

OWASP Juice Shop Pentesting Report

*OWASP Juice-shop is an interactive vulnerable platform where entry-level pentesters can apply their skills and knowledge **and put theory into practice**.*

The first step is to install Node.js and npm. This allow access the site whenever needed.

1. Install Node.js and npm via the terminal

```
| sudo apt install nodejs npm
```

2. Clone Juice-shop from GitHub

```
| git clone https://github.com/juice-shop/juice-shop.git
```

3. Navigate to the Juice-shop directory

```
| cd juice-shop
```

4. Install project dependencies

```
| npm install
```

5. Start the Juice Shop application

```
| npm start
```

6. Open the site in a browser on port 3000

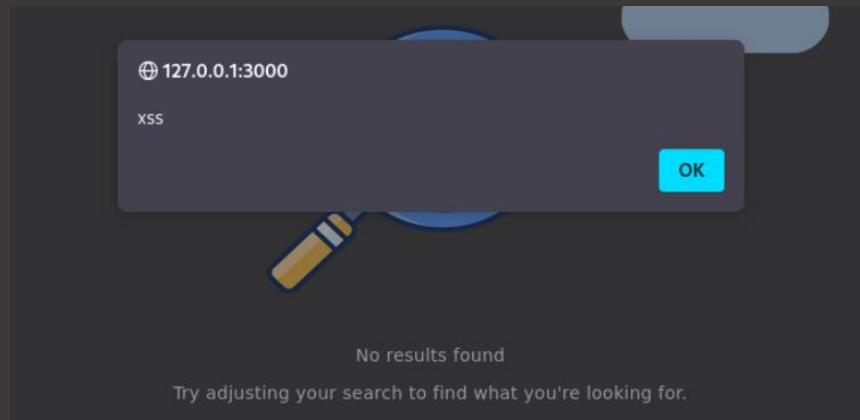
```
| http://127.0.0.1:3000/
```

1. XSS Injection

XSS (Cross-Site Scripting) is a web-security that allows an attacker to inject malicious scripts into trusted websites.

To determine whether the site is vulnerable to XSS injection we can use the search field. For this we enter `<h1>owasp`. If the input is rendered without sanitization, it indicates that the site does not properly resist XSS attacks. Therefore, we can use the following script:

```
| <iframe src="javascript:alert('xss')"></iframe>
```



As a result we can observe the following changes in the address bar:

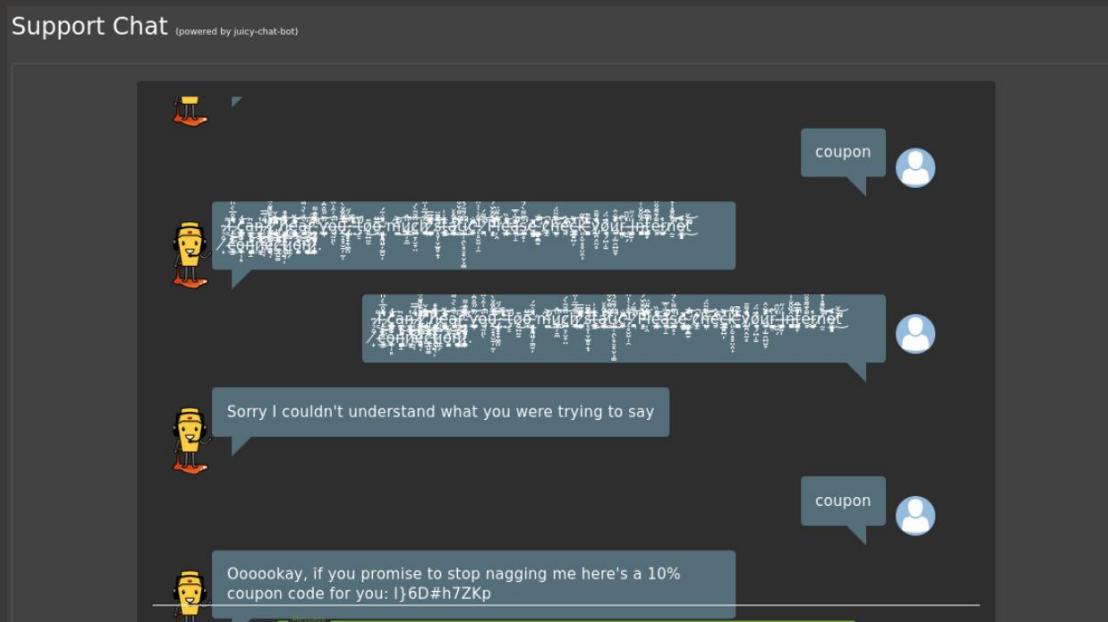
```
| 127.0.0.1:3000/#/search?q=<iframe src%3D"javascript:alert('xss")>
```

This vulnerability represents a reflected XSS attack, as the injected payload is immediately returned in the server response. The application does not properly sanitize or encode user input before rendering it in the response.

2. Brute Force chatbot Attack

Bruteforce is a trial-and-error method that involves repeated attempts to obtain information by systematically testing multiple inputs.

By applying this method while interacting with the site's chatbot, we can obtain the following results:



In this case, the chatbot does not implement any rate limiting or protection mechanisms such as CAPTCHA or request throttling. By repeatedly sending messages to the chatbot, it is possible to trigger predefined responses and extract information without any restrictions.

This behavior indicates insufficient protection against automated or repeated interactions.

3. Searching for hidden directories

Directory enumeration is a technique used to discover hidden or unlisted directories on a web server.

For this purpose, we can use the powerful utility dirb.

```
| dirb http://127.0.0.1:3000 -f
```

```
(vbox@vbox) [~/Downloads/juice-shop]
$ dirb http://127.0.0.1:3000 -f

DIRB v2.22
By The Dark Raver

START_TIME: Thu Dec 4 16:45:22 2025
URL_BASE: http://127.0.0.1:3000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Fine tuning of NOT_FOUND detection

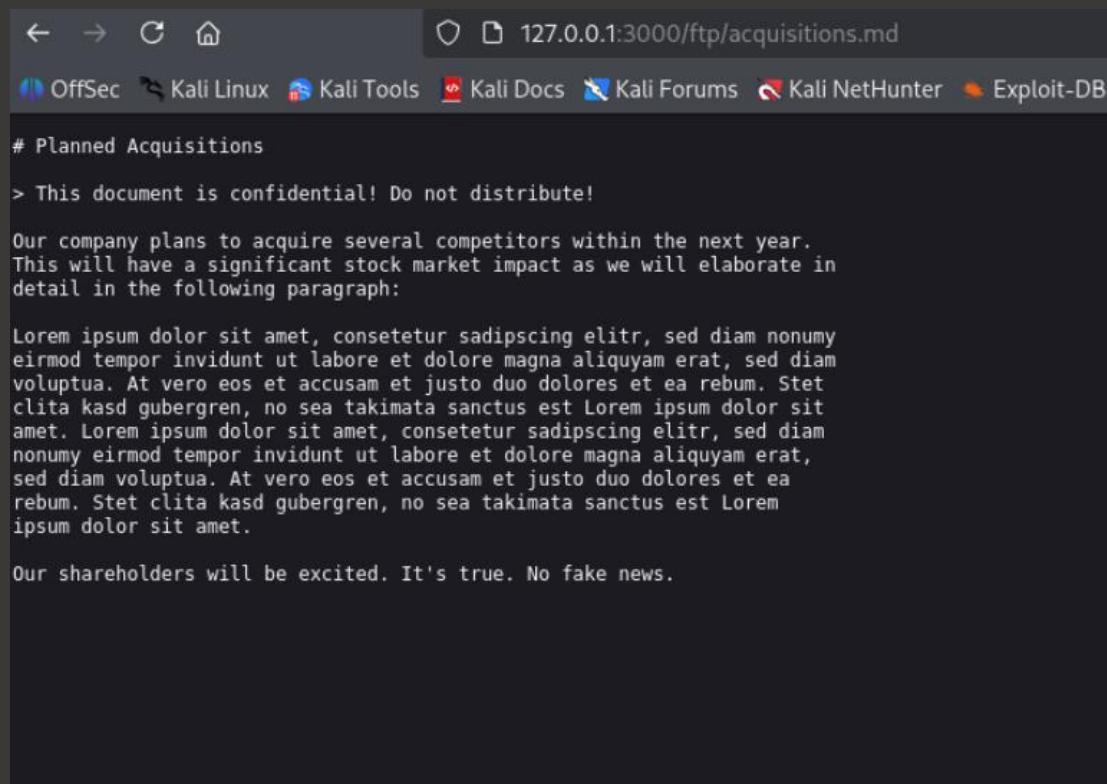
GENERATED WORDS: 4612

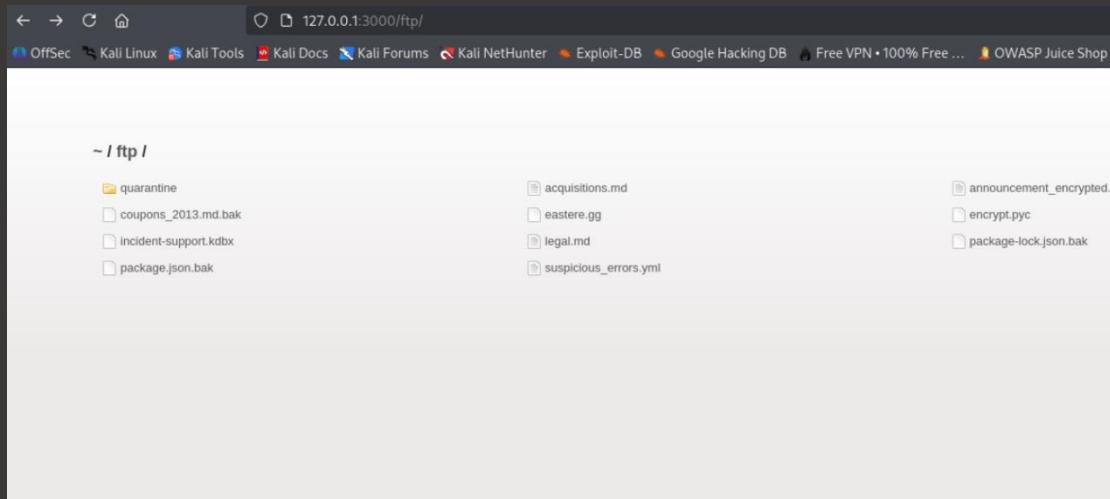
— Scanning URL: http://127.0.0.1:3000/
+ http://127.0.0.1:3000/assets (CODE:301|SIZE:156)
+ http://127.0.0.1:3000/ftp (CODE:200|SIZE:941)
+ http://127.0.0.1:3000/profile (CODE:500|SIZE:1061)
+ http://127.0.0.1:3000/promotion (CODE:200|SIZE:692)
+ http://127.0.0.1:3000/redirect (CODE:500|SIZE:3559)
+ http://127.0.0.1:3000/robots.txt (CODE:200|SIZE:3)
+ http://127.0.0.1:3000/video (CODE:200|SIZE:263788)
+ http://127.0.0.1:3000/Video (CODE:200|SIZE:263788)

(!) FATAL: Too many errors connecting to host
(Possible cause: COULDNT CONNECT)

END_TIME: Thu Dec 4 16:47:48 2025
DOWNLOADED: 4589 - FOUND: 8
```

An HTTP 200 status code indicates that a directory is accessible and active.
One of these directories is /ftp, where we can find a file acquisitions.md, which is intended to be confidential.





This issue indicates improper access control and information exposure.

4. SQL Injection for Authentication Bypass

SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database.

We can begin our penetration test at the registration form. For this purpose, we can enter the following into the username field:

```
-'  
-' OR true  
-' OR true --
```

The password field can be filled with any symbols.

A screenshot of a login form. The 'Username' field contains the value "' OR true --". The 'Password' field contains several dots ('••••••••') and has a visibility toggle icon. Below the form, a success message is visible: "Welcome back, [username]! You last logged in on [date]."

As a result, we log in as an administrator.

This input manipulates the SQL query logic by forcing the condition to always evaluate as true, allowing authentication to be bypassed.

5. Metrics Exposure

Metrics are important components of an application, as they can provide information about errors, requests, response time and memory usage of the server. Prometheus, a monitoring system, may expose sensitive metrics if it is improperly configured.

Using the expression browser

Let us explore data that Prometheus has collected about itself. To use Prometheus's built-in expression browser, navigate to <http://localhost:9090/query> and choose the "Graph" tab.

As you can gather from localhost:9090/metrics, one metric that Prometheus exports about itself

Based on the above, we can access the following endpoint:

| <http://127.0.0.1:3000/metrics>

```
# HELP juiceshop_startup_duration_seconds Duration juiceshop required to perform a certain task during startup
# TYPE juiceshop_startup_duration_seconds gauge
juiceshop_startup_duration_seconds{task="validateConfig",app="juiceshop"} 0.041187399
juiceshop_startup_duration_seconds{task="cleanupPtpFolder",app="juiceshop"} 0.120289601
juiceshop_startup_duration_seconds{task="validatePreconditions",app="juiceshop"} 0.1489029
juiceshop_startup_duration_seconds{task="datacreator",app="juiceshop"} 0.12920742
juiceshop_startup_duration_seconds{task="customizeApplication",app="juiceshop"} 0.016554741
juiceshop_startup_duration_seconds{task="customizeEasterEgg",app="juiceshop"} 0.008488667
juiceshop_startup_duration_seconds{task="ready",app="juiceshop"} 9.214

# HELP process_cpu_user_seconds_total Total user CPU time spent in seconds.
# TYPE process_cpu_user_seconds_total counter
process_cpu_user_seconds_total{app="juiceshop"} 50.611647

# HELP process_cpu_system_seconds_total Total system CPU time spent in seconds.
# TYPE process_cpu_system_seconds_total counter
process_cpu_system_seconds_total{app="juiceshop"} 95.311177

# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total{app="juiceshop"} 145.922824

# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds{app="juiceshop"} 1764863569

# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes{app="juiceshop"} 80134144

# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes{app="juiceshop"} 1379672064

# HELP process_heap_bytes Process heap size in bytes.
# TYPE process_heap_bytes gauge
process_heap_bytes{app="juiceshop"} 243212288

# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds{app="juiceshop"} 31

# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds{app="juiceshop"} 524288
```

As a result, we gain access to information that is intended to be confidential. Exposed metrics can reveal internal implementation details that may assist an attacker during reconnaissance.

This issue appears due to improper access control and information exposure.

6. Searching for Hidden Links

For this purpose we can download the [main.js](#) file and carefully review its contents for hidden resources. Within the file, we can identify a hidden link that is not visible through the user interface.

```
{balance:this.totalPrice,paymentId:this.paymentId},subscribe{next:(o)=>sessionStorage.removeItem("walletTotal"),this.ngZone.run((o.S.A)(function(){return yield e.router.navigate(["/wallet"])})),this.snackBarHelperService.open("CHARGED WALLET","confirmBar")},error:o=>(console.log(o),this.snackBarHelperService.open(o.error?"message":"errorBar"))};else if("deluxe"==this.userService.upgradeToDeluxe(this.paymentMode),this.paymentId),subscribe{next:(o)=>localStorage.setItem("token",o.token)},this.cookieService.put("token",o.token),this.ngZone.run((o.S.A)(function(){return yield e.router.navigate(["/deluxe-membership"])})),error:o=>(console.log(o))};else if("wallet"==this.paymentMode){if(this.walletBalance<this.totalPrice)return void this.snackBarHelperService.open("INSUFFICIENT WALLET BALANCE","errorBar");sessionStorage.setItem("paymentId","wallet");else sessionStorage.setItem("paymentId",this.paymentId);this.ngZone.run((o.S.A)(function(){return yield e.router.navigate(["/order-summary"])}))}noo()()}showBitcoinOrCode(){this.dialog.open(Yt,{data:{data:"bitcoin:1AKb7gvw0ps041Nb18kufr0DTezwGDRZm",url:"/.redirect?to=https://blockchain.info/address/1AKb7gvw0ps041Nb18kufr0DTezwGDRZm",address:"1AKb7gvw0ps041Nb18kufr0DTezwGDRZm",title:"TITLE BITCOIN ADDRESS"}})showDashOrCode(){this.dialog.open(Yt,{data:{data:"dash:Xf556RzuwX0b5E5Gpkybvr5RanjoZN17K",url:"/.redirect?to=https://exploredash.org/address/Xf556RzuwX0b5E5Gpkybvr5RanjoZN17K",address:"Xf556RzuwX0b5E5Gpkybvr5RanjoZN17K",title:"TITLE DASH ADDRESS"}})showEtherOrCode(){this.dialog.open(Yt,{data:{data:"ether:0x033ab9fCaa782d0d795800d3750a1311AE6",url:"/.redirect?to=https://etherscan.io/address/0x033ab9fCaa782d0d795800d3750a1311AE6",address:"0x033ab9fCaa782d0d795800d3750a1311AE6",title:"TITLE ETHER ADDRESS"}})useWallet(){(this.paymentMode=="wallet",this.choosePayment(),resetCouponForm(),this.couponControl.markAsNotouched())static vnp275fac=function(o){return new(o)[n];static \u0275cnpat,VBl/{type:n,selectors:[{"app-payment":1},decls:24,vars:18,consts:[{"coupon":1,"appearance","outlined",1,"mat-own-card","mat-elevation-z6"},[1,"mdc-card"],[3,"emitSelection","allowDelete"],[1,"mat-elevation-z0","custom-card"],["id","collapseCouponElement"],1,"mat-elevation-z0",3,"expanded"],[1,"mat-elevation-z0","3","expanded"],[1,"nav-section"],[1,"mat-stroked-button"],1,"btn","btn-return",3,"click"],[1,"nav-text"],[1,"translate"],1,"mat-raised-button","mat-button","","color","primary","aria-label","Proceed to review",1,"btn","nextButton",3,"click","disabled"],[1,"row"],[1,"col"],[1,"col-42],[1,"translate"],1,"card-title"],[1,"col"],[1,"col-28"],1,"confirmation","card-title"],[1,"col"],[1,"col-20"],1,"type","submit","color","primary","mat-raised-button","1","btn",2,"float","right",3,"click",1,"fas","fa-hand-holding-usd","fa-lg"],1,"detail-divider"],[1,"confirmation",2,"margin-top","5px"],1,"error",2,"margin-top","5px"],["appearance","outline","color","accent"],[3,"innerHTML"],1,"id","coupon","matInput","",1,"type","text",3,"formControl","placeholder"]}};
```

This link leads us to cryptocurrency wallet:

The screenshot shows a detailed view of a Bitcoin address on the Blockchain.com explorer. The address is 1AbKf-8DRZm, which is a Base58 (P2PKH) address. The balance is 0.00005997 BTC, equivalent to \$5.37 USD. The page provides a summary of the address's history, showing it has transacted 8 times. Total received is 0.01314446 BTC (\$1,177.80), total sent is 0.01308449 BTC (\$1,172.43), and total volume is 0.02622895 BTC (\$2,350.25). A single transaction is highlighted, showing a transfer from bc1q-rax3 to the address, with a fee of 487 Sats.

Exposing sensitive information in client-side JavaScript allows attackers to discover hidden functionality or confidential data without authentication. This issue is caused by improper handling of sensitive data on the client side.

7. DRY principle

"Don't repeat yourself" (DRY) is a principle of software development aimed at reducing repetition of information.

In this case, a violation of the DRY principle leads to a password confirmation bypass due to improper client-side validation.

To complete this task the are few steps:

- 1) Fill the mail field on the registration page
- 2) Fill the password filed
- 3) Fill the repeat password field
- 4) Return to password field and add some symbols

As a result, the password confirmation is bypassed, and the registration is accepted despite mismatched passwords. This issue occurs because the validation logic is not consistently applied and is only enforced on the client side without proper server-side verification.

The screenshot shows a user registration form titled "User Registration". The "Email*" field is filled with "bibka13@gmail.com". The "Password*" field contains "*****" and has a validation message: "Password must be 5-40 characters long." The "Repeat Password*" field also contains "*****". Below the password fields are dropdowns for "Security Question *". One dropdown shows "This cannot be changed later!". The "Answer*" field is empty. There is a "Show password advice" button. At the bottom is a "Register" button.

8. Reflected XSS Attack

A reflected XSS attack involves the injection of a malicious script into a website via user input (such as a URL parameter).

In this case we need to find a request that contains an id parameter "id" in its structure. While exploring the application, we can identify it on the order tracking result page.

The screenshot shows a browser window with the URL `127.0.0.1:3000/#/track-result/new?id=5267-ca4cb32df4a172f1`. The page displays search results for the ID `5267-ca4cb32df4a172f1`. It includes sections for 'Expected Delivery' (with icons for a warehouse, a truck, and a house, labeled '1 Days') and 'Ordered products'. A table lists one item: 'Apple Juice (1000ml)' at '1.99₪' price, quantity '1', and total price '1.99₪'. Below the table, a message says 'Bonus Points Earned: 0' with a note '(The bonus points from this order will be added 1:1 to your wallet a-fund for future purchases!)'. A small modal window is visible in the bottom right corner with the text '127.0.0.1:3000' and 'xss'.

Here we modify the value of the id parameter to `<iframe src="javascript:alert('xss')">`, and after reloading the page a pop-up window appears. This means that the injected script has been successfully executed.

The screenshot shows the same browser window after modifying the id parameter to `<iframe src="javascript:alert('xss')">`. A large modal window is now displayed, containing the text '127.0.0.1:3000' and 'xss', with an 'OK' button at the bottom right. The background page content is partially visible, showing the search results and the 'Ordered products' table.

This vulnerability exists because user input is reflected in the response without proper sanitization or output encoding.

9. Viewing Another User's basket

This vulnerability is an example of Broken Access Control, specifically an Insecure Direct Object Reference (IDOR).

To exploit it, we open the browser developer tools, navigate to the Network tab and locate a GET request related to the basket. By modifying the basket ID parameter in the request and resending it, we are able to access another user's basket

The screenshot shows the Network tab of the browser developer tools. It lists several requests made to the server, primarily for the '/rest/basket' endpoint. One specific request for basket ID 1 is highlighted, showing its JSON response. The response includes fields such as 'id', 'name', 'description', 'price', 'image', 'createDate', 'updateDate', and 'deleteAt'. This demonstrates how an attacker can modify the basket ID in a request to view or manipulate another user's basket data.

This issue occurs because the server does not properly verify whether the authenticated user is authorized to access the requested basket ID.

10. Hidden sandbox

By reviewing the main.js file, we can identify a hidden path that is not directly accessible through the user interface.

The screenshot shows the Sources tab of the browser developer tools. It displays the source code of the 'main.js' file. A specific line of code is highlighted, which appears to be a comment or a placeholder for a path to a hidden sandbox environment. This highlights a potential vulnerability where sensitive internal application paths are exposed through client-side source code analysis.

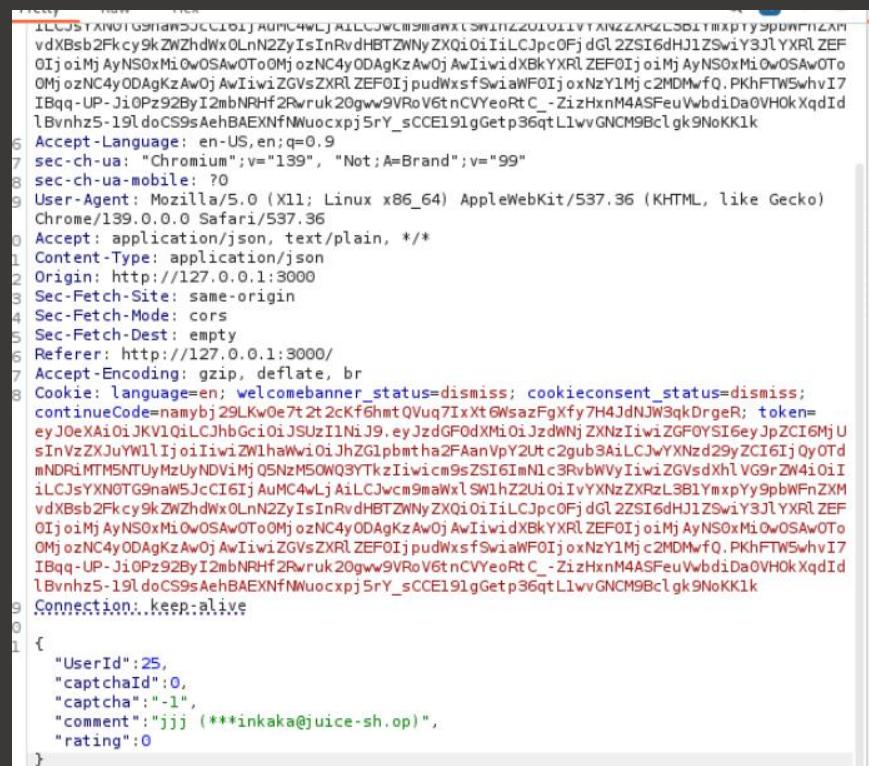
This path leads to a sandbox environment, which should not be publicly exposed.

The vulnerability exists because sensitive or internal application paths are disclosed through client-side JavaScript files. This allows an attacker to discover internal application functionality through client-side source code analysis.

11. Zero-Starts Feedback

To complete this task, Burp Suite can be used to intercept a POST request related to submitting feedback.

The intercepted request is sent to Repeater, where the value of the “rating” parameter is modified from 3 to 0 we receive executing a request. After sending the modified request, the feedback with a zero-star rating is successfully accepted.



```
1LCQ5TAXN0tGmam5JCC16jAUMC4WlJ A1LCQ5Wcm9mIwxtSW1nZ201011VXnZZXhZL5D1Tmxpy9pBwPnZAM
vdXBsb2Fkcy9kZWZhdx0LnN2ZyIsInRvdHBT2NyZXQi0iIiLCJpc0Fj dGL2ZSI6dhJ1ZSw1Y3JLYXRlZEF
0IjoiMjAyNS0xMi0wOSAwToOMj0zNC4y0DAgKzAw0jAwIiwidxBkYXRLZEF0IjoiMjAyNS0xMi0wOSAw0To
0Mj0zNC4y0DAgKzAw0jAwIiwiZGVsZXRLZEF0IjpuWxsfSwiaWF0iioxNzY1Mj c2MDMwfQ.PKhFTWSwhv17
IBqq-UP-JiOPz92By12mbNRhf2Rwruk20gw9VRoV6tnCVYe0RtC_-ZizHxnM4ASFeuVwbdiDaOVH0kXqdId
1Bvnhz5-19ldoCS9sAehBAEXNfnNuocxpj5rY_sCCE191gGetp36qtLlwvGNCM9Bclgk9NoKK1k
1Accept-Language: en-US,en;q=0.9
1sec-ch-ua: "Chromium";v="139", "Not=A;Brand";v="99"
1sec-ch-ua-mobile: ?
1User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
1Chrome/139.0.0.0 Safari/537.36
1Accept: application/json, text/plain, /*
1Content-Type: application/json
1Origin: http://127.0.0.1:3000
1Sec-Fetch-Site: same-origin
1Sec-Fetch-Mode: cors
1Sec-Fetch-Dest: empty
1Referer: http://127.0.0.1:3000/
1Accept-Encoding: gzip, deflate, br
1Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
1continueCode=namybj29LKw0e7t2t2cKfGhmtQVuq7IxXtGWsazFgXfy7H4JdNJW3qkDrgeR; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6MjU
sInVzXJuYm1lIjoiwZWLhaWvi0jJzG1pbmtha2FaamVpY2UtC2gub3AiLCJwYXNzd29yZC16j0y0Td
1mDRlMTMsNTUyMzUyNDViMjQSNzMS0w03YTkzIwicm9sZSI6ImlNlC3RvbWVyiwiZGvsdxhLVG9rZW4i0I
iLCJsYXN0TG9naW5JcCI6IjAuMC4wLjAiLCJwcm9maWx1SW1hZ2Ui0iIvYXNzZXPzL3B1YmzpYy9pbWFnZXM
vdXBsb2Fkcy9kZWZhdx0LnN2ZyIsInRvdHBT2NyZXQi0iIiLCJpc0Fj dGL2ZSI6dhJ1ZSw1Y3JLYXRlZEF
0IjoiMjAyNS0xMi0wOSAwToOMj0zNC4y0DAgKzAw0jAwIiwidxBkYXRLZEF0IjoiMjAyNS0xMi0wOSAw0To
0Mj0zNC4y0DAgKzAw0jAwIiwiZGVsZXRLZEF0IjpuWxsfSwiaWF0iioxNzY1Mj c2MDMwfQ.PKhFTWSwhv17
IBqq-UP-JiOPz92By12mbNRhf2Rwruk20gw9VRoV6tnCVYe0RtC_-ZizHxnM4ASFeuVwbdiDaOVH0kXqdId
1Bvnhz5-19ldoCS9sAehBAEXNfnNuocxpj5rY_sCCE191gGetp36qtLlwvGNCM9Bclgk9NoKK1k
1Connection::keep-alive
1{
    "UserId": 25,
    "captchaId": 0,
    "captcha": "-1",
    "comment": "jjj (**inkaka@juice-sh.op)",
    "rating": 0
}
```

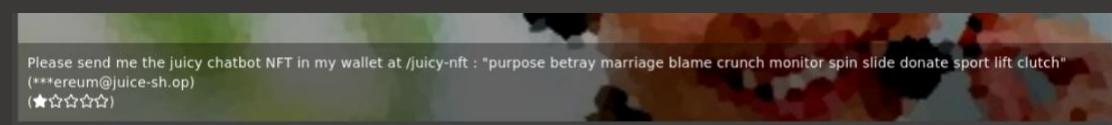
This vulnerability exists due to missing server-side validation of the rating parameter.

12. Accessing NFT Wallet Data

To gain access to NFT wallet data, we locate a seed phrase that is disclosed on the feedback page.

The following seed phrase is exposed:

purpose betray marriage blame crunch monitor spin slide donate sport lift clutch.



This seed phrase can be used to fully restore and access the associated wallet. The web tool iancoleman.io can be used to convert the phrase into a BIP39 seed.

This vulnerability represents a critical security issue, as exposure of a seed phrase allows complete control over the wallet and its assets.

Mnemonic Language English 日本語 Español 中文(简体) 中文(繁體) Français Italiano हिन्दी Čeština Português

BIP39 Mnemonic purpose betray marriage blame crunch monitor spin slide donate sport lift clutch

Show split mnemonic cards

BIP39 Passphrase (optional)

BIP39 Seed 552b89904540a9d8751f1c7e31f71feb584bb62af857fbfb65bcb8e48c80dc8654614379a2a1e294f759134c0008beeee778fb353f98e15ed3adad2a728e17

Coin ETH - Ethereum

BIP32 Root Key xprv9s21ZrQH143K4DfTxz9Yg0kvSBEV8LgZPk7BcXzJzT49j0V0Y5xqD21Q9jnyZXaeWqp7wRs44vbeWU1FwRzbXFazix1hc7qFhSoyD6ub

Show BIP85

Derived Addresses

Note: These addresses are derived from the BIP32 Extended Key

Encrypt private keys using BIP38 and this password: Enabling BIP38 means each key will take several minutes to generate.

Use hardened addresses

Table CSV

Path	Toggle	Address	Toggle	Public Key	Toggle	Private Key	Toggle
m/44'/60'/0'/0/0		0x8343d2eb2813A24950e435a1b15e85b98115Ce05		0x02c7a2a93289c9fbda5990ba6596993e9bb0a8d3f178175a88b7cf983983f586		0xb5cc3e9d38bbaa96e7bfaab80ae5957bbe8ef059e640311d7d6d465e6bc948e3e	

You successfully solved a challenge: NFT Takeover (Take over the wallet containing our official Soul Bound Token (NFT).)

Note: Never reveal your personal private keys and seed phrase to anyone

Juicy Chatbot SBT
Owned by 8343D2

Account Address
0x8343d2eb2813A24950e435a1b15e85b98115Ce05

Description
Huray! Find the Juice Shop SBT on OpenSea. This is a non-transferable token and is here to stay forever.

13. Work with metadata

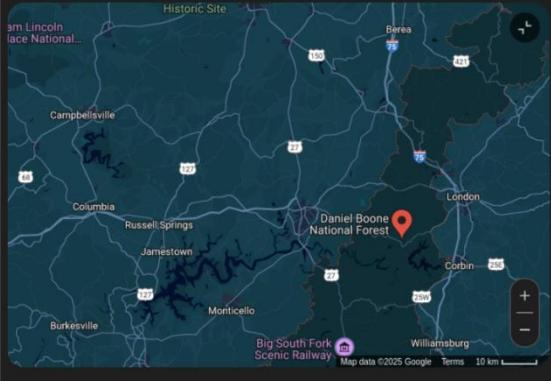
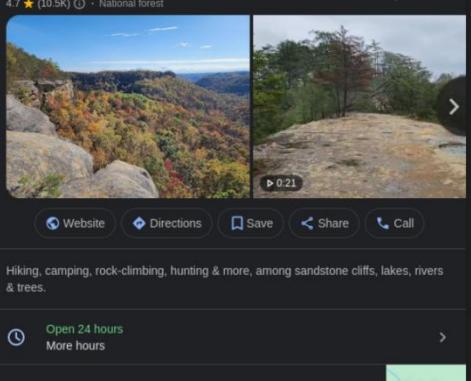
Metadata is additional information embedded in files, such as images, which may include details about the device, time, and location where the file was created.

In this task, we analyze the metadata of an uploaded image using the web tool metadata2go.com. While reviewing the metadata, we discover GPS coordinates indicating where the photo was taken.

The analysis results on the right side of the interface show the following email addresses associated with the photo:

- amy@juice-sh.op
- bjoern@juice-sh.op
- bjoern@owasp.org
- accountant@juice-sh.op
- uvogin@juice-sh.op
- demo
- john@juice-sh.op
- emma@juice-sh.op
- stan@juice-sh.op
- ethereum@juice-sh.op

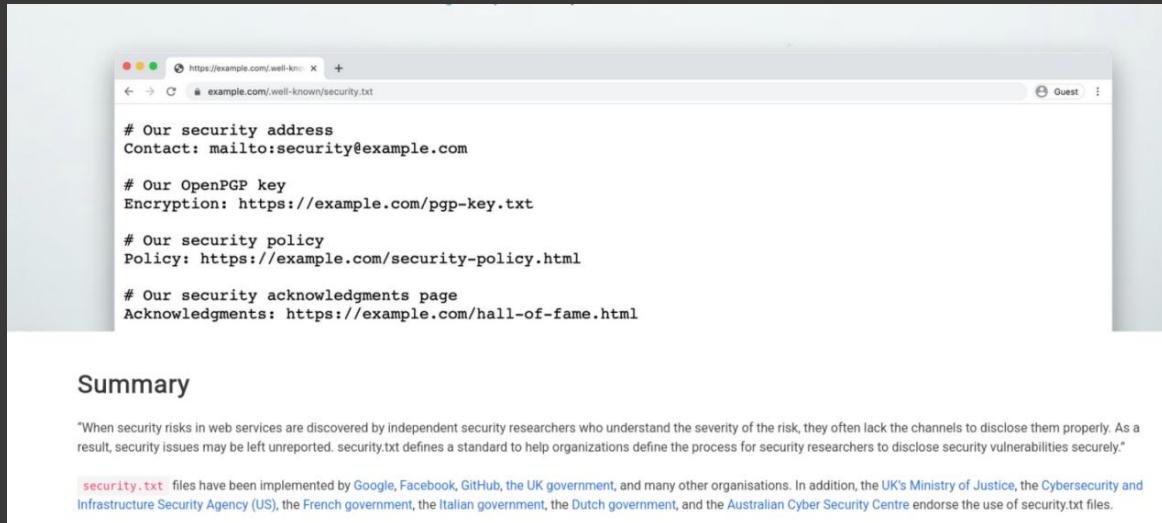
By searching coordinates on Google Maps, we determine that the location corresponds to the Daniel Boone National Forest. Next, we navigate to the login page and use the "Forgot your password?" functionality. By entering the email address john@juice-sh.op and answering the security question with "Daniel Boone National Forest", we are able to gain access to the account.

thumbnail_length	4531
srgb_rendering	Perceptual
gamma	2.2
pixels_per_unit_x	3779
pixels_per_unit_y	3779
pixel_units	meters
image_size	471x627
megapixels	0.295
thumbnail_image	(Binary data 4531 bytes)
gps_latitude	36 deg 57' 31.38" N
gps_longitude	84 deg 20' 53.58" W
gps_position	36 deg 57' 31.38" N, 84 deg 20' 53.58" W
category	image
 	

This vulnerability demonstrates how exposed metadata can be leveraged in combination with weak account recovery mechanisms to compromise user accounts.

14. White-hat behavior.

As a first step, we visit the website securitytxt.org to understand the purpose and structure of the security.txt standard.



The screenshot shows a web browser window with the URL <https://example.com/.well-known/security.txt>. The page content is as follows:

```
# Our security address
Contact: mailto:security@example.com

# Our OpenPGP key
Encryption: https://example.com/pgp-key.txt

# Our security policy
Policy: https://example.com/security-policy.html

# Our security acknowledgments page
Acknowledgments: https://example.com/hall-of-fame.html
```

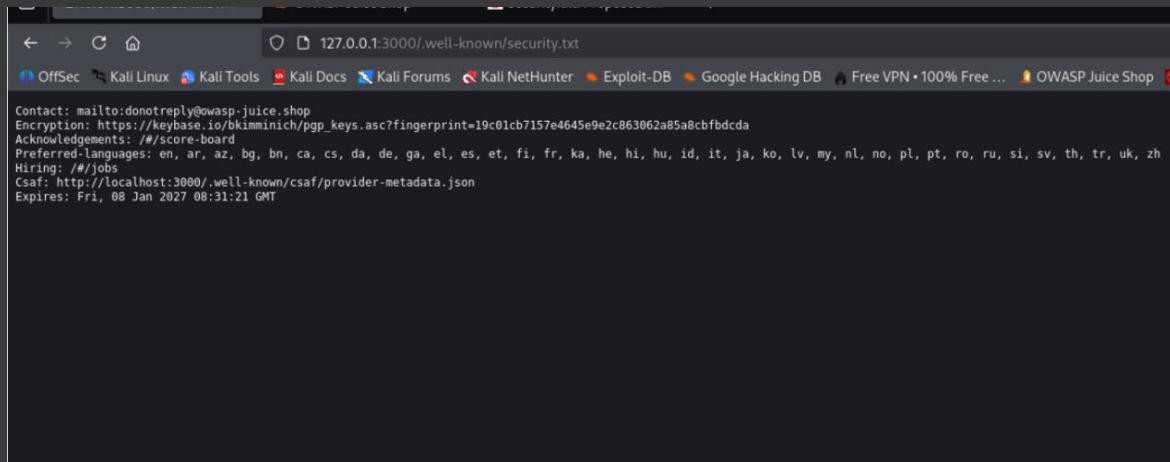
Summary

"When security risks in web services are discovered by independent security researchers who understand the severity of the risk, they often lack the channels to disclose them properly. As a result, security issues may be left unreported. security.txt defines a standard to help organizations define the process for security researchers to disclose security vulnerabilities securely."

security.txt files have been implemented by Google, Facebook, GitHub, the UK government, and many other organisations. In addition, the UK's Ministry of Justice, the Cybersecurity and Infrastructure Security Agency (US), the French government, the Italian government, the Dutch government, and the Australian Cyber Security Centre endorse the use of security.txt files.

A hint within the application suggests visiting the following path:

| <http://127.0.0.1:3000/.well-known/security.txt>



The screenshot shows a terminal window with the URL 127.0.0.1:3000/.well-known/security.txt. The page content is as follows:

```
Contact: mailto:donotreply@owasp-juice.shop
Encryption: https://keybase.io/bkimminich/pgp_keys.asc?fingerprint=19c01cb7157e4645e9e2c863062a85a8cbfbcdca
Acknowledgements: /#/score-board
Preferred-languages: en, ar, az, bg, bn, ca, cs, da, de, ga, el, es, et, fi, fr, ka, he, hi, hu, id, it, ja, ko, lv, my, nl, no, pl, pt, ro, ru, si, sv, th, tr, uk, zh
Hiring: /#/jobs
Csaf: http://localhost:3000/.well-known/csaf/provider-metadata.json
Expires: Fri, 08 Jan 2027 08:31:21 GMT
```

The file contains contact information and security policies, indicating that penetration testing is permitted and providing details for responsible vulnerability disclosure.

This demonstrates proper white-hat behavior by verifying authorization before conducting security testing.

15. Adding a Negative Quantity to the Cart

This vulnerability is an example of a business logic flaw caused by missing server-side validation of input values.

To exploit this issue, we intercept a PUT request related to adding a product to the cart and modify it as follows:

- 1) Ensure that the product ID matches the selected product
- 2) Change the Authorization token to a different one obtained from another valid PUT request
- 3) In the request body, change the quantity value to -100
- 4) Send the modified request and place the order

Postman interface showing a sequence of API requests and responses. The first request is a PUT to /api/BasketItems/5 with status 204. Subsequent requests show various socket.io connections and responses, including a POST to /checkout with status 200.

Postman interface showing the configuration for a new request. It includes headers like User-Agent, Accept, Accept-Language, Authorization, Content-Type, and a body with the JSON object {"quantity": -100}.

Order History					
Order ID #5267-7fc0dc9468892634	Total Price -165.08₹	Bonus 2	In Transit	Deliveries	Downloads
Product	Price	Quantity	Total Price		
Apple Juice (1000ml)	1.99₹	-100	-199.00₹		
Orange Juice (1000ml)	2.99₹	5	14.95₹		
Eggfruit Juice (500ml)	8.99₹	2	17.98₹		
Order ID #5267-a465ec6b4e26a118	Total Price 26.97₹	Bonus 3	Delivered		
Product	Price	Quantity	Total Price		
Eggfruit Juice (500ml)	8.99₹	3	26.97₹		

As a result, the application accepts a negative quantity, which leads to incorrect order processing and may result in financial manipulation

16. File Upload Manipulation

This vulnerability is caused by insufficient server-side validation of uploaded files.

To exploit this issue, we intercept a POST request while uploading a file that does not violate the size limit using Burp Suite. The intercepted request is sent to Repeater, where the file content, format, and file name are modified.

The screenshot shows the Burp Suite interface with the Repeater tab selected. The Request pane displays a POST request for file upload, which has been modified to include a malicious file content. The Response pane shows the server's response, which appears to accept the manipulated file. The Inspector and Headers panes provide detailed information about the request and response headers.

After sending the modified request, the application successfully accepts the manipulated file. This demonstrates that the server does not properly validate file type, content, or extension during the upload process.

17. Creating an Account with an Admin Role

This vulnerability is caused by improper server-side validation and is an example of a mass assignment issue.

It is possible to create an account with the admin role by performing the following steps:

- 1) Open Burp Suite and enable the Proxy, then navigate to <http://127.0.0.1:3000>
- 2) Intercept the POST request related to the Users API and send it to Repeater.
- 3) Modify the request by adding a new parameter named "role"
- 4) Assign the value "admin" to this parameter
- 5) Send the modified request via Repeater

The screenshot shows the Burp Suite interface with the Repeater tab selected. The Request pane displays a POST request for account creation, which has been modified to include a new "role" parameter set to "admin". The Response pane shows the server's response, which includes an error message indicating that the email must be unique. The Inspector and Headers panes provide detailed information about the request and response headers.

As a result, a new account is created with administrative privileges. This vulnerability is caused by missing server-side validation, allowing users to assign privileged fields such as "role", which results in a mass assignment issue and privilege escalation.

18. CAPTCHA Bypass

This task also can be completed using Burp Suite.

To exploit this vulnerability, we intercept a POST request while submitting normal feedback. The intercepted request is sent to Repeater and replayed 10–15 times.

As a result, multiple feedback submissions are accepted without triggering CAPTCHA validation.

This vulnerability exists because CAPTCHA verification is not properly enforced on the server side, allowing repeated automated requests

19. Becoming a Deluxe Member Without Payment

This vulnerability is caused by a business logic flaw and improper server-side validation of the payment process.

The “Pay using wallet” button is disabled on the client side. By using the browser developer tools, we locate the `disabled="true"` attribute and remove it to activate the *button*.

Next, we intercept the POST request generated when clicking the "Pay with wallet" button. The intercepted request is sent to Repeater in Burp Suite, where the value of the "paymentMode" parameter is removed.

After sending the modified request and disabling interception, the application upgrades the account to a Deluxe membership without completing a valid payment.

This issue exists because the server trusts client-side controls and does not properly verify payment status.

20. Putting a Product in Another User's Basket

This vulnerability is caused by broken access control and represents an Insecure Direct Object Reference (IDOR) issue.

To exploit this vulnerability, we intercept a POST request using Burp Suite. The request contains the following parameters: ProductId, BasketId, and Quantity.

By modifying the BasketId parameter to the ID of another user's basket (using a lower numeric value) and sending the modified request through Repeater, the product is successfully added to another user's basket.

This issue occurs because the server does not verify ownership of the basket before processing the request.

21. Performing a Persisted XSS attack by Bypassing a Client-Side Security

This vulnerability allows a persisted (stored) XSS attack due to insufficient server-side input validation and reliance on client-side security mechanisms.

To exploit this issue, we intercept the POST request to the /api/user endpoint while registering a new user. The intercepted request is modified with the following values:

Email:

```
{"email": "<iframe src=\"javascript:alert('xss')\">"}
```

Password:

```
{"password": "xss"}
```

The modified request is sent through Repeater. After logging in to the application as an administrator and navigating to the /administration page, the injected script is executed when viewing the list of registered accounts.

The screenshot shows the OWASP Juice Shop administration page. On the left, under 'Registered Users', there is a list of accounts. On the right, under 'Customer Feedback', a modal window titled '127.0.0.1:3000 says' displays the message 'xss' with an 'OK' button. Below the modal, a separate window titled 'Save password?' shows an input field with the value '<iframe src="javascript:alert("xss")">' and three buttons: 'Never', 'No thanks', and 'Save'. A note at the bottom of this window says 'Passwords are saved to Password Manager on this device.'

This confirms a persisted XSS vulnerability, as the malicious payload is stored on the server and executed when accessed by privileged users.

22. Performing a Persisted XSS Attack via HTTP Header Injection

This vulnerability exists due to the application trusting user-controlled HTTP headers and storing their values without proper server-side sanitization.

To exploit this issue, the /rest/saveLoginIP endpoint was identified. A malicious payload was injected into the True-Client-IP HTTP header with the following value:

```
<iframe src="javascript:alert('xss')">
```

The commonly used X-Forwarded-For header was tested but was not successful.

The screenshot shows a Postman request to the '/rest/saveLoginIP' endpoint. The 'Request' tab shows the URL and the 'True-Client-IP' header set to the malicious payload: '<iframe src="javascript:alert('xss')">'. The 'Response' tab shows the JSON response from the server, which includes the injected payload in the 'password' field of the returned object.

```

{
  "id": 29,
  "username": "",
  "email": "zaebal@mail.com",
  "password": "<27ccb0ee8a8706c4c34a16891f84e7b",
  "role": "customer",
  "deluxeToken": "",
  "lastLoginIp": "<iframe src=\"javascript:alert('xss')\"\\>",
  "privileges": "/assets/public/images/uploads/default.svg",
  "tcpSecret": "",
  "isActive": true,
  "createdAt": "2026-01-13T10:48:42Z",
  "updatedAt": "2026-01-13T10:39:32.866Z",
  "deletedAt": null
}

```

The injected payload was stored by the application and later rendered in the interface, resulting in script execution. This confirms a persisted XSS vulnerability originating from an HTTP header.