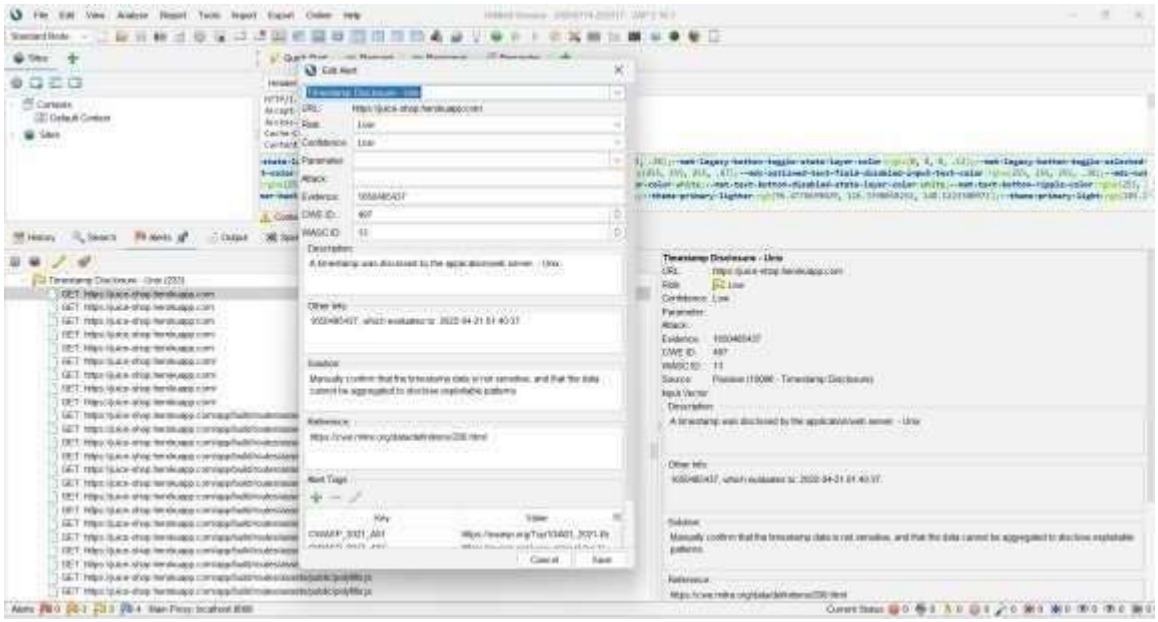
Risk Level: Low

Description: Timestamps disclosed in HTTP responses may help attackers to determine server software versions or other information useful in targeted attacks.

Explanation:
Unix timestamps are present in responses or URLs, potentially revealing information about backend systems.

Impact:
Helps attackers profile the application, servers, or session generation patterns.

OWASP Mapping: A6 – Sensitive Data Exposure



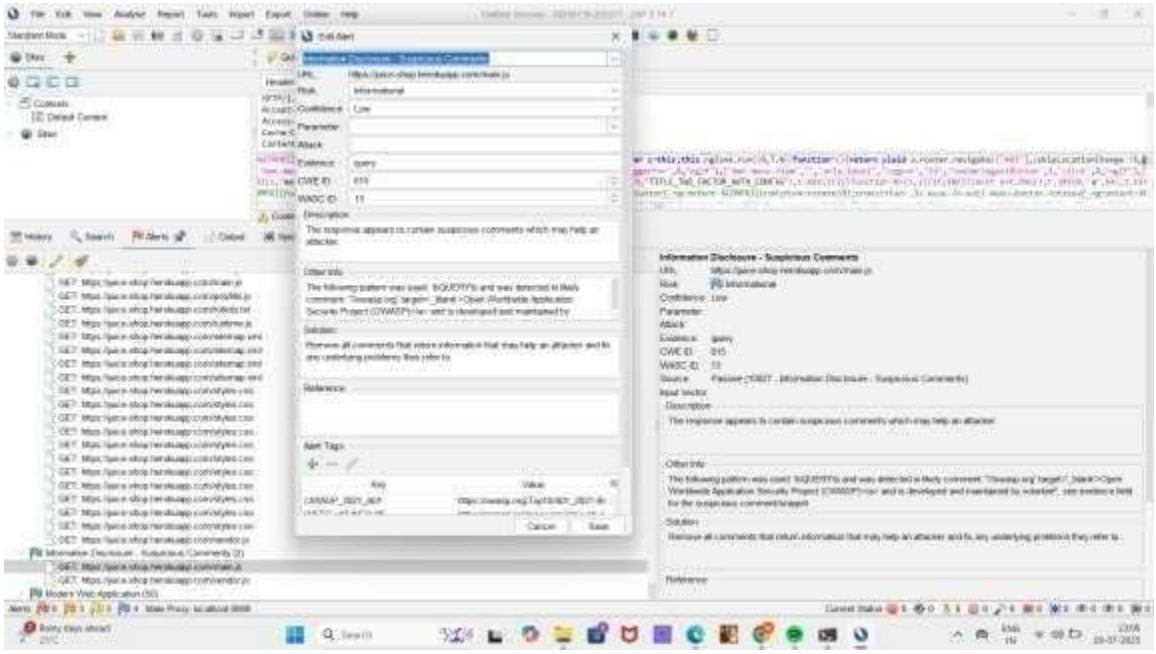
Information Disclosure - Suspicious Comments

Risk Level: Low

Description: Source code contains suspicious or debug-related comments. These might reveal sensitive internal information to an attacker.

Explanation:
Developer comments are visible in the source code.
Impact:
These may reveal debugging information, credentials, or internal logic useful to attackers

OWASP Mapping: A6 – Sensitive D Exposure



ata

Modern Web Application

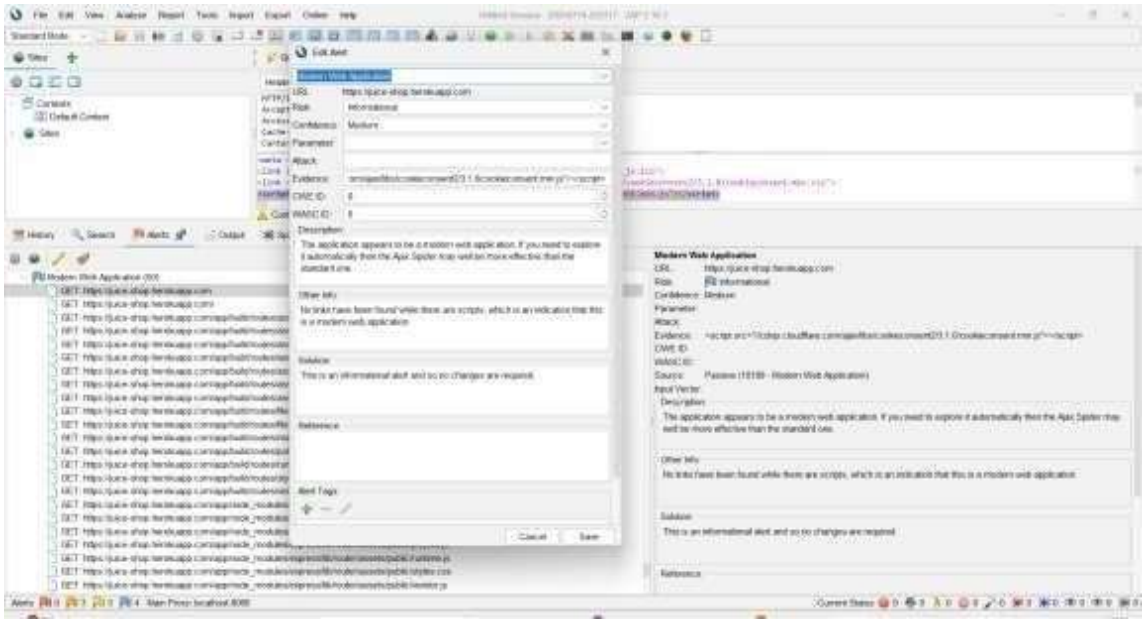
Risk Level: Informational

Description: General detection of a modern web application. This is not necessarily a vulnerability but useful information for testers.

Explanation:
Indicates the application is built using a modern frontend framework.

Impact:
No direct security impact. Helps understand the technology stack used.

OWASP Mapping: Not Applicable



Hidden File Found

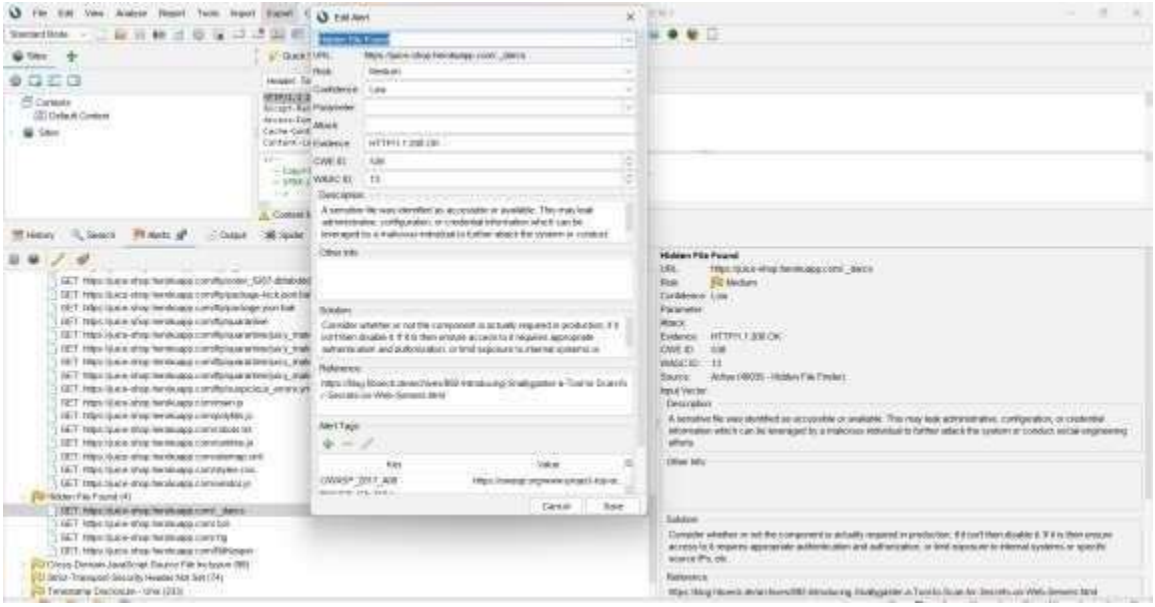
Risk Level: Medium

Description: Sensitive hidden file was found accessible (_darcs). This may leak credentials or configuration data.

Explanation:
A hidden file (_darcs) was discovered, which might contain configuration or history.

Impact:
Could leak credentials, source code, or internal configuration.

OWASP Mapping: A5 – Security Misconfiguration



User Agent Fuzzer

Risk Level: Medium

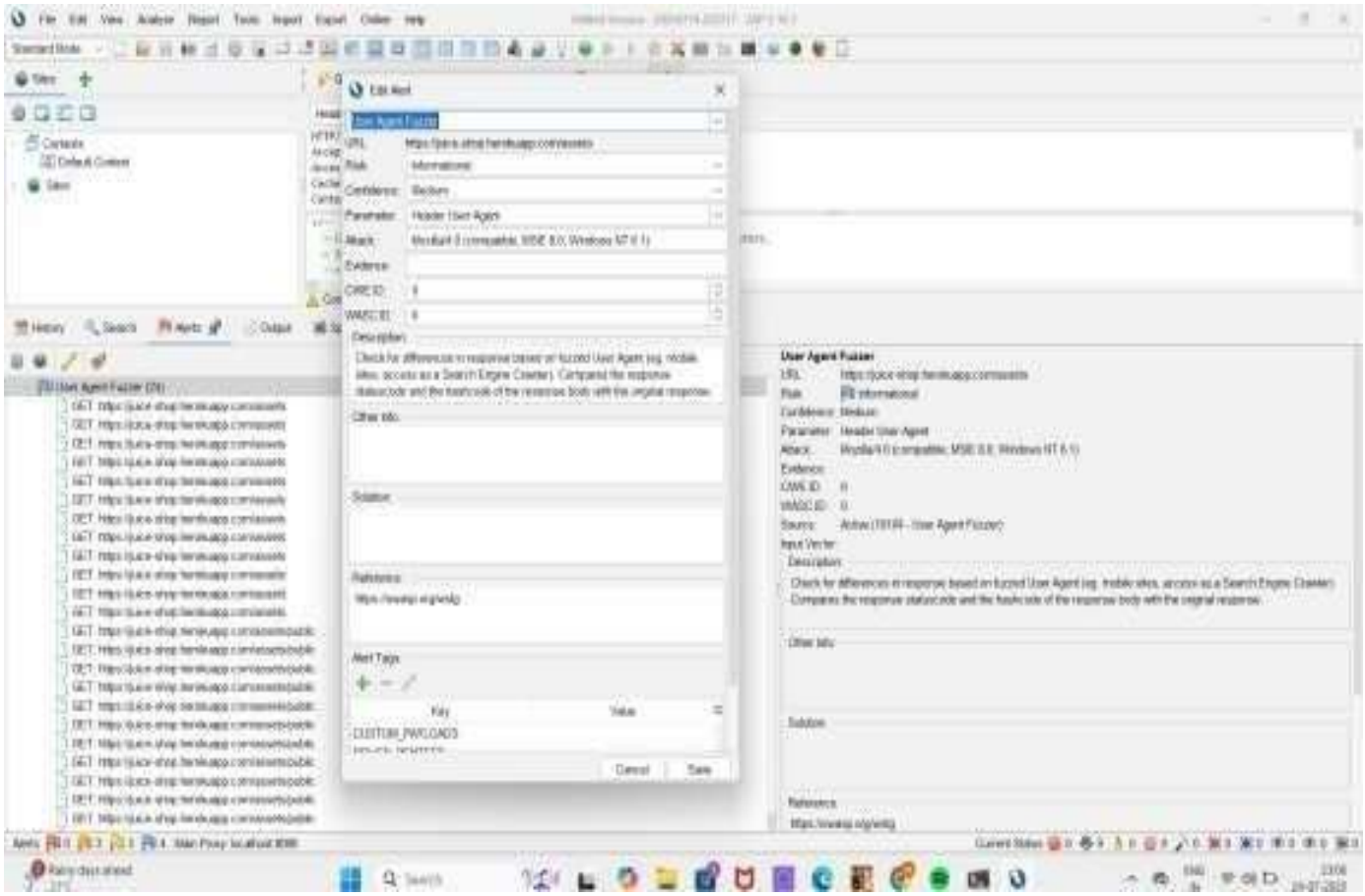
Description: Check for differences in response based on fuzzed User Agent (eg. mobile sites, access as a Search Engine Crawler). Compares the response statuscode and the hashcode of the response body with the original response.

Explanation:
The application responded differently when accessed with modified User-Agent headers.

Impact:

Can be used to fingerprint the application or bypass features.

OWASP Mapping: A9 – Insufficient Logging & Monitoring



Re-examine Cache-control Directives

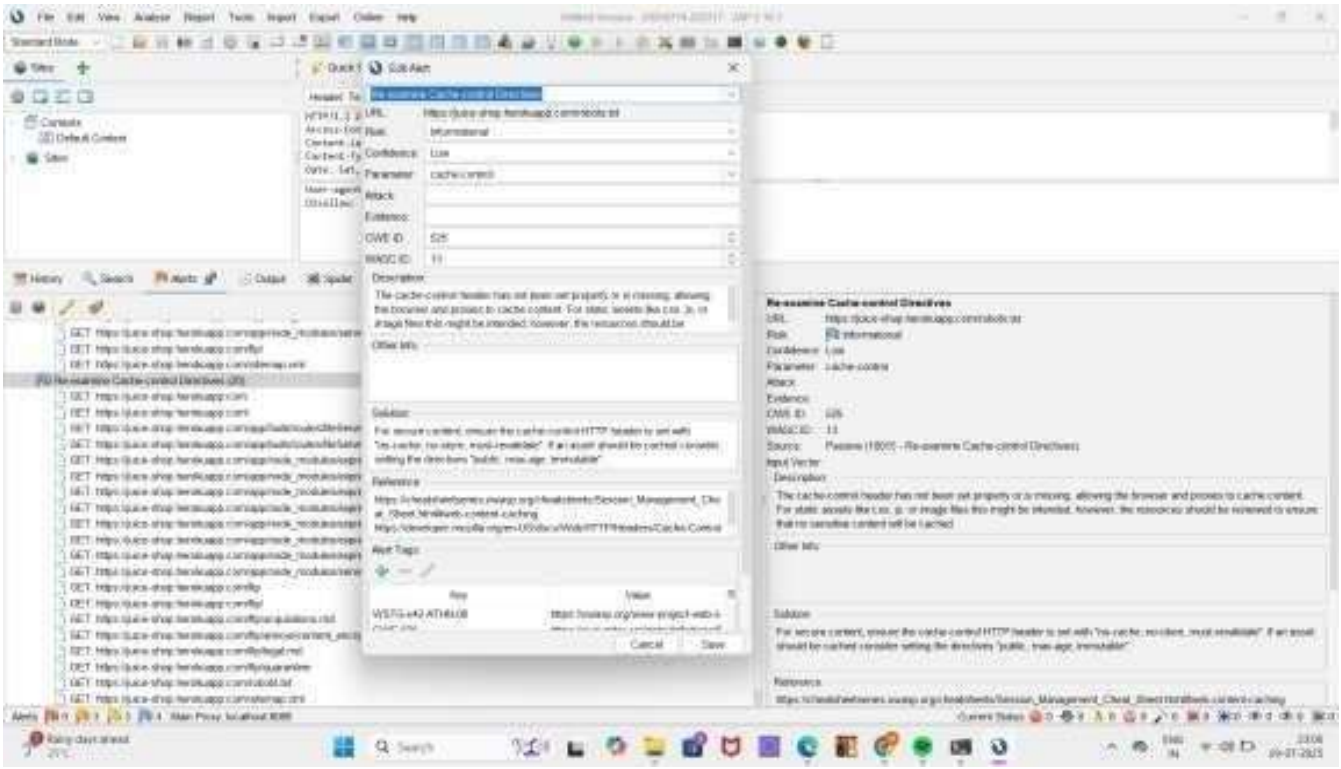
Risk Level:Low

Description: The cache-control header has not been set properly or is missing, allowing the browser and proxies to cache content. For static assets like css, js, or image files this might be intended, however, the resources should be reviewed to ensure that no sensitive content will be cached.

Explanation:
Improper or missing Cache-Control headers allow sensitive content to be cached.

Impact:
Sensitive data might be stored in shared/public caches (e.g., proxy servers, browsers).

OWASP Mapping: A5 – Security Misconfiguration



Vulnerability: SQL Injection – Authentication Bypass

Location:

Login Page – `/#/login`

Description:

An SQL Injection vulnerability was discovered on the login page of the OWASP Juice Shop application. The application fails to properly validate and sanitize user inputs before incorporating them into SQL queries. As a result, an attacker can inject malicious SQL code into the input fields to alter the logic of the SQL query.

Attack Vector:

Payload used:

```
vbnet
Copy code
' OR 1=1--
```

- Injected into the **email/username field**
- Any value (e.g., `abc`) was entered as the password
- Upon submitting the form, the application logged the user in successfully

---

Impact:

- Bypass of login authentication
- Unauthorized access to user or administrator accounts
- Potential access to sensitive data
- Increased risk of privilege escalation and data breaches

OWASP Top 10 Mapping:

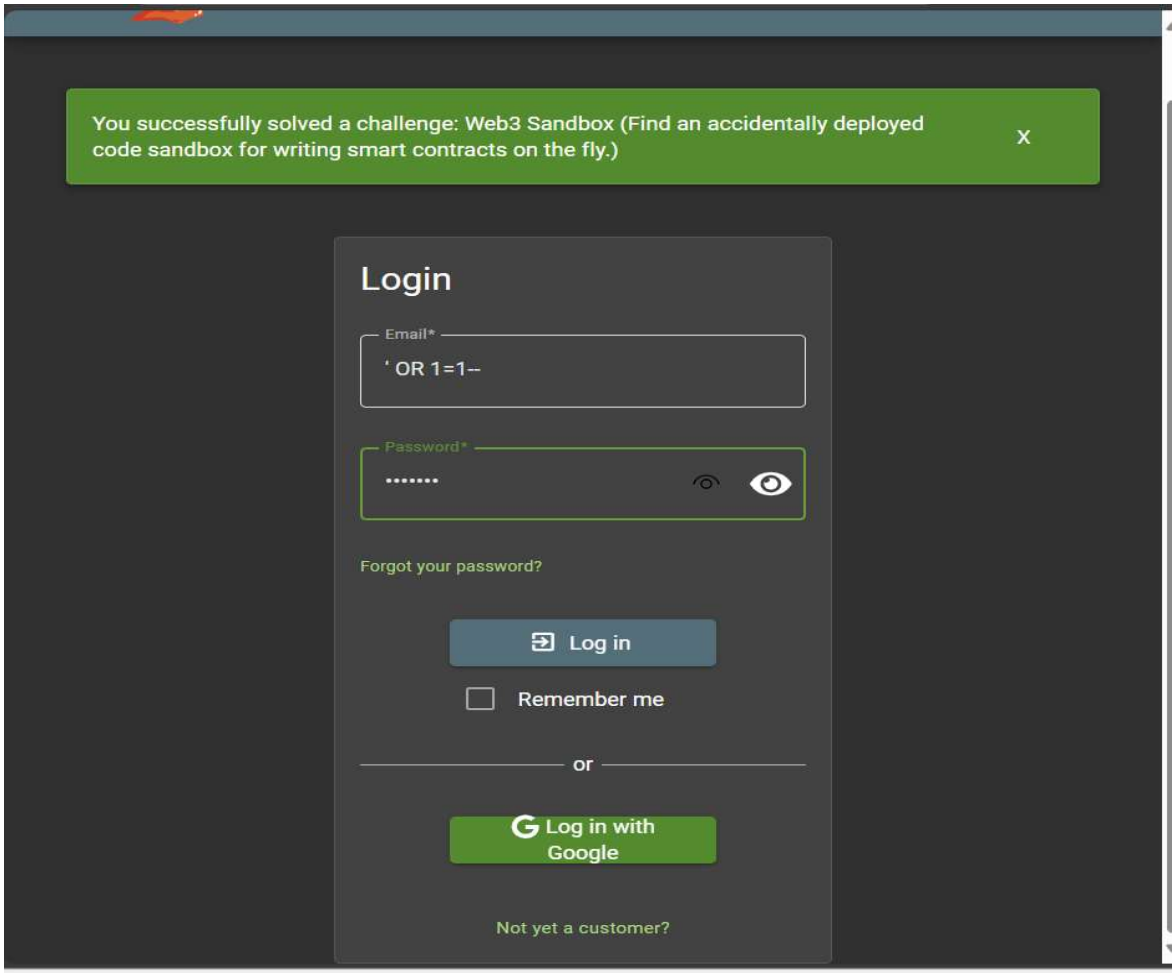
- **A03:2021 – Injection**

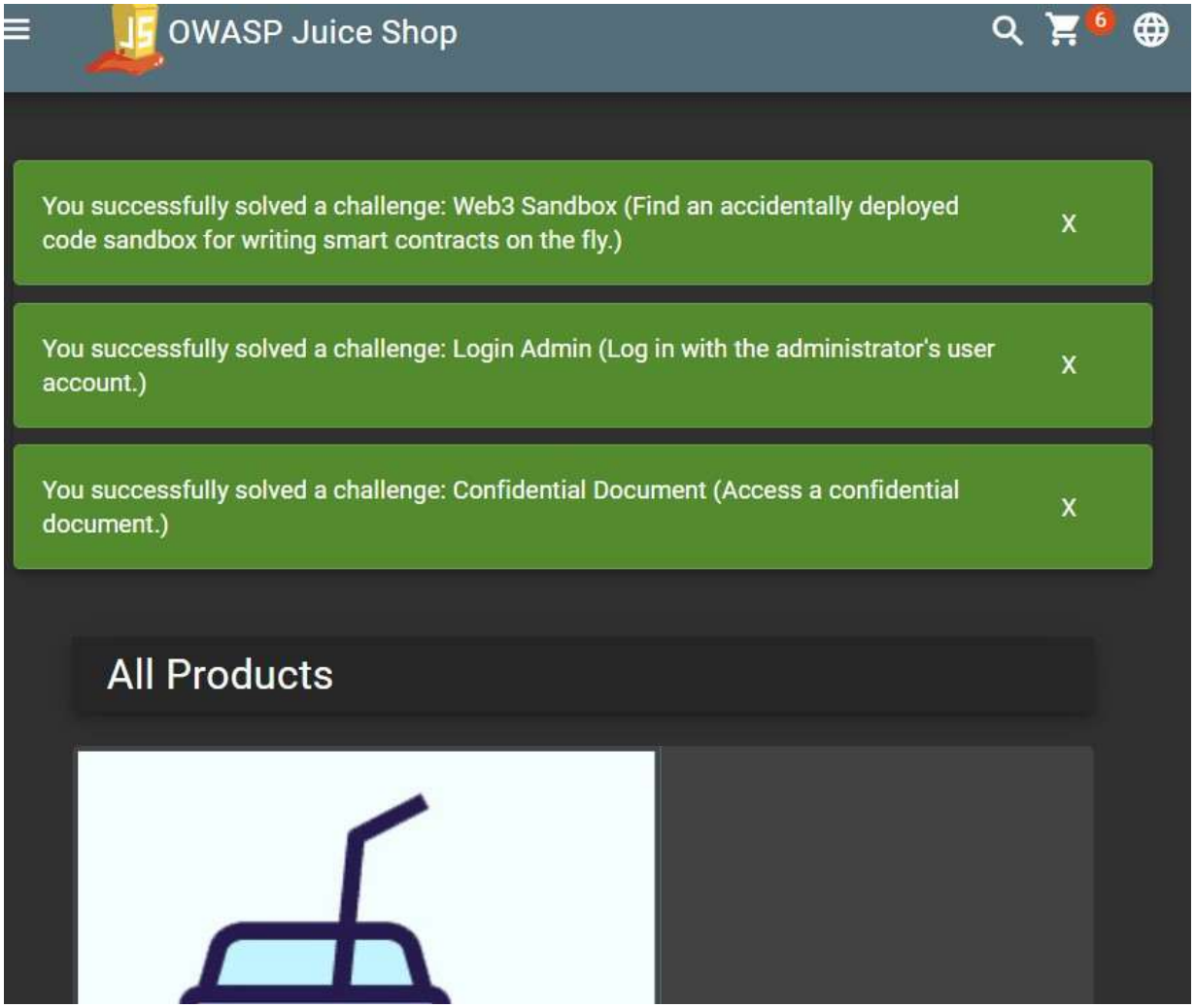
Recommended Remediation:

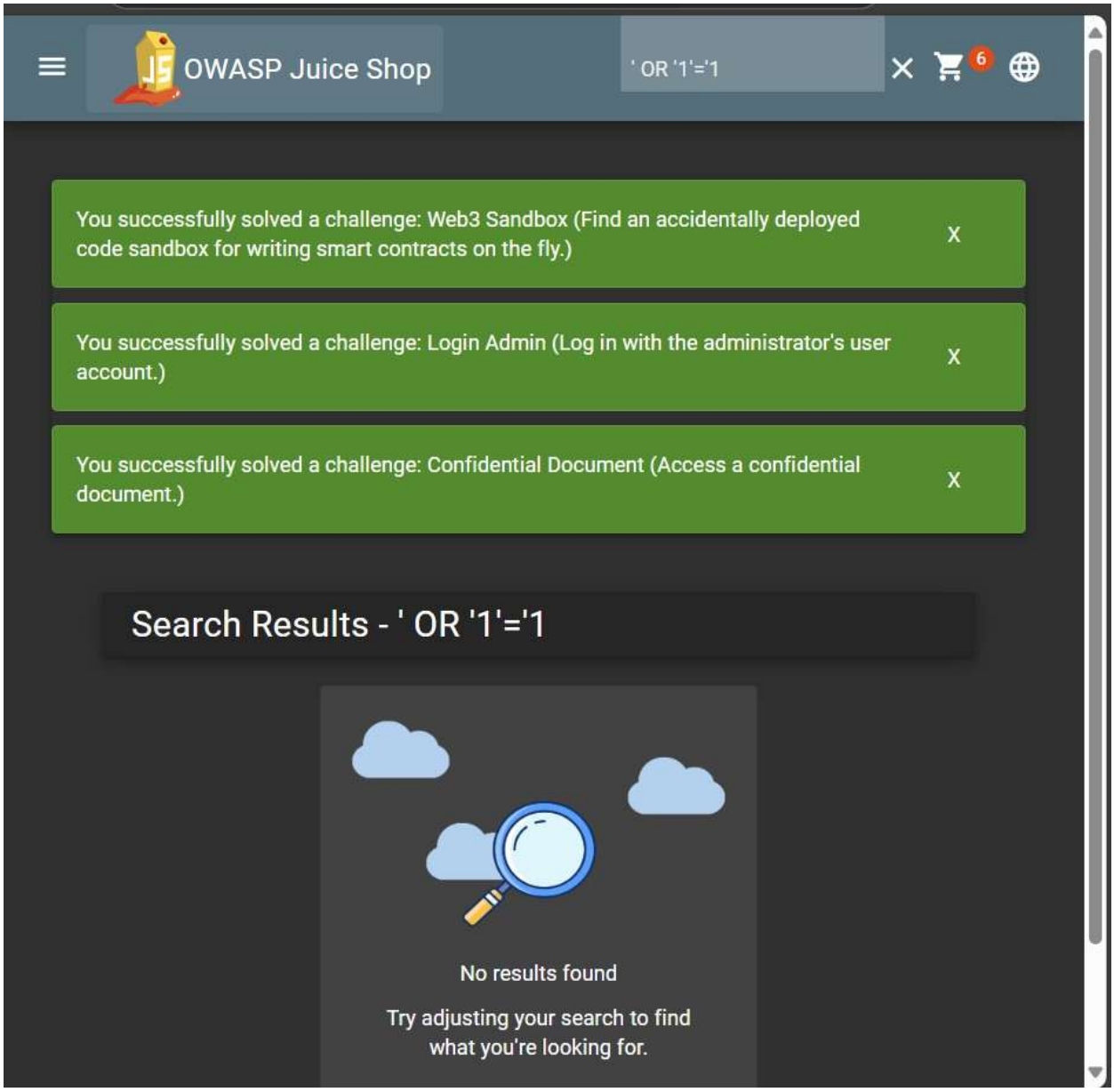
- Use **parameterized queries (prepared statements)** instead of dynamic SQL.
- Validate and sanitize all user inputs on both client-side and server-side.
- Avoid directly incorporating user input into database queries.
- Implement a **Web Application Firewall (WAF)** to monitor and block injection attempts.

Evidence:

- Screenshot of the login form with the injection payload
- Screenshot showing successful login after payload submission




OWASP Juice Shop
6
You successfully solved a challenge: Web3 Sandbox (Find an accidentally deployed code sandbox for writing smart contracts on the fly.)
x
You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)
x
You successfully solved a challenge: Confidential Document (Access a confidential document.)
x
All Products


OWASP Juice Shop
' OR '1'='1
6
You successfully solved a challenge: Web3 Sandbox (Find an accidentally deployed code sandbox for writing smart contracts on the fly.)
X
You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)
X
You successfully solved a challenge: Confidential Document (Access a confidential document.)
X
Search Results - ' OR '1'='1
No results found
Try adjusting your search to find what you're looking for.