# Security Vulnerability in Red Roosters Online Ordering System \* CVSS 5.3

By [Jack Weller](https://github.com/jackwellerreal)

Please note that this exploit works as of making this document (15/12/2024)

On December 10, 2024, I discovered a vulnerability in Red Rooster's online ordering system. This issue allows customers to place orders without payment, whether for pickup or delivery. The vulnerability arises from the ability to modify the "cart" variable in session storage to set the total price to $0. This manipulation causes the website to mistakenly recognize the transaction as a special offer or voucher, bypassing the payment page and allowing the order to be processed without requiring payment details.

This is a clear example of client-side validation being relied upon without sufficient server-side checks, leaving the system vulnerable to exploitation. If this issue is not addressed, there is a risk that others may discover and abuse the exploit, leading to significant financial losses for franchise owners and the company as a whole.

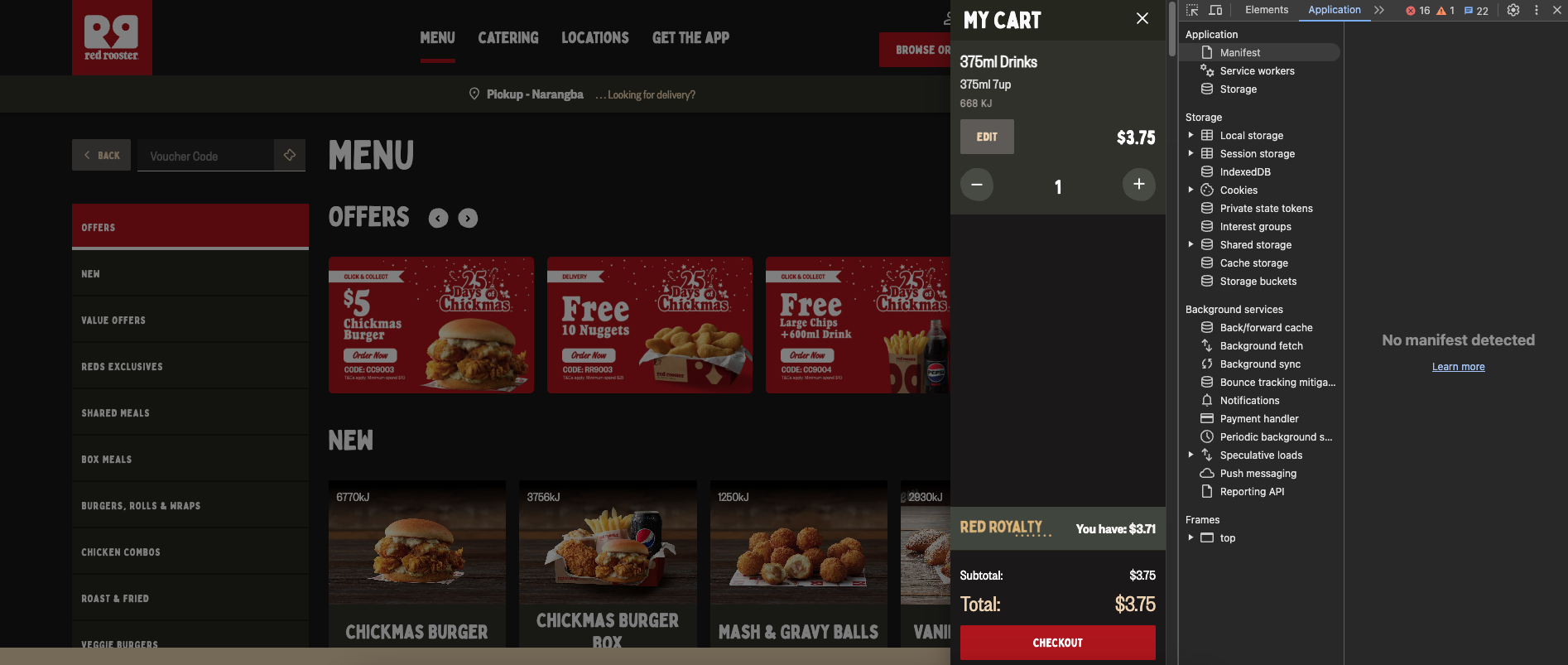
## **Steps to Reproduce**:

1. Open the Red Rooster website and add items to the cart

A screenshot of a menu

Description automatically generated

1. Open the developer tools built into the web browser and navigate to “Application” for Chrome and Edge, or “Storage” for Firefox



1. Click on “Session Storage” and locate the URL for the Red Rooster website

A screenshot of a food store

Description automatically generated

1. Copy the cart variable

A screenshot of a food store

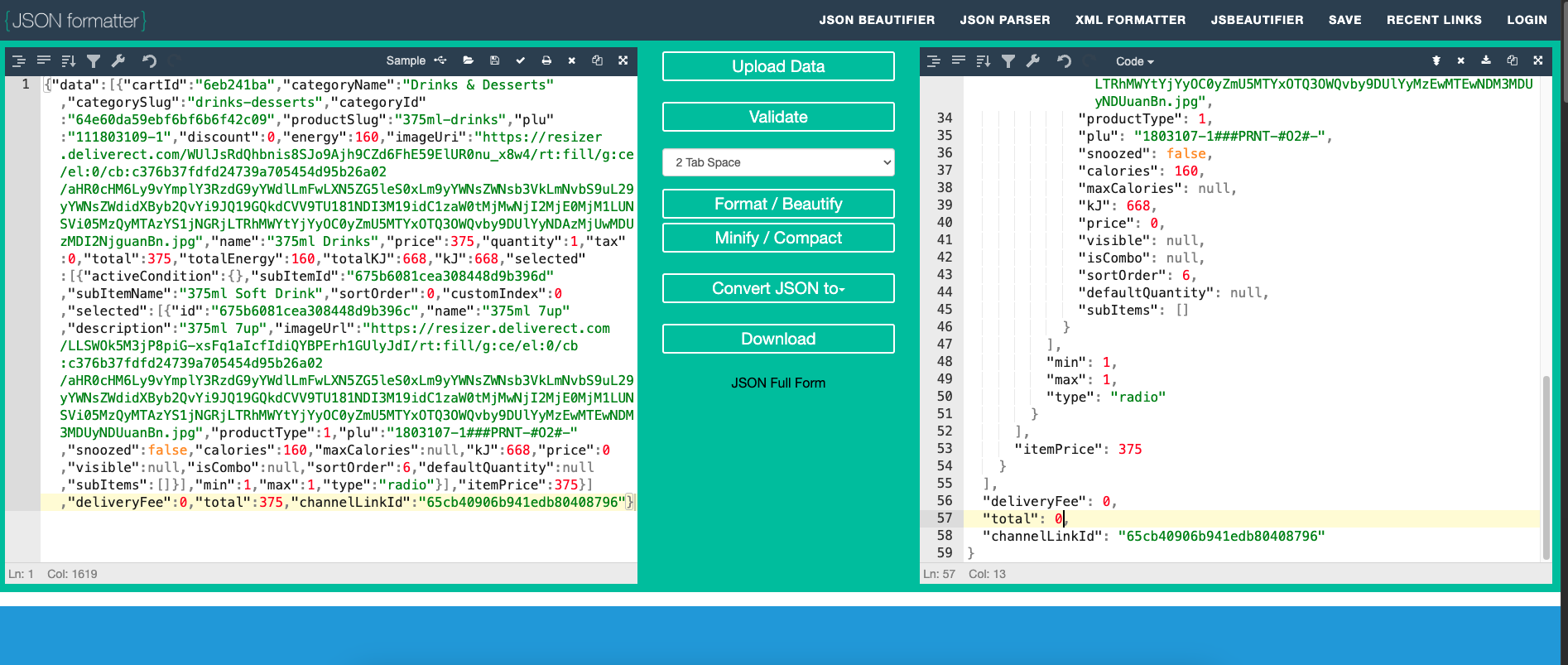
Description automatically generated

1. Go to a website like [jsonformatter.org](https://jsonformatter.org/) and beautify it to make the JSON readable

A screenshot of a computer

Description automatically generated

1. Scroll to the bottom of the text and change the total value to 0 or whatever price you want to pay



1. Click minify and copy the compacted text, then head over to the red rooster website and replace the cart variable with the new code.

A screenshot of a food store

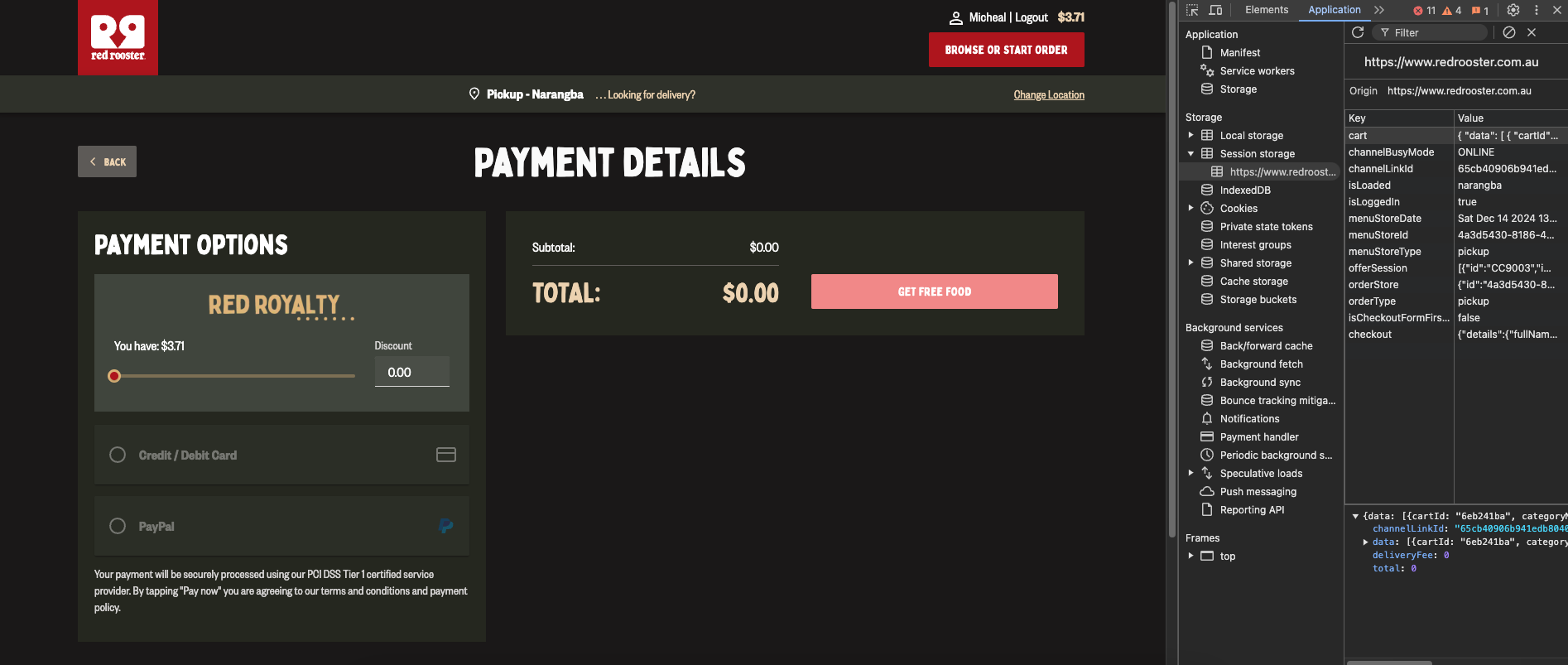
Description automatically generated

1. Refresh the page and go to my cart, if you did everything correctly it should say “Total: $0.00”

A screenshot of a menu

Description automatically generated

1. Click on “CHECKOUT” and “CONTINUE TO PAYMENT”, then you can click “GET FREE FOOD” to submit the order



In this example I did not purchase the item, it was for demonstration purposes only.

This exploit only works with in store pickup. Delivery doesn’t work because it is managed by Door Dash.

## **Technical Details**:

**Vulnerability Type**: Client-side validation flaw

**Affected Component**: Online ordering system, specifically the cart management and payment page

**Exploit Mechanism**:  
The vulnerability arises because the cart’s total price is stored in the browser’s session storage, which can be easily modified using developer tools. The server does not validate the manipulated price, allowing an attacker to change the cart’s total to $0. Once this is done, the website incorrectly assumes the item being purchased is free, skipping the payment page and processing the order without asking for payment details.

**Detailed Example of the Exploit:**

1. The cart object in session storage contains product data, quantities, prices, and totals.
2. The attacker can modify the "total" value in session storage to $0 using developer tools.

**Example of cart data before manipulation**:  
{

"data": [

{

"cartId": "6eb241ba",

"categoryName": "Drinks & Desserts",

"productSlug": "375ml-drinks",

"name": "375ml Drinks",

"price": 375,

"quantity": 1,

"total": 375,

"selected": [

{

"subItemName": "375ml Soft Drink",

"selected": [

{

"name": "375ml 7up",

"price": 0

}

]

}

],

"itemPrice": 375

}

],

"total": 375,

"deliveryFee": 0

}

**Example of cart data after manipulation**:

{

"data": [

{

"cartId": "6eb241ba",

"categoryName": "Drinks & Desserts",

"productSlug": "375ml-drinks",

"name": "375ml Drinks",

"price": 375,

"quantity": 1,

"total": 375,

"selected": [

{

"subItemName": "375ml Soft Drink",

"selected": [

{

"name": "375ml 7up",

"price": 375

}

]

}

],

"itemPrice": 375

}

],

"total": 0, // Manipulated total

"deliveryFee": 0

}

Please note that the JSON strings above will not work when pasted due to having certain identification codes removed in order to make the code more readable

**Impact**:

If exploited, this allows the attacker to bypass the payment page and submit the order with a total price of $0. This would result in unauthorized free orders, causing financial losses for Red Rooster.

## **Impact Assessment**:

If this vulnerability is exploited, it could allow malicious actors to place orders without paying, leading to significant revenue loss for the company. As Red Rooster is a popular brand with a high volume of orders, the exploitation of this flaw could result in large-scale financial losses, especially if discovered and exploited by others.

In addition, the news of a security vulnerability that allows unauthorized orders could damage the company’s reputation. If customers become aware that the online ordering system is insecure, it could erode trust in the brand and drive potential customers away.

Furthermore, Red Rooster’s loyalty program, Red Royalty, requires customers to provide personal information such as their full name, phone number, email address, birthday, and home address for delivery orders. If a cybercriminal were to exploit this vulnerability, they could gain access to this sensitive data. This could lead to identity theft or other malicious activities, and the cybercriminal would likely succeed due to the lack of sufficient security controls in place.

## **Suggested Fixes**:

**Implement Server-Side Validation**:  
Currently, the system relies on client-side validation, which can be manipulated using developer tools. To prevent this, all price-related calculations and validation should be moved to the server side. The server should verify the total amount and ensure that it matches the expected price before allowing the order to be placed.

**Encrypting Session Storage**:  
Instead of storing the cart in plain text, encrypt the cart data so that the client cannot spoof it. This will prevent unauthorized manipulation of the cart total and other sensitive information. The server should decrypt the data for processing and validation, ensuring that any changes to the cart total are securely handled.

**Penetration Testing**:

Regularly conduct penetration testing using third party companies, to identify potential vulnerabilities in the application. Additionally, consider setting up a formal bug bounty program to encourage responsible disclosures and improve the security of your platform.

## **Conclusion:**

The vulnerability discovered in Red Rooster’s online ordering system poses a significant security risk, as it allows malicious actors to bypass payment processes and place unauthorized orders. If left unaddressed, this could lead to substantial financial losses, reputational damage, and potential exploitation of customer data.

By implementing server-side validation, encrypting sensitive data stored on the client-side, and strengthening security measures overall, Red Rooster can mitigate these risks and improve the security of its platform. Prompt action is crucial to prevent the vulnerability from being exploited by others, which could harm the business and its customers.

I urge Red Rooster to address this vulnerability as soon as possible and consider implementing additional security measures, such as regular security testing and the establishment of a bug bounty program, to enhance the long-term protection of the platform.

Thank you for your attention to this matter. I look forward to your response and to working together to resolve this issue.

## **Timeline of Events**:

* **December 10, 2024:** I discovered the exploit on Red Rooster’s website, which allows manipulated orders to be placed without payment.
* **December 11, 2024:** I sent an email to Craveable Brands asking if they had a bug bounty policy, inquiring about how I could report security vulnerabilities.
* **December 12, 2024:** I called Craveable Brands’ customer support and left a voicemail with my details, explaining the security vulnerability I had found.
* **December 14, 2024:** I sent Craveable Brands an email stating that I had found a security vulnerability and requesting the appropriate contact to report it.
* **December 14, 2024:** I documented all the details about the exploit, including its impact and potential fixes, to ensure clear and accurate communication of the issue.
* **December 14, 2024:** I sent Red Rooster an Instagram DM stating that I had found a security vulnerability and requesting the appropriate contact to report it.
* **December 16, 2024:** I left a comment on their latest post informing them to check their direct messages since I have found a security vulnerability.
* **December 18, 2024:** I sent Craveable Brands an Instagram DM following up on the last message, asking for who I should contact.
* **December 23, 2024:** I sent a bug report in the Red Rooster app, asking them to send me an email.