PT Report

SuperDuperMarket

Tester name: Michael Liberman
Organization: TDX Arena Certificate Exam
Time testing: 8 hours and 30 minutes

Testing process : The test began with accessing the application using a special account provided by the owner, which didn't require payment details. I explored the web application to find sensitive information. A vulnerability was identified in the way the application processed certain requests. This vulnerability was then used to retrieve sensitive information from the system.

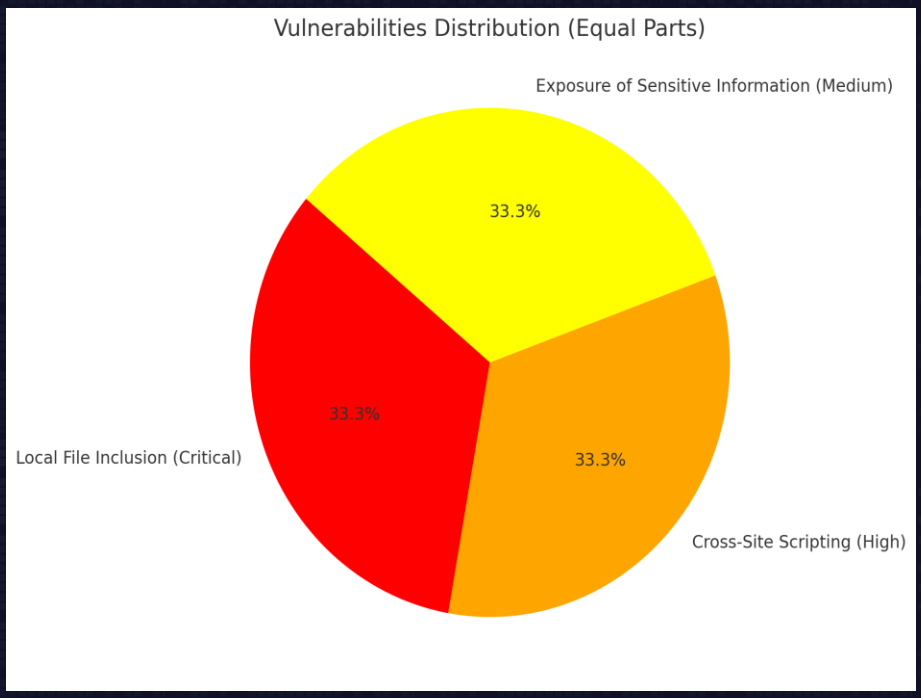
Scoping: None

Executive Summary

During the assessment, weaknesses were found that allowed access to sensitive files and information on the system. These weaknesses could potentially expose important data or be used to carry out harmful actions. The process involved analyzing hidden files and testing how the system handles user input. This highlights the need for better protections to prevent unauthorized access and safeguard sensitive information.

Conclusions

- Local File Inclusion “LFI” (Critical)
- Cross-Site Scripting “XSS” (High)
- Exposure of Sensitive Information (Medium)



Finding Details

VULN-001 Local File Inclusion “LFI” (CRITICAL)

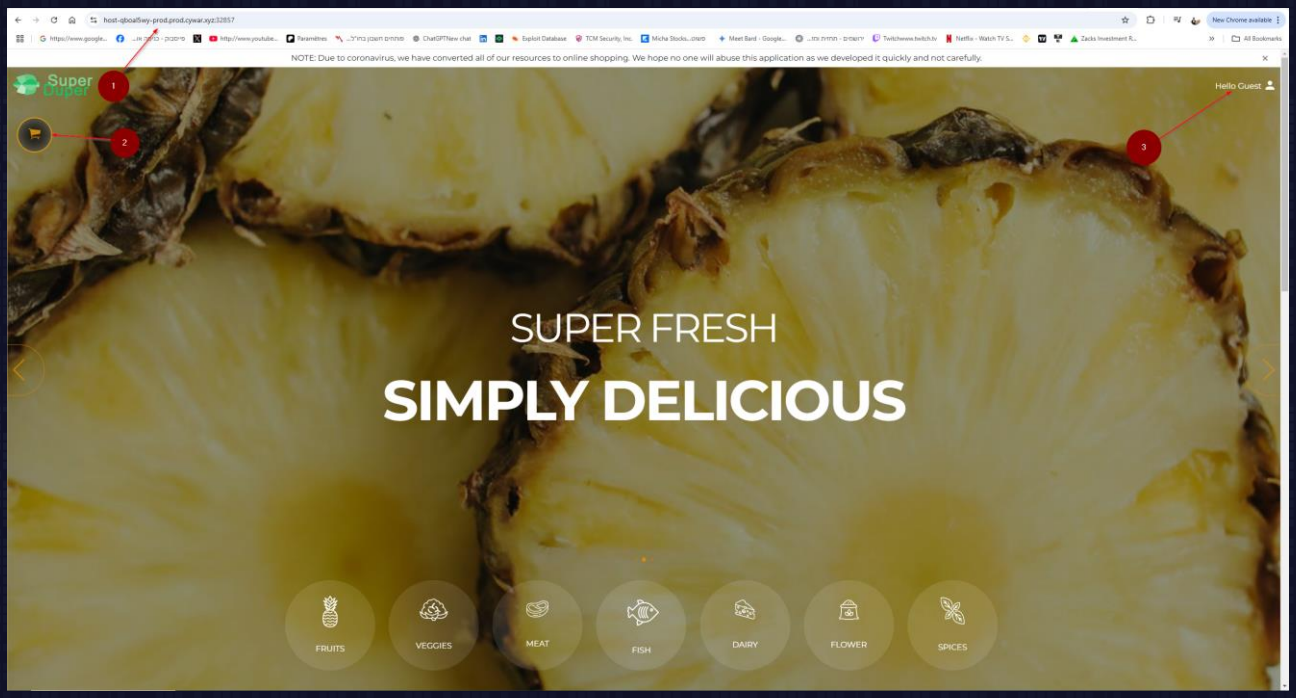
Description:

This vulnerability allows attackers to access and read files stored on the server by manipulating file paths. In some cases, it can lead to executing arbitrary code if further exploited.

Details:

By exploiting this vulnerability, an attacker can access files like system configurations, logs, or sensitive application data stored on the server. This could reveal critical information, such as database credentials, encryption keys, or user data.

Picture1:

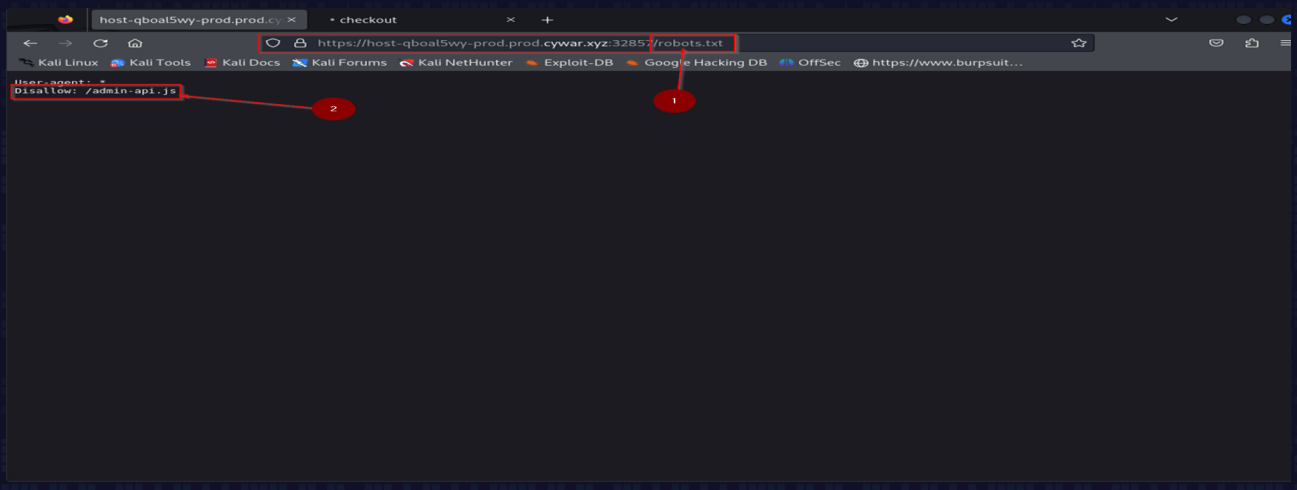


1. The URL at the top of the page shows the web address of the online store.

2. The shopping cart icon allows users to proceed to the checkout page after collecting items they want to purchase.

3. The "Hello Guest" indicator shows that the user is logged in as a recognized guest account.

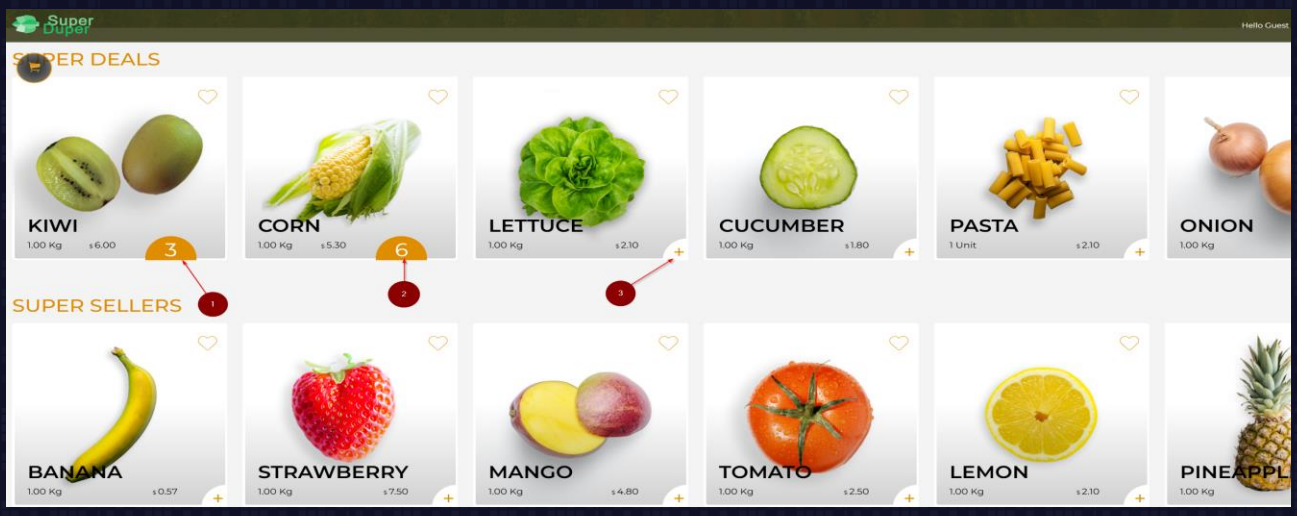
Picture2:



1. The first numbered marker highlights the URL where /robots.txt was appended. This file is commonly used to instruct web crawlers about which parts of the site should not be accessed.

2. The second marker points to the disallowed path /admin-api.js, which may contain sensitive or restricted information. This file could be explored further for additional insights into the system.

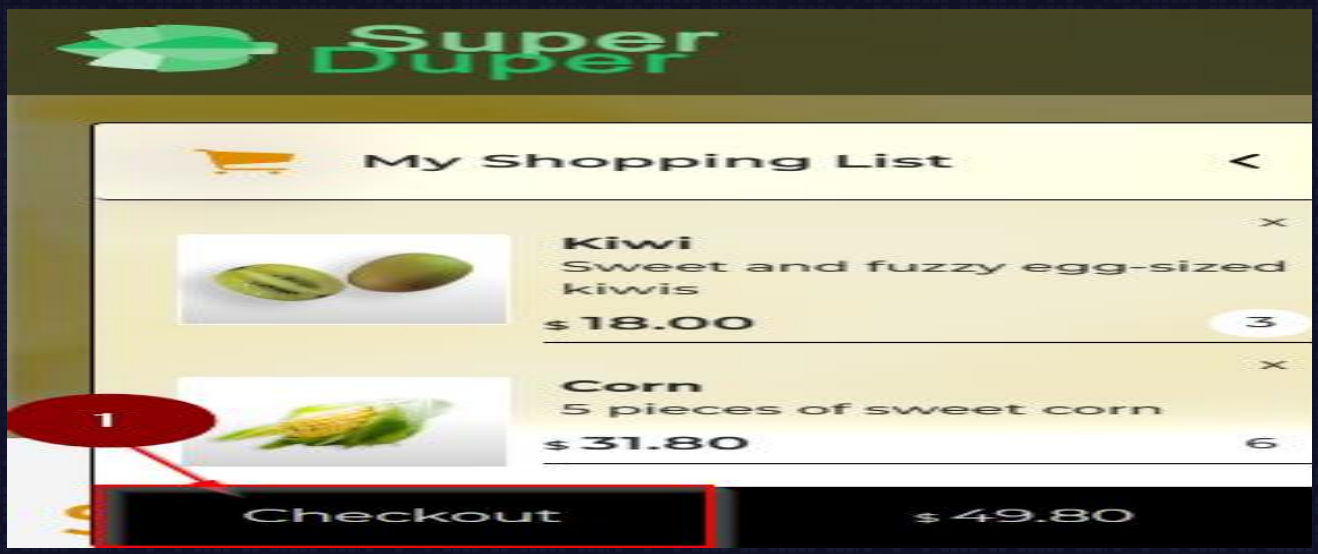
Picture3:



1. The markers (1,2) represent the quantities of products ("KIWI" and "CORN") added to the cart. For each product, the respective quantity is displayed beside it.

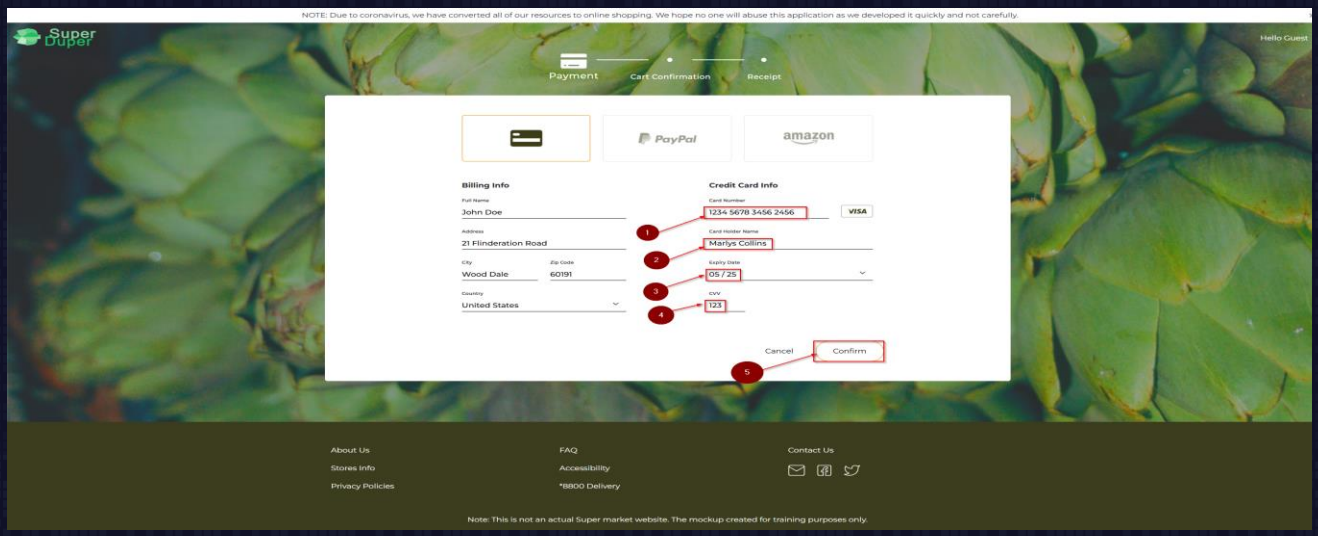
2. The "+" icon indicates an option to add more items to the cart for products not yet selected.

Picture4:



1. The shopping cart dropdown menu shows the list of items added to the cart. It includes details like product names, quantities, and individual prices.
2. The "Checkout" button allows the user to proceed to finalize the purchase.

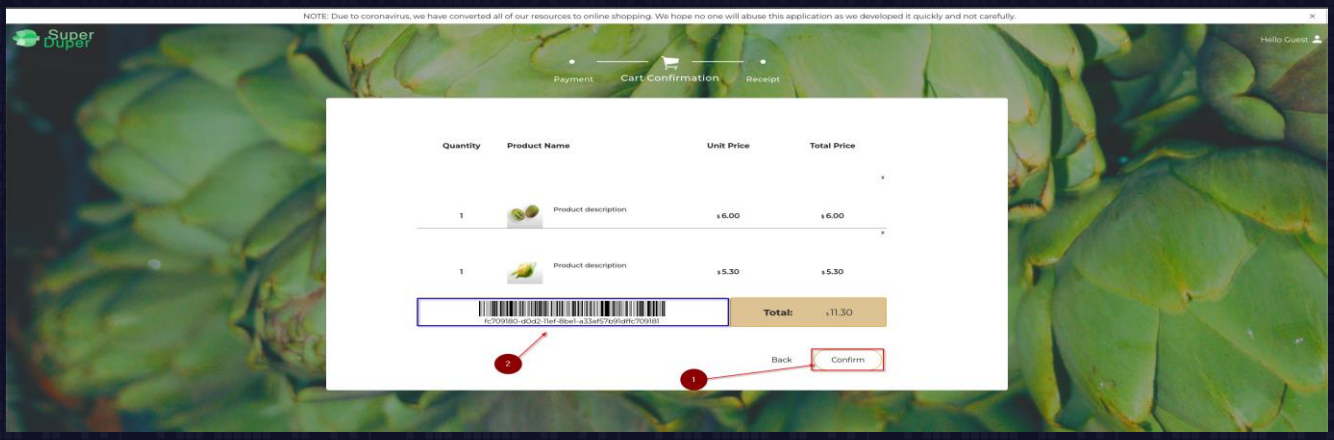
Picture5:



1. The fields display (1-4) the saved payment details of the user, including the cardholder's name ("Marlys Collins"), card number, expiration date ("05/25"), and CVV ("123").

2. The "Confirm" button at the bottom advances to the payment confirmation step.

Picture6:

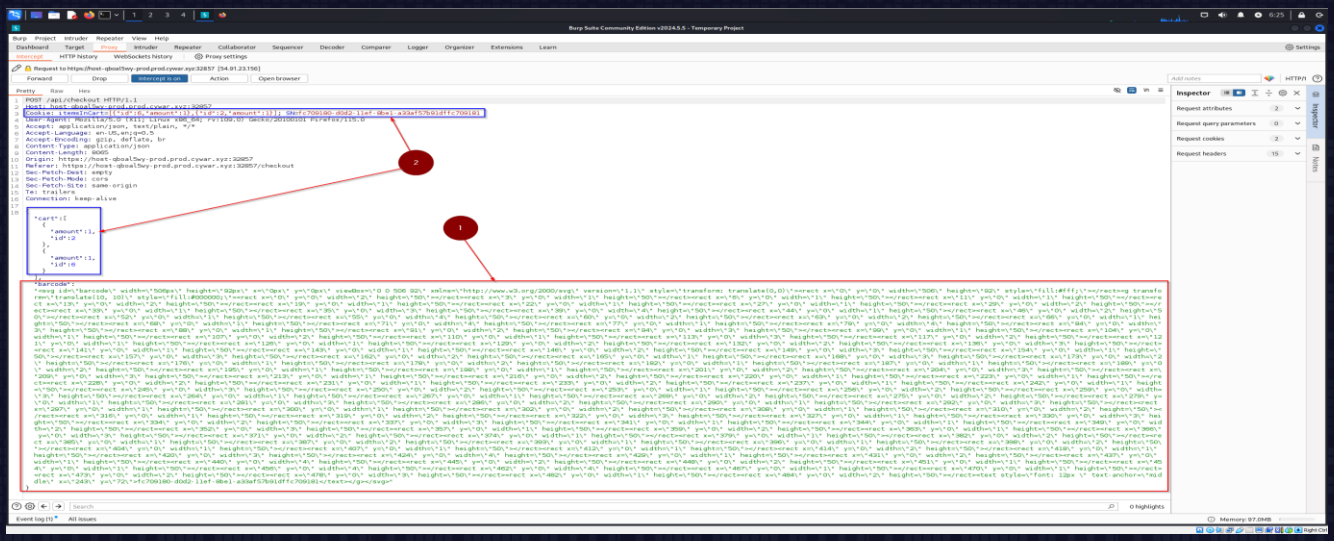


1. A barcode is visible, which could be useful for tracking or further analysis of the transaction.
2. The "Confirm" button finalizes the payment and completes the transaction.

Picture7:

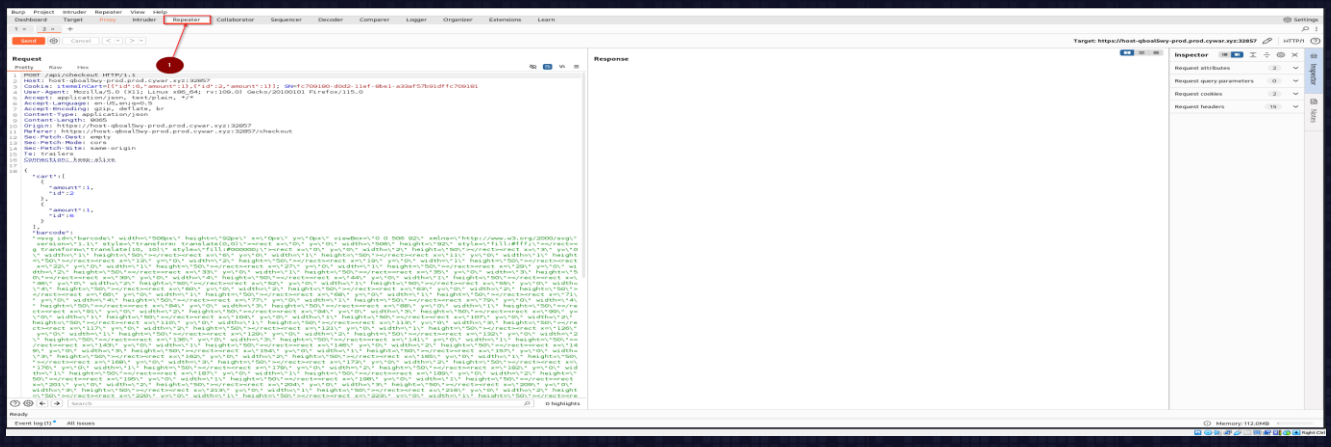
1. The FoxyProxy browser extension is configured to direct traffic through a proxy server on port 8080. This setup allows capturing requests with Burp Suite.

Picture8:



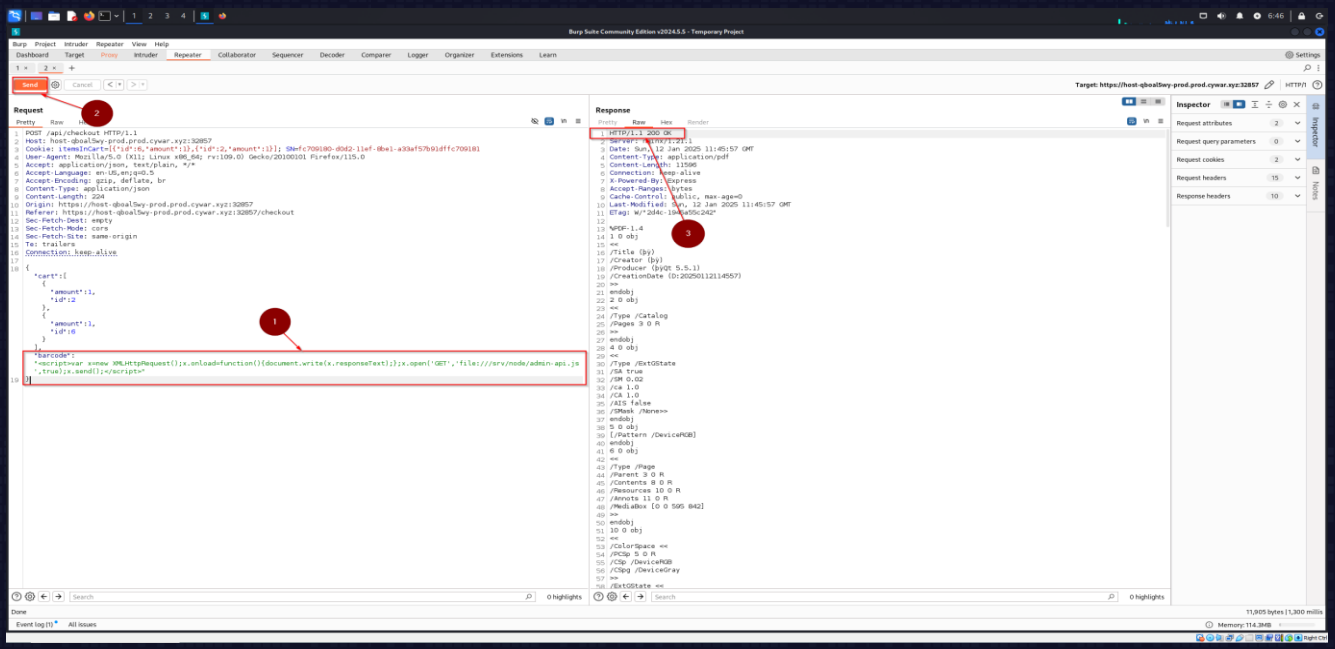
1. The content of the barcode is shown in the intercepted request within Burp Suite. It includes detailed information embedded in the checkout process.
2. Relevant parameters related to the products added to the cart (e.g., product IDs and quantities) are displayed, which could be analyzed for further understanding.

Picture9:



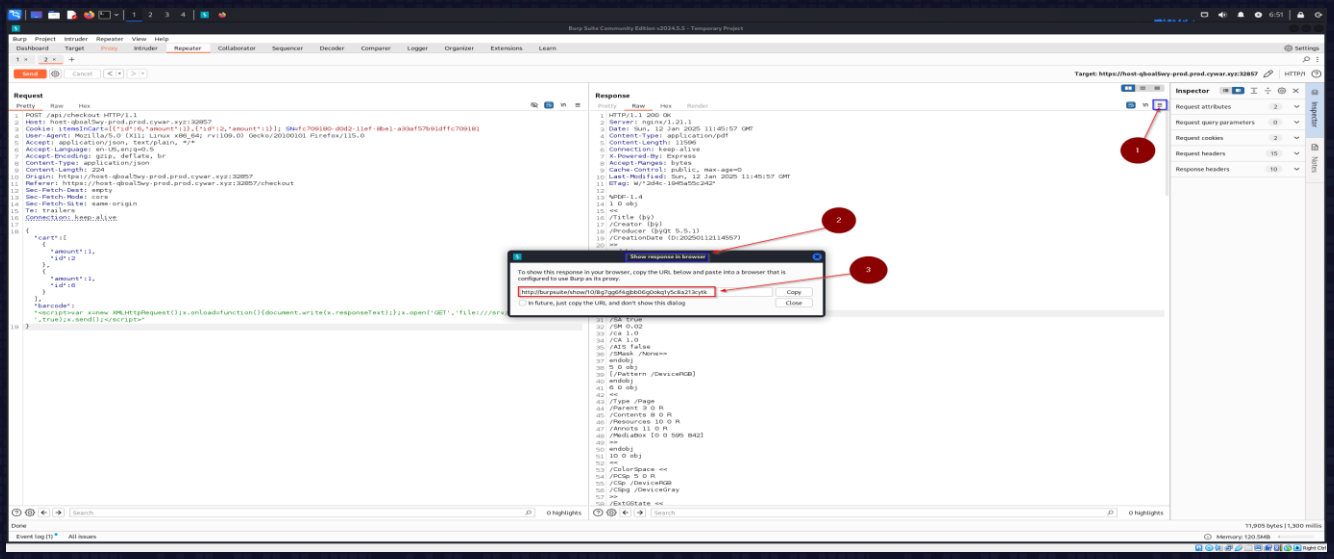
1. The request is transferred to the "Repeater" tool in Burp Suite. This tool allows testing and modifying requests before sending them to the server.

Picture10:



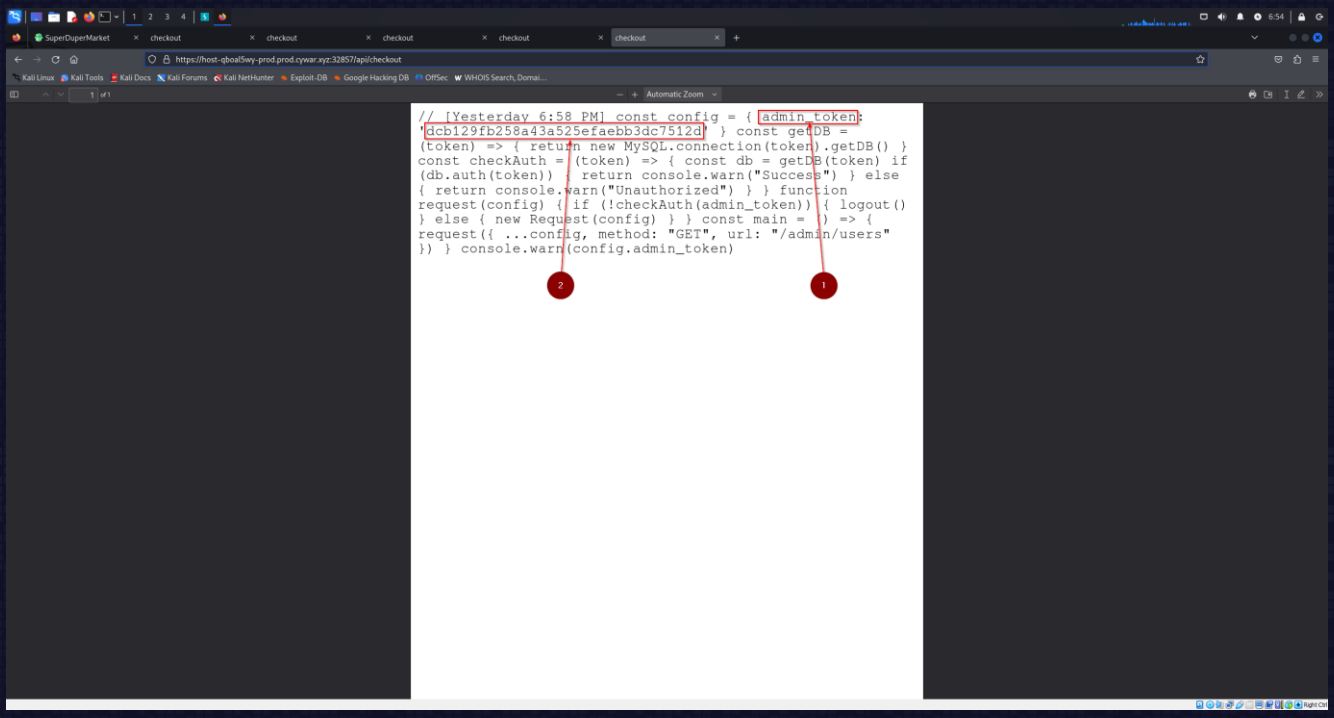
1. An XSS payload is added to the request under the "barcode" parameter. The script aims to execute on the server or client-side to retrieve sensitive data.
2. The server response (HTTP 200) confirms successful processing of the request.
3. The response headers indicate the content is a PDF served directly from the server.

Picture11:



1. The interface shows options for interacting with the server response after executing the XSS payload.
2. The server's response is highlighted, including the URL generated for accessing the resource.
3. The generated link refers to the server's response, providing insight into potential vulnerabilities or data extraction opportunities.

Picture12:



1. The admin_token identifies the admin's access credentials.
2. The token value itself acts as the flag for the challenge.

Recommendations:

Local File Inclusion "LFI" (Critical)

Validate and sanitize user-supplied file paths, allowing access only to files from a predefined whitelist. Avoid dynamic file inclusion and restrict sensitive functions like include() and require(). Harden the server by securing file permissions and disabling unnecessary features. Regularly update software and perform security audits to address vulnerabilities.

VULN-002 Cross-Site Scripting “XSS”(High)

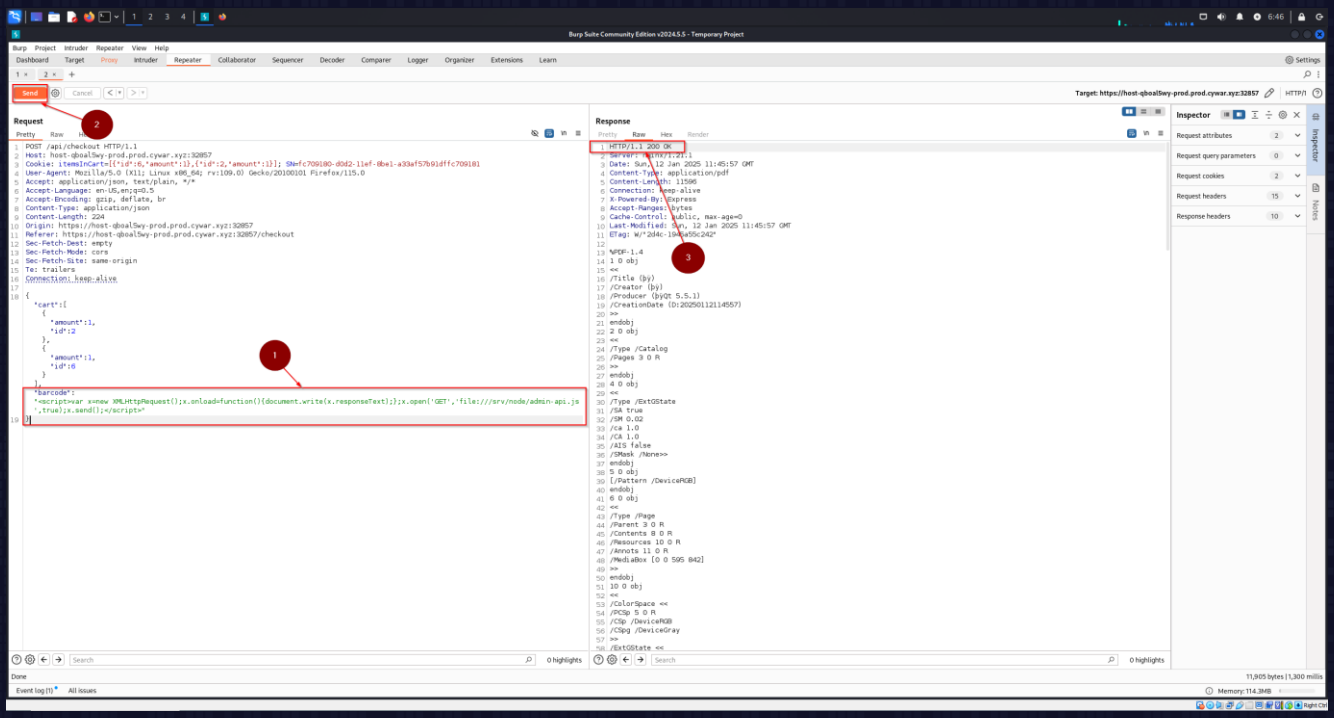
Description:

This occurs when attackers inject malicious scripts into a web application, which are then executed in the browsers of other users. The vulnerability arises due to insufficient input sanitization.

Details:

An attacker can use this to steal user session cookies, redirect users to malicious websites, or manipulate the web application’s behavior. This is especially dangerous if administrative users are targeted.

Picture1:



1. The attacker injects a malicious script into the application through an input field or parameter, which is reflected or stored in the response.
2. The script executes in the victim's browser when they access the vulnerable page, stealing data or altering the page.
3. The stolen information, like session tokens, is sent to the attacker or used to perform unauthorized actions.

Recommendations:

Cross-Site Scripting "XSS"(High)

Validate all user inputs on both the client and server side to ensure they match the expected format (e.g., no special characters or HTML tags). Sanitize inputs by removing or escaping characters like <, >, ', and " to prevent scripts from being injected. Use built-in frameworks or libraries that handle validation and sanitization to reduce human error.

VULN-003 Exposure of Sensitive Information (Medium)

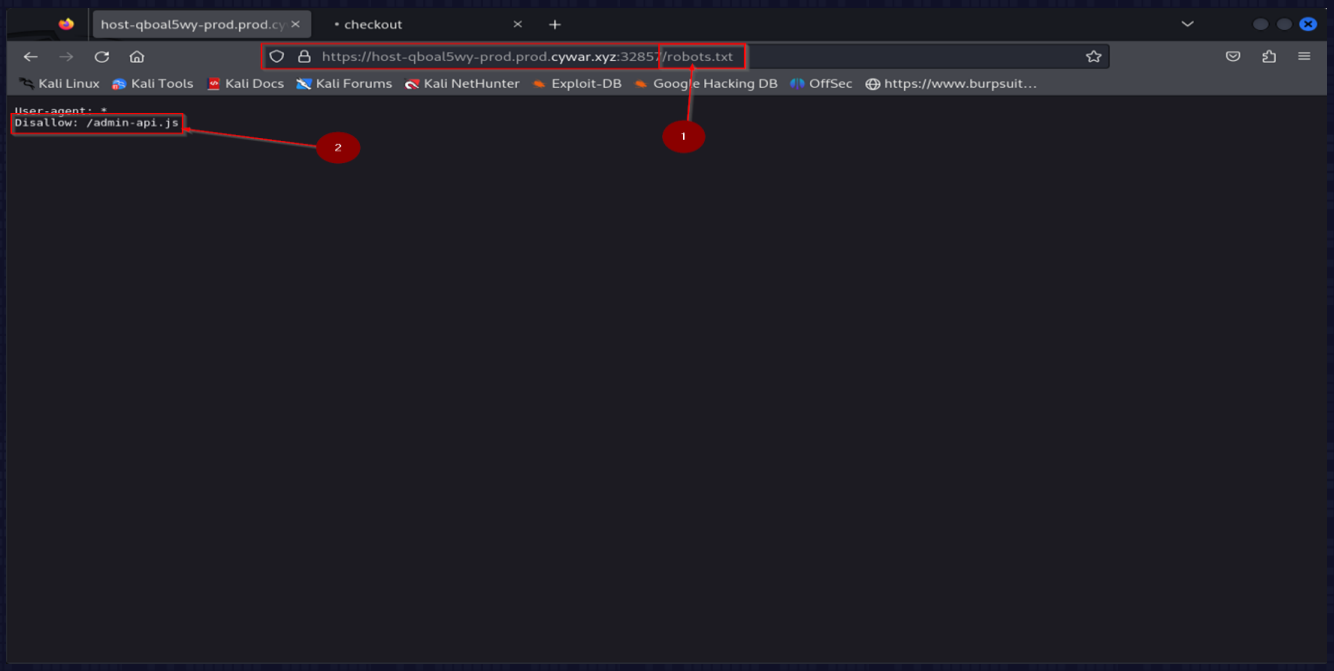
Description:

Important files or information, like system paths or settings, are accidentally made available to attackers. This can happen due to weak security settings, misconfigurations, or files like robots.txt.

Details:

The leaked information helps attackers understand the system's structure, locate important files, or access internal APIs. For example, robots.txt might list restricted areas, such as admin-api.js, that attackers can exploit.

Picture1:



1. The robots.txt file exposes restricted paths, providing attackers with a roadmap to sensitive resources.
2. The /admin-api.js path may contain sensitive administrative data or functionality, increasing the risk of exploitation.

Recommendations:

Exposure of Sensitive Information (Medium)

Avoid exposing sensitive data in publicly accessible files like robots.txt or error messages. Limit access to sensitive endpoints using authentication and authorization mechanisms. Encrypt sensitive data at rest and in transit, and regularly review code and configurations to prevent inadvertent exposure.