



Audit Report

May, 2023

For



Table of Content

Executive Summary	01
Techniques and Methods	02
Checked Vulnerabilities	04
Manual Testing	05
High Severity Issues	05
Medium Severity Issues	09
Low Severity Issues	18
Informational Issues	24
Closing Summary	25
About QuillAudits	26



Executive Summary

Project Name	Bit5
Overview	Bit5 is a Digital Asset Marketplace, providing progressive, secure, and entertaining Web3 products.
Timeline	3 May, 2023 - 23 May, 2023
Scope of Audit	The scope of this pentest was to analyze the Web App, WebSocket AND corresponding API endpoints for quality, security, and correctness.

In Scope:

Web-App and API:

<https://bit5.com>

<https://lending.bit5.com/>

<https://test.bit5.com/gaming>

https://bit5.com/api/*

https://lending.bit5.com/api/*

WebSocket API: <stream.prod.bit5.com> [codepen.io/tha_dev/pen/WNgWvQj?editors=1111]

<https://gamingapi.test.bit5.com>



High

Low

Medium

Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	3	5	3	1

Techniques and Methods

Throughout the pentest of BIT-Wallet, care was taken to ensure:

- Information gathering – Using OSINT tools information concerning the web architecture, information leakage, web service integration, and gathering other associated information related to web server & web services.
- Using Automated tools approach for Pentest like Nessus, Wireshark, etc.
- Platform testing and configuration
- Error handling and data validation testing
- Encryption-related protection testing
- Client-side and business logic testing

Tools and Platforms used for Pentest

- Burp Suite
- DNSenum
- Dirbuster
- SQLMap
- Acunetix
- Neucli
- Nabbu
- Turbo Intruder
- Nmap
- Metasploit
- Horusec
- Postman
- Netcat
- Nessus and many more.



Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Checked Vulnerabilities

✓ Improper Authentication	✓ Broken Access Controls
✓ Improper Resource Usage	✓ Insecure Cryptographic Storage
✓ Improper Authorization	✓ Insufficient Cryptography
✓ Insecure File Uploads	✓ Insufficient Session Expiration
✓ Insecure Direct Object References	✓ Insufficient Transport Layer Protection
✓ Client-Side Validation Issues	✓ Unvalidated Redirects and Forwards
✓ Rate Limit	✓ Information Leakage
✓ Input Validation	✓ Broken Authentication and Session Management
✓ Injection Attacks	✓ Denial of Service (DoS) Attacks
✓ Cross-Site Scripting (XSS)	✓ Malware
✓ Cross-Site Request Forgery	✓ Third-Party Components
✓ Security Misconfiguration	



Manual Testing

High Severity Issues

1. Incorrect Flow for Lending

Description

Usually in NFT marketplaces, it is always checked first if the user has an appropriate balance, if yes then and only then are they allowed to make lending calls and sign the messages accordingly.

Here the Flow of this lending process is set a spending cap, then asked to approve the transaction and where you approve or reject you still get the feature to ask for sign and this shouldn't happen if it's been rejected. This takes place and allows me to make an offer without even having the money in my wallet.

Vulnerable File Location: <https://lending.bit5.com/lend/<id>>

Steps to Reproduce

1. Click any NFT that are available to lend in lending page
2. Click on Make an offer with value of 1000 WBNB for any random amount of date and period and interest.
3. Click on make Offer
4. Now you are asked to set a spending cap, go for any value, i tried 0.2
5. Now you are asked for a txn to approve or reject . Click on Reject
6. Next thing you will get to Sign a txn with no gas fees in it.
7. Sign that message , and you have made an offer with no money in wallet.

Recommendation

1. Fix the flow of application on lending endpoint
2. Check for the user's balance and the amount they are offering to lend prior to allowing them to make just like in NFT Marketplace.
3. Throw an insufficient fund error or request rejected once the txn is rejected by the user.

POC: <https://www.youtube.com/watch?v=uhXulmJqkV8>

Status

Resolved



2. PII Information Leaking (Email Address)

Description

PII (Personally Identifiable Information) leaking of email addresses occurs when an unauthorized person gains access to email addresses that were not intended for public or external use. Email addresses are considered PII because they can be used to identify an individual and used for targeted attacks, spamming, phishing, and other malicious activities.

Vulnerable File Location: /api/users/get?walletAddress=0x.....&refreshDate=xxxxxxx

Recommendation

1. User's personal information such as Email Addresses are not found anywhere in the profile of any user
2. However, the endpoint /api/users/get?walletAddress= is leaking such information about any user.
3. Request:

```
GET /api/users/get?  
walletAddress=0x445e5Bc684708266CB849CB653D2885AB9d12a6d&refreshDate=1683314390954  
HTTP/2  
Host: bit5.com  
Cookie: _ga=GA1.1.681166751.1683313715; cookie_consent_is_true=true;  
_ga_T4JV3KPN4X=GS1.1.1683313714.1.1.1683314300.0.0.0  
Sec-Ch-Ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"  
X-Newrelic-Id: VwECWFBXCBADU1RbDgEFVVQ=  
Tracestate: 3759640@nr=0-1-3759640-538466024-c06472930bbf59c2---1683314390967  
Traceparent: 00-5aa554c1f9fa6758bc5150e42293bd00-c06472930bbf59c2-01  
Sec-Ch-Ua-Mobile: ?0  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/113.0.0.0 Safari/537.36  
Newrelic:  
eyJ2IjpbMCwxXSwiZCI6eyJ0eSI6IkJyb3dzZXIiLCJhYyI6IjM3NTk2NDAiLCJhcCI6IjUzODQ2NjAyNCIsImIkIjoiY  
zA2NDcyOTMwYmJmNTIjMilsInRyljoiNWFlhNTU0YzFmOWZhNjc1OGJjNTE1MGU0MjI5M2JkMDAiLCJ0aSl  
6MTY4MzMxNDM5MDk2N319  
Accept: application/json, text/plain, */*  
Sec-Ch-Ua-Platform: "Windows"  
Sec-Fetch-Site: same-origin  
Sec-Fetch-Mode: cors  
Sec-Fetch-Dest: empty  
Referer: https://bit5.com/profile/0xbE77d1C7C69Ad79c40265D45c3fC1C0DCdEB4De6  
Accept-Encoding: gzip, deflate  
Accept-Language: en-US,en;q=0.9
```

Recommendation

1. Web3 helps people stay pseudonymous and this breaks it by revealing their email address.
2. Hide the email address from user who are not authorized.
3. Implement proper access controls: Limit access to email addresses to only those who need it for legitimate business purposes.
4. Encrypt email addresses: Implement encryption to protect email addresses in transit and at rest.
5. Use tokenization or pseudonymization: Instead of using actual email addresses, use tokenization or pseudonymization techniques to replace email addresses with unique identifier.

POC

```
Pretty Raw \n Actions ▾ Pretty Raw Render \n Actions ▾
1 GET /api/users/get?walletAddress=
  0xbE77d1C7C69Ad79c40265D45c3fc1c0DCdEB4De6&refreshDate=1683485193555
  HTTP/2
2 Host: bit5.com
3 Cookie: _ga=GA1.1.681166751.1683313715; cookie_consent_is_true=true;
  _ga_RC7ZBVHJD9=GS1.1.1683463519.2.1.1683464936.0.0.0; __cuid=
  2989cfb9a34748378139aa6566aeab76; amp_fefie8=
  4dd43a0a-de0d-46bd-97c1-b0060a823abaR...lgvr91g0t.lgvr91to9.7.3.a;
  connect.sid=
  s%3AwSCN2X3n18YIHh3J5geA8fEAEr8KNNjR.WCMQE2kvsD%2ByUHTe4%2Fr7mmFGHq3uWAHQ
  nMd12FDwv24k; _ga_T4JV3KPN4X=GS1.1.1683483602.6.1.1683485194.0.0.0
4 Sec-Ch-Ua: "Google Chrome";v="113", "Chromium";v="113",
  "Not-A-Brand";v="24"
5 X-Newrelic-Id: VwECWFBXBADU1RbDgEFVVQ=
6 Tracestate:
  3759640@nr=0-1-3759640-538466024-aeb4821891b5c8da---1683485204013
7 Traceparent: 00-084056d03ea21c50fc35158be90b0e00-aeb4821891b5c8da-01
8 Sec-Ch-Ua-Mobile: 20
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
10 Newrelic:
  eyJ2IjpbMCwxXSwiZCI6eyJ0eSI6IkJyb3dz2XIIiLCJhYyI6IjM3NTk2NDAiLCJhcI6IjUzO
  DQCNjAyNCIsImIkIjo1YWWiNDgyMTg5MWI1YzhkYSIsInRyIjo1MDg0MDU2ZDAzZWEyMWM1MG
  ZjMzUxNThizTkwYjBlMDAiLCJ0aSI6MTY4MzQ4NTIwNDAxM319
11 Accept: application/json, text/plain, */*
12 Sec-Ch-Ua-Platform: "Windows"
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://bit5.com/gaming
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Connection: close
1
2 Date: Sun, 07 May 2023 18:45:11 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 540
5 X-Powered-By: Express
6 Vary: Origin, Accept-Encoding
7 Access-Control-Allow-Credentials: true
8 Etag: W/"21c-QL1MbTNMC1jfLv3uKinB/ByBBLO"
9 Strict-Transport-Security: max-age=15724800; includeSubDomains
10
11 {
  "status": "success",
  "data": {
    "bio": "radhakrsna",
    "name": "radhakrsna",
    "coverPhoto": null,
    "createdAt": "2023-05-05T19:17:44.211Z",
    "id": "081e61b8-2570-4bc7-8017-0b145c5bdb1f",
    "instagram": null,
    "isActive": null,
    "isEmailVerified": null,
    "isVerified": null,
    "profilePhoto": null,
    "role": null,
    "telegram": null,
    "twitter": null,
    "walletAddress": "0xbE77d1C7C69Ad79c40265D45c3fc1c0DCdEB4De6",
    "website": "https://cdn.jsdelivr.net/gh/Moksh45/host-xss.rocks/index.js",
    "showWidgetsAsHome": false,
    "email": "radhakrsna@radhakrsna.com",
    "updatedAt": "2023-05-05T20:38:34.791Z"
  }
}
```

Status

Resolved



3. WebSocket API-KEY leaked

Description

During our testing, we identified a significant issue with the security of the web application. We found that a WebSocket API key was leaked in a file in the GitHub repository. The API key was found in the file config.js, which was accessible to anyone with access to the repository. This API key was used to authenticate the WebSocket connections and provide secure communication between the client and server.

However, the leaking of the API key in a public repository is a significant security vulnerability as it allows anyone to access the WebSocket endpoint and potentially compromise the server or access sensitive data.

Vulnerable File Location: [Github Link](#)

Recommendation

1. We recommend immediately revoking the leaked API key and generating a new one. Additionally, we recommend implementing a security policy that ensures sensitive data, such as API keys, are never committed to the repository.
3. To prevent similar issues in the future, we suggest conducting regular code reviews to identify any potential security vulnerabilities and ensuring that sensitive data is stored securely and never leaked to unauthorized parties.

POC

Backend-New / apps / websocket-app / src / websocket-authentication / websocket-authent

OzcanI Eslint formats and Websocket Event Image Paths

Code Blame 10 lines (8 loc) · 270 Bytes

```
1 import { Injectable } from '@nestjs/common';
2
3 @Injectable()
4 export class WebsocketAuthenticationService {
5   private readonly apiKeys = new Set(['9a73a0cb90b3b16d475e69090d651fe9']);
6
7   validateApiKey(apiKey: string): boolean {
8     return this.apiKeys.has(apiKey);
9   }
10 }
```

Bit5 Websocket Client
tha_dev + Follow

HTML

```
1 <html>
2   <head>
3     <script src="https://cdn.socket.io/4.3.2/socket.io.min.js"
4       integrity="sha384-KAZ4DtjNhLch0B/hxXuKjhMLVvx3bSMl155xPeINmREKRzeEm+RVP1InAn0ajQNs"
5       crossorigin="anonymous"></script>
6
7     const socket = io('https://stream.prod.bit5.com/?
8       apiKey=9a73a0cb90b3b16d475e69090d651fe9');
9
10    socket.on('connect', function() {
11      console.log('Connected');
12    });
13    socket.on('onItemListed', function(data) {
14      console.log('onItemListed', data);
15    });
16    socket.on('onItemSold', function(data) {
17      console.log('onItemSold', data);
18    });
19    socket.on('onItemTransferred', function(data) {
20      console.log('onItemTransferred', data);
21    });
22    socket.on('onItemCancelled', function(data) {
23      console.log('onItemCancelled', data);
24    });
25    socket.on('onItemReceivedBid', function(data) {
26      console.log('onItemReceivedBid', data);
27    });
28  
```

Console

```
"Disconnected"
"Connected"
"Disconnected"
"Connected"
"Connected"
>
```

Status

Resolved

Medium Severity Issues

4. Issues in JWT token being used

Description

The New Relic header JWT is being used and it has multiple issues and can have multiple impacts that we can discuss below

Issue1.a

During our testing, we identified an issue with the JWT token used for authentication. We found that the JWT token being sent in the header did not have an expiration time, which makes it possible for an attacker to use the same token indefinitely.

The absence of an expiration time for the JWT token in the header is a significant security vulnerability that could allow an attacker to manipulate new relic's analytics of user accounts. Implementing an expiration time for JWT tokens is a simple yet effective security measure that can significantly improve the web application's security posture.

The screenshot shows the OWASP ZAP tool interface. The 'Request' tab displays a POST request to `/api/users/update` with various headers and a JSON payload. The 'Response' tab shows the server's response, which includes a JSON object with `"status": "success"` and `"data": true`. This response is highlighted with a red box. The 'INSPECTOR' tab on the right shows the selected text from the response and its decoded form, both of which are also highlighted with a red box. The decoded form shows a Base64 string representing the JSON object.

Issue 1.b

During our testing, we identified an issue with the JWT header used for authentication with New Relic's analytics services. We found that the JWT header was not being verified, which means that an attacker could potentially intercept and use any user's JWT. Without proper verification of the JWT header, an attacker could potentially modify the header to impersonate a legitimate user or bypass authentication altogether. This vulnerability could lead to a data breach or unauthorized access to sensitive information.

The screenshot shows a web browser window with a Bit5 profile page and a NetworkMiner tool overlay. The profile page displays a cover photo placeholder, a 'Profile Image' section, a 'Username' input field, and a 'Bio' section. The NetworkMiner tool captures a request and response. The request shows a JWT token in the Authorization header. The response shows a successful 201 Created status with JSON data.

```

Request
Pretty Raw \n Actions
"Chromium":v="113", "Not-A.Brand":v="24"
6 X-Newrelic-Id: VwECWFBCBADU1RbDgEFVVQ=
7 Tracestate:
3759640@nr=0-1-3759640-538466024-0f009d6
14ccf338e---1683314349224
8 traceparent:
00-c6dd67f1993c9e276ff1c3f27bdc8000-0f00
9d614ccf338e-01
9 Sec-Ch-Ua-Mobile: 70
10 User-Agent: Mozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/113.0.0.0
safari/537.36
11 Newrelic:
eyJJIjpbMCwxXSwiZCI6eyJ0eSI6IkJyb3dzZXIi
LCJhYyI6IjM3NTk2NDaiLCJhcI6IjUzODQCNjAy
NCI6ImlkIjoiMGYwMD1kNjE0Y2NmMzM4ZSIyInRy
IjoiYzdkNmQ3jE5OTNjOWUYNzZmijFjM2Vyn2Jk
YzgwMDAiLCJ0aSI6MY4MzNxNDM0GtiyNH19
12 Content-Type: application/json
13 Accept: application/json
14 Sec-Ch-Ua-Platform: "Windows"
15 Origin: https://evil.com
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-Mode: cors
18 Sec-Fetch-Dest: empty
19 Referer:
https://bit5.com/profile/0xbE77d1C7C69Ad79c40265D45c3fc1C0DCdE84De6
20 Accept-Encoding: gzip, deflate
21 Accept-Language: en-US,en;q=0.9
22 Connection: close
23
24 {"status": "success", "data": true}
    
```

Response

```

HTTP/2 201 Created
Date: Fri, 05 May 2023 20:38:34 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 32
X-Powered-By: Express
Access-Control-Allow-Origin: https://evil.com
Vary: Origin, Accept-Encoding
Access-Control-Allow-Credentials: true
Etag: W/"20-LpvOmjUM2g6vtazb7wSJ1MN1rM"
Strict-Transport-Security: max-age=15724800
1 {
    "status": "success",
    "data": true
}
    
```

token still active

Recommendation

1. We recommend implementing proper verification of the JWT header to ensure that only legitimate requests are accepted by the web application. Verification of the JWT header should include validating the signature of the header, verifying the issuer, and checking the expiration time.
2. Additionally, the web application should be configured to reject any requests that include an invalid JWT header or a header that has been tampered with.

Status

Resolved

5. Cross-Origin Resource Sharing on POST Request

Description

During our testing, we identified an issue with the Cross-Origin Resource Sharing (CORS) policy used by the web application. We found that CORS was enabled on the web application's POST endpoints, which means that these endpoints could be accessed from any domain. Enabling CORS on POST endpoints can be a significant security vulnerability as it allows an attacker to send malicious requests to the endpoint from a different domain, which can lead to unauthorized access, data leakage, or other security risks.

Vulnerable File Location: <https://bit5.com/api/users/update/>

Steps to Reproduce

1. Update your profile and capture the request in Burpsuite.
Change the origin header to <https://evil.com> and send the request and in response to
2. the request you will get an
Access-Control-Allow-Origin: https://evil.com with an
3. Access-Control-Allow-Credentials: true

This can prove that it can leak any sensitive information.

POST /api/users/update HTTP/2

Host: bit5.com

Cookie: _ga=GA1.1.681166751.1683313715; cookie_consent_is_true=true;
_ga_T4JV3KPN4X=GS1.1.1683313714.1.1.1683314300.0.0.0

Content-Length: 331

Sec-Ch-Ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"

X-Newrelic-Id: VwECWFBXCBADU1RbDgEFVVQ=

Tracestate: 3759640@nr=0-1-3759640-538466024-0f009d614ccf338e----1683314349224

Traceparent: 00-c6d6d7f1993c9e276ff1c3f27bdc8000-0f009d614ccf338e-01

Sec-Ch-Ua-Mobile: ?0

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/113.0.0.0 Safari/537.36

Newrelic:

eyJ2JpbMCwxXSwiZCI6eyJ0eSI6IkJyb3dzZXIiLCJhYyI6IjM3NTk2NDAiLCJhcCI6IjUzODQ2NjAyNCIsImIkIjoiMGYwMDIkNjE0Y2NmMzM4ZSIsInRyljoiYzZkNmQ3ZjE5OTNjOWUyNzZmZjFjM2YyN2JkYzgwMDAiLCJ0aSI6MTY4MzMxNDM0OTIyNH19

Content-Type: application/json

Accept: application/json

Sec-Ch-Ua-Platform: "Windows"

Origin: <https://evil.com>

Sec-Fetch-Site: same-origin

Sec-Fetch-Mode: cors

Sec-Fetch-Dest: empty



Referer: https://bit5.com/profile/0xbE77d1C7C69Ad79c40265D45c3fC1C0DCdEB4De6

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.9

Connection: close

```
{"data": {"name": "radhakrsna", "bio": "radhakrsna", "email": "radhakrsna@radhakrsna.com", "profilePhoto": null, "coverPhoto": null, "walletAddress": "0xbE77d1C7C69Ad79c40265D45c3fC1C0DCdEB4De6"}, "signature": "0x1c581c0f34b97f362ef5acecd5d05f55376affe5e9da0878702f1167124d99627f3f37ef050ffede2960e11815fe21016da3f8937c54d2dbc83bc8385797c7ee1c"}
```

Recommendation

1. We recommend disabling CORS on the web application's POST endpoints or limiting access to these endpoints to only trusted domains. This can be achieved by configuring the web application's server to include appropriate headers, such as Access-Control-Allow-Origin, Access-Control-Allow-Methods, and Access-Control-Allow-Headers.
2. Additionally, we recommend implementing other security measures, such as input validation and rate limiting, to prevent any malicious requests from being processed by the web application's POST endpoints.

POC

The screenshot shows the OWASP ZAP interface with the 'Repeater' tab selected. The 'Request' pane displays a POST request to `/api/users/update` with various headers and a JSON payload. The 'Response' pane shows a successful `HTTP/2 201 Created` response with a JSON object containing the user data and a success status. The 'INSPECTOR' panel on the right shows the raw XML response.

```
POST /api/users/update HTTP/2
Host: bit5.com
Cookie: _ga=GAI.1.601166751.1603313715; cookie_consent_is_true=true; _ga_T4JV3KPN4X=GS1.1.1
Content-Length: 331
Sec-Ch-Ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
X-Newrelic-Id: VwECWFBNXBADU1RbDgKFVVQ=
Tracestate: 37596409nr=0-1-3759640-538466024-0f009d614ccf338e---1683314349224
Traceparent: 00-c6ded7f1993c9e276ff1c3f27bdc8000-0f009d614ccf338e-01
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) 10
Newrelic: eyJ2IjpbMCwxdBwl2CI6eyJ0e816IkJyb3dz3XIiLCJhYry16IjM3NTk2NDALCJhcC16IjUzODQCNjAyt11
Content-Type: application/json
Accept: application/json
Sec-Ch-Ua-Platform: "Windows"
Origin: https://evil.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://bit5.com/profile/0xbE77d1C7C69Ad79c40265D45c3fC1C0DCdEB4De6
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
}
{"data": {"name": "radhakrsna", "bio": "radhakrsna", "email": "radhakrsna@radhakrsna.com", "profilePhoto": null, "coverPhoto": null, "walletAddress": "0xbE77d1C7C69Ad79c40265D45c3fC1C0DCdEB4De6"}, "signature": "0x1c581c0f34b97f362ef5acecd5d05f55376affe5e9da0878702f1167124d99627f3f37ef050ffede2960e11815fe21016da3f8937c54d2dbc83bc8385797c7ee1c"}
```

Target: https://bit5.com

Status

Resolved

6. User Enumeration via /api/users/get?walletAddress= endpoint

Description

The website is vulnerable to user enumeration through the /api/users/get?walletAddress endpoint. This vulnerability allows an attacker to determine the existence of user accounts on the system by exploiting a lack of proper access controls or rate limiting mechanisms. User enumeration can have several negative consequences, including:

1. Increased vulnerability to brute-force attacks: Attackers can use the information obtained through user enumeration to launch targeted brute-force attacks on identified user accounts. By knowing valid usernames, they can focus their efforts on cracking passwords, potentially leading to unauthorized access to user accounts.
2. Account enumeration for reconnaissance: Attackers can use user enumeration as part of their reconnaissance activities. By identifying valid user accounts, they can gather intelligence about the organization's user base, potentially aiding in further targeted attacks or social engineering attempts.
3. Privacy concerns: User enumeration can expose personally identifiable information (PII) of users. This information can be misused for identity theft, phishing attempts, or other malicious activities.

Vulnerable Endpoint: [Link here](#)

Steps to Reproduce

1. Visit [here](#).
2. If the user with address exists on your platform, it will list the user's details such as username, bio, wallet address, website, etc.
3. If the user does not exist, the response will be displayed - {"status":400,"message":"User not found"} error

Recommendation

1. Implement generic messages instead of user details on the endpoint to avoid user enumeration.
2. You can also implement rate limiting mechanisms to restrict the number of requests that can be made to the endpoint within a specific time period. This helps prevent automated enumeration attempts and brute-force attacks.



POC

1. User not found error

The screenshot shows a Postman interface with a failed API request. The request URL is `/api/users/get?walletAddress=0xd4e96ef8eee8678dbff4d535e033ed1a4f7605b7&refreshDate=1683485193555`. The response status is `HTTP/2 400 Bad Request`, with the message `"status": 400, "message": "User not found"`.

```
1 GET /api/users/get?walletAddress=0xd4e96ef8eee8678dbff4d535e033ed1a4f7605b7&refreshDate=1683485193555
HTTP/2
2 Host: bit5.com
3 Cookie: _ga=GA1.1.681166751.1683313715; cookie_consent_is_true=true; _ga_RC72BVHJD9=GS1.1.1683463519.2.1.1683464936.0.0.0; __cuid=2989cfb9a34748378139aa6566aeab76; amp_fefile8=4dd43a0a-de0d-46bd-97c1-b0060a823abaR...lgvr91g0t.lgvr91to9.7.3.a; connect.sid=s%3AwSCN2X3n18YIHh3J5geA8fEAeR8KNNjR.WCMQE2kvxD%2BYuHTE4%2Fr7mmFGHq3uWAHQnMd%2FDwvs24k; _ga_T4JV3KPN4X=GS1.1.1683483602.6.1.1683485194.0.0.0
4 Sec-Ch-Ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
5 X-Newrelic-Id: VwECWFBXCBAU1RbDgEFVVQ=
```

```
1 HTTP/2 400 Bad Request
2 Date: Sun, 07 May 2023 18:45:56 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 41
5 X-Powered-By: Express
6 Vary: Origin, Accept-Encoding
7 Access-Control-Allow-Credentials: true
8 Etag: W/"29-lQ+B2glNgU953gqyhildRRMIGT0"
9 Strict-Transport-Security: max-age=15724800; includeSubDomains
10
11 {
    "status": 400,
    "message": "User not found"
}
```

2. User successfully found

The screenshot shows a successful API request. The request URL is the same as the previous one, but the wallet address is valid. The response status is `HTTP/2 200 OK`, with the message `"status": "success", "data": { ... }`.

```
1 GET /api/users/get?walletAddress=0xb277d1c7c69ad79c40265d45c3fc1c0dcde84de6&refreshDate=1683485193555
HTTP/2
2 Host: bit5.com
3 Cookie: _ga=GA1.1.681166751.1683313715; cookie_consent_is_true=true; _ga_RC72BVHJD9=GS1.1.1683463519.2.1.1683464936.0.0.0; __cuid=2989cfb9a34748378139aa6566aeab76; amp_fefile8=4dd43a0a-de0d-46bd-97c1-b0060a823abaR...lgvr91g0t.lgvr91to9.7.3.a; connect.sid=s%3AwSCN2X3n18YIHh3J5geA8fEAeR8KNNjR.WCMQE2kvxD%2BYuHTE4%2Fr7mmFGHq3uWAHQnMd%2FDwvs24k; _ga_T4JV3KPN4X=GS1.1.1683483602.6.1.1683485194.0.0.0
4 Sec-Ch-Ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24"
5 X-Newrelic-Id: VwECWFBXCBAU1RbDgEFVVQ=
6 Tracestate: 3759640@nr=0-1-3759640-538466024-aeb4821891b5c8da---1683485204013
7 Traceparent: 00-084056d03ea21c50fc35158be90b0e00-aeb4821891b5c8da-01
8 Sec-Ch-Ua-Mobile: 70
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36
10 Newrelic: eyJ2IjpzbWwxXSwiZCI6eyJ0eSI6IkJyb3dzZXIiLCJhYyI6IjM3NTk2NDALCJhcCI6IjUz0DQCNjAyNCisImIkIjoiYWWiNDgyMTg5MWI1YzhkYSTsInRyIjoiMDg0MDU2ZDAzZWEyMWM1MGZjMzUxNThiZTkwyjB1MDA1LC70asI6MTY4MzQ4NTIwNDAxM319
11 Accept: application/json, text/plain, /*
12 Sec-Ch-Ua-Platform: "Windows"
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://bit5.com/gaming
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
```

```
1 HTTP/2 200 OK
2 Date: Sun, 07 May 2023 18:45:11 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 540
5 X-Powered-By: Express
6 Vary: Origin, Accept-Encoding
7 Access-Control-Allow-Credentials: true
8 Etag: W/"Zlc-QhLM8TNMCljfLv3uKiNb/ByBBi0"
9 Strict-Transport-Security: max-age=15724800; includeSubDomains
10
11 {
    "status": "success",
    "data": {
        "bio": "radhakrsna",
        "name": "radhakrsna",
        "coverPhoto": null,
        "createdAt": "2023-05-05T19:17:44.211Z",
        "id": "081e61b8-2570-4bc7-8017-0b145c5bdb1f",
        "instagram": null,
        "isActive": null,
        "isEmailVerified": null,
        "isVerified": null,
        "profilePhoto": null,
        "role": null,
        "telegram": null,
        "twitter": null,
        "walletAddress": "0xb277d1c7c69ad79c40265d45c3fc1c0dcde84de6",
        "website": "https://cdn.jsdelivr.net/gh/Moksh45/host-xss.rocks/index.js",
        "showWidgetsAsHome": false,
        "email": "radhakrsna@radhakrsna.com",
        "updatedAt": "2023-05-05T20:38:34.791Z"
    }
}
```

Status

Resolved



7. Multiple Deprecated Libraries in yarn.lock

Description

yarn.lock Stores files that can be useful for the dependency of the application. This is used for locking the dependency with the installed version. It will install the exact latest version of that package in your application and save it in package. This arises a problem if the dependency used has an exploit in the version mentioned. It can create a backdoor for an attacker.

Vulnerable File Location:

engine.io	update to 6.4.2
xml2js	update to 0.5.0
request	update to latest

Recommendation

1. Update all the above Mentioned Dependencies
2. Remove any Library Not needed.

Impact

Multiple of these libraries have public exploits and CVE-registered issues that have been patched and can help your application stay more secure from any dependency-vulnerable issues.

Status

Resolved



8. Insecure Direct Object Reference (IDOR) in POST /api/users/favoriteItems/create Endpoint

Description

The POST request endpoint `/api/users/favoriteItems/create` is found to be vulnerable to Insecure Direct Object Reference (IDOR) attacks. Specifically, the body parameter `from` containing the wallet address is susceptible to manipulation, allowing an attacker to like NFTs from the victim's account. The vulnerability arises from the lack of appropriate access control mechanisms for the `from` parameter. As a result, an attacker can manipulate the value of "from" parameter with a different user's wallet address to impersonate a victim's account and perform the action of liking NFTs on their behalf. By modifying the wallet address in `from` parameter to a valid victim's account address, the attacker can bypass any authorization checks and associate the favorite item with the victim's account, giving the appearance that the victim has liked the NFT when, in reality, they may not have done so. Attackers can exploit the vulnerability to like NFTs from a victim's account without their consent or knowledge. This can lead to false representations of the victim's preferences, potentially affecting their reputation or causing confusion among other users.

Vulnerable Endpoint: <https://bit5.com/api/users/favoriteItems/create>

Steps to Reproduce

1. Send the following POST request with any user's wallet address in the `from` parameter in the body.

POST `/api/users/favoriteItems/create` HTTP/2

Host: test.bit5.com

```
{"from":"0x445e5Bc684708266CB849CB653D2885AB9d12a6d","eventType":"like","to":"0xeb721aBDa93aF4c7513cDf01ae52187D042E50b3/53","itemId":"53","collectionAddress":"0xeb721aBDa93aF4c7513cDf01ae52187D042E50b3"}
```

2. Check the NFT with address `0xeb721aBDa93aF4c7513cDf01ae52187D042E50b3/53`.
3. It will be liked by the address you added in the `from` parameter.

Recommendation

1. Implement strict authorization and access control mechanisms to ensure that users can only perform actions on their own resources. Validate the ownership or permission of the "from" parameter before allowing the liking action.
2. Implement a proper Authorization Header which validates who is executing the request from which account.

POC

The screenshot shows a web browser displaying a Bit5 application. The application has a dark theme with a cartoon character profile picture. A message at the top says "You are using Testnet. Mint Now". Below the character are several interaction buttons: a heart icon with "1", a comment icon, an upvote icon, and a downvote icon. The text "undefined#53" is visible below these buttons. To the right of the browser is a NetworkMiner tool window. The "Repeater" tab is selected. The "Request" section shows a POST request to "/api/users/favoriteItems/create" with various headers and a JSON payload. The "Response" section shows a successful HTTP/2 201 Created response with a JSON object containing event details. The "Pretty" button is highlighted in both sections.

```
POST /api/users/favoriteItems/create HTTP/2.0
Host: test.bit5.com
Cookie: _ga=GAI.1.681166751.1683313715; co...
Content-Length: 204
Sec-Ch-Ua: "Google Chrome";v="113", "Chrom...
Accept: application/json
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win...
Sec-Ch-Ua-Platform: "Windows"
Origin: https://test.bit5.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://test.bit5.com/item/0x61/0xeb...
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
From: 0x445e58c684708266CB849CB853D288
Event-Type: like
To: 0xeb721aBDa93aF4c7513cDf01ae52187D
ItemId: 53
CollectionAddress: 0xeb721aBDa93aF4c75
```

```
HTTP/2 201 Created
Date: Sun, 07 May 2023 11:45:09 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 396
X-Powered-By: Express
Access-Control-Allow-Origin: https://test.bit5.com
Vary: Origin, Accept-Encoding
Access-Control-Allow-Credentials: true
Etag: W/"18c-klltclmJiyPc+pyiXixVSR+HyE"
Strict-Transport-Security: max-age=15724800
Status: success
Data: {
  "id": "40bcd4c4-4ee6-8a6e-d353db53d288",
  "collectionAddress": "0xeb721aBDa93aF4c7513cDf01ae52187D",
  "createdAt": "2023-05-07T11:45:09.002Z",
  "eventType": "like",
  "from": "0x445e58c684708266CB849CB853D288",
  "itemId": "53",
  "price": null,
  "to": "0xeb721aBDa93aF4c7513cDf01ae52187D",
  "txHash": null,
  "updatedAt": "2023-05-07T11:45:09.002Z",
  "orderId": null
}
```

Status

Resolved

Low Severity Issues

9. Cross-Site Request Forgery to Delete a Live Game Session of users

Description

The endpoint /game-events/delete-session is vulnerable to a CSRF vulnerability that allows an attacker to force a victim to end his current game session. This will not allow user on your platform to play a game comfortably. An attacker can send a simple link /game-events/delete-session to a user. As soon as the victim clicks on this link, his current game session will be deleted and his current score will be recorded as his highest score.

Vulnerable Endpoint: <https://bit5.com/game-events/delete-session>

Steps to Reproduce

1. Go to <https://bit5.com>
2. Login inside your account
3. Go to Gaming
4. To reproduce this issue, start a Quiz game
5. Play the game
6. Visit this link in a new browser tab - <https://bit5.com/game-events/delete-session>. You will find the following JSON data as response - {"status":"success","data":true} which indicates your current active game session is deleted.
7. Go back to your game and you won't be able to play it anymore.

Recommendation

1. Implement an anti-CSRF token. A CSRF token can be the same UUID used to log user inputs in the game. Example: https://bit5.com/game-events/register-event/54280d61-bb5e-4b5c-8c7c-d13d50df85ed/select_answer-0-0
2. Use POST instead of GET request to execute sensitive actions which lie under the CRUD operations.

POC

The screenshot shows the Bit5 [Beta] website interface. At the top, there's a navigation bar with links for Marketplace, Finance, Gaming, Launchpad, Services, and a search bar. Below the navigation is a banner encouraging users to "Play Games With Community" and "Join Our Challenges". On the left, there are two game cards: "Good Old 2048" and "Endless Quizz". The main area displays a game session titled "Select right answers to catch up with top scores!". It shows a fail count of "Fails: 1 / 3" and a score of "Score: 0". A question asks, "Which of these American cities has fewer than 1,000,000 people?", with options: San Francisco, California; Phoenix, Arizona; San Antonio, Texas; and Philadelphia, Pennsylvania. The "San Francisco, California" option is highlighted with a red border.

This screenshot is similar to the one above, showing the same game session and question. The "San Francisco, California" option is now highlighted with a red border, indicating it has been selected.

A screenshot of a browser window showing a successful API response. The URL is bit5.com/game-events/delete-session. The response body is {"status": "success", "data": true}.

Status

Resolved

10. CSRF to Force a User to Play a Move In a Game

Description

The GET request endpoint `/game-events/register-event/3fc5431d-random-uid-6777/move-left` is found to be vulnerable to Cross-Site Request Forgery (CSRF) attacks. This vulnerability can be exploited if the attacker possesses knowledge of the unique identifier (ID) `3fc5431d-random-uid-6777`. The vulnerability arises from the lack of any protective measures against CSRF attacks. This means that an attacker who knows the ID `3fc5431d-random-uid-6777` can create a malicious website or craft a malicious request that, when accessed by a victim who is authenticated in the application, automatically triggers the GET request to play a move for the specified event ID.

Vulnerable Endpoint: <https://bit5.com/game-events/delete-session>

Steps to Reproduce

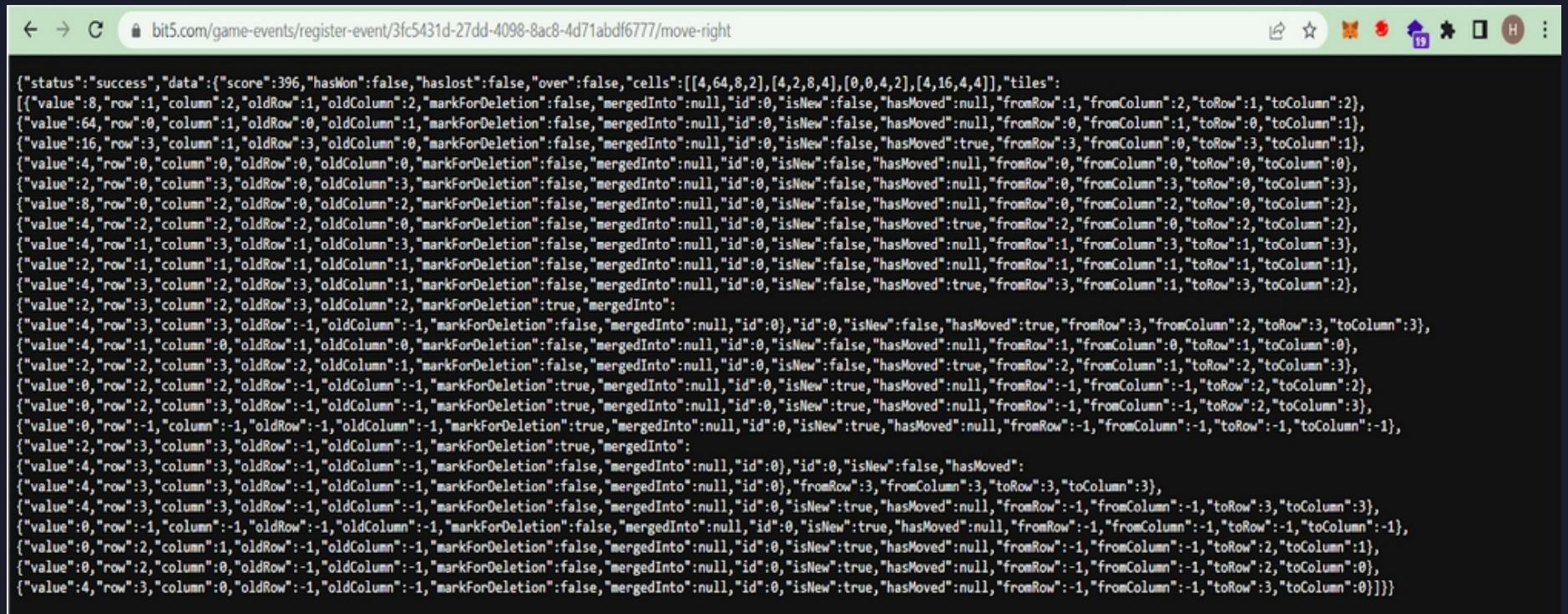
1. Go to <https://bit5.com>
2. Login inside your account
3. Go to Gaming
4. To reproduce this issue, start a Quiz game
5. Play the game
6. Intercept the request using a proxy tool such as Burp Suite when you execute any action in the game. You will find a GET Request to `/game-events/register-event/` endpoint.
7. Generate a CSRF PoC of that request and save the code in a `poc.html` file.
8. Open the `poc.html` file in your browser. A corresponding action will be executed in your current game session.

Recommendation

Implement an anti-CSRF token. Implement strong and reliable Cross-Site Request Forgery (CSRF) protection mechanisms. This can be achieved by using techniques such as including anti-CSRF tokens in the requests or verifying the origin and referer headers of incoming requests



POC



```
{"status": "success", "data": {"score": 396, "hasWon": false, "haslost": false, "over": false, "cells": [[[4, 64, 8, 2], [4, 2, 8, 4], [0, 0, 4, 2], [4, 16, 4, 4]]], "tiles": [{"value": 8, "row": 1, "column": 2, "oldRow": 1, "oldColumn": 2, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": 1, "fromColumn": 2, "toRow": 1, "toColumn": 2}, {"value": 64, "row": 0, "column": 1, "oldRow": 0, "oldColumn": 1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": 0, "fromColumn": 1, "toRow": 0, "toColumn": 1}, {"value": 16, "row": 3, "column": 1, "oldRow": 3, "oldColumn": 0, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": true, "fromRow": 3, "fromColumn": 0, "toRow": 3, "toColumn": 1}, {"value": 4, "row": 0, "column": 8, "oldRow": 0, "oldColumn": 0, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": 0, "fromColumn": 0, "toRow": 0, "toColumn": 0}, {"value": 2, "row": 0, "column": 3, "oldRow": 0, "oldColumn": 3, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": 0, "fromColumn": 3, "toRow": 0, "toColumn": 3}, {"value": 8, "row": 0, "column": 2, "oldRow": 0, "oldColumn": 2, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": 0, "fromColumn": 2, "toRow": 0, "toColumn": 2}, {"value": 4, "row": 2, "column": 2, "oldRow": 2, "oldColumn": 0, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": true, "fromRow": 2, "fromColumn": 0, "toRow": 2, "toColumn": 2}, {"value": 4, "row": 1, "column": 3, "oldRow": 1, "oldColumn": 3, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": 1, "fromColumn": 3, "toRow": 1, "toColumn": 3}, {"value": 2, "row": 1, "column": 1, "oldRow": 1, "oldColumn": 1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": 1, "fromColumn": 1, "toRow": 1, "toColumn": 1}, {"value": 4, "row": 3, "column": 2, "oldRow": 3, "oldColumn": 1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": true, "fromRow": 3, "fromColumn": 1, "toRow": 3, "toColumn": 2}, {"value": 2, "row": 3, "column": 2, "oldRow": 3, "oldColumn": 2, "markForDeletion": true, "mergedInto": {"value": 4, "row": 3, "column": 3, "oldRow": -1, "oldColumn": -1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": true, "fromRow": 3, "fromColumn": 2, "toRow": 3, "toColumn": 3}, {"value": 4, "row": 1, "column": 0, "oldRow": 1, "oldColumn": 0, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": 1, "fromColumn": 0, "toRow": 1, "toColumn": 0}, {"value": 2, "row": 2, "column": 3, "oldRow": 2, "oldColumn": 1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": true, "fromRow": 2, "fromColumn": 1, "toRow": 2, "toColumn": 3}, {"value": 0, "row": 2, "column": 2, "oldRow": -1, "oldColumn": -1, "markForDeletion": true, "mergedInto": {"value": 0, "row": 2, "column": 1, "oldRow": -1, "oldColumn": -1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": true, "fromRow": 2, "fromColumn": 2, "toRow": 2, "toColumn": 1}, {"value": 0, "row": 0, "column": 3, "oldRow": -1, "oldColumn": -1, "markForDeletion": true, "mergedInto": {"value": 0, "row": 0, "column": 2, "oldRow": -1, "oldColumn": -1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": true, "hasMoved": null, "fromRow": -1, "fromColumn": -1, "toRow": -1, "toColumn": -1}, {"value": 2, "row": 3, "column": 3, "oldRow": -1, "oldColumn": -1, "markForDeletion": true, "mergedInto": {"value": 2, "row": 3, "column": 2, "oldRow": -1, "oldColumn": -1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": true, "hasMoved": null, "fromRow": -1, "fromColumn": -1, "toRow": -1, "toColumn": -1}, {"value": 4, "row": 3, "column": 0, "oldRow": -1, "oldColumn": -1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": -1, "fromColumn": -1, "toRow": -1, "toColumn": -1}, {"value": 4, "row": 2, "column": 0, "oldRow": -1, "oldColumn": -1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": -1, "fromColumn": -1, "toRow": -1, "toColumn": -1}, {"value": 0, "row": 2, "column": 1, "oldRow": -1, "oldColumn": -1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": -1, "fromColumn": -1, "toRow": -1, "toColumn": -1}, {"value": 0, "row": 0, "column": 1, "oldRow": -1, "oldColumn": -1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": -1, "fromColumn": -1, "toRow": -1, "toColumn": -1}, {"value": 0, "row": 0, "column": 0, "oldRow": -1, "oldColumn": -1, "markForDeletion": false, "mergedInto": null, "id": 0, "isNew": false, "hasMoved": null, "fromRow": -1, "fromColumn": -1, "toRow": -1, "toColumn": -1}]}]}
```

Status

Resolved



11. CLICKJACKING

Description

A Clickjacking vulnerability was identified on the website <https://bit5.com>. Clickjacking, also known as a UI redress attack, is a technique that tricks users into clicking on malicious content or performing unintended actions without their knowledge or consent. In this case, the vulnerability allows an attacker to overlay or embed malicious content on top of the legitimate Bit5 website, potentially leading to various forms of abuse or exploitation.

Vulnerable Endpoint:

<https://bit5.com/>

<https://lending.bit5.com/>

<https://test.bit5.com/gaming>

Steps to Reproduce

1. Create a malicious web page or use an existing website under your control.
2. Modify the malicious web page's HTML to include an iframe that loads <https://bit5.com>:

```
<html>
<body>
  <h1>Malicious Website</h1>
  <iframe src="https://bit5.com"></iframe>
</body>
</html>
```
3. Host the malicious web page on a web server.
4. Open the link where the malicious web page is hosted in your browser. You will find your website embedded in an iframe.

Recommendation

1. Set the X-Frame-Options HTTP response header to deny or sameorigin. This will prevent the website from being loaded inside an iframe on malicious websites.
2. Implement a strong Content Security Policy that includes the frame-ancestors directive with 'self' or specific trusted domains to restrict which websites can embed Google's content.

POC

The screenshot shows the Clickjacker web application interface. At the top, there's a navigation bar with links for 'ABOUT' and 'TEST'. Below it, a URL input field contains 'https://bit5.com' and a 'TEST' button next to it. A share link 'Share result via url: https://clickjacker.io/test?url=https://bit5.com' is displayed with a 'COPY' button. The main area is titled 'Test Results:' and lists the following information:

Site:	https://bit5.com
IP Address:	188.166.193.62
Time:	Sun May 07 2023 12:02:45 GMT+0000 (Coordinated Universal Time)
X-Frame-Options:	✖ Missing header
CSP Header (Frame-Ancestors):	✖ Missing anti-framing policy

Below the table, there's a note about toggling an object in an iframe and a counter for total scans: 'Total scans so far: 1,520,904'.

Status

Resolved



Informational Issues

12. Order Cancellation Information Disclosure Vulnerability

Description

When a user cancels an NFT order, the backend successfully cancels the order. However, the front end fails to update and still displays the order value and details, creating a discrepancy between the backend and the user interface. This can potentially lead to confusion and information disclosure as users may believe the order is still active.

Vulnerable File Location: <https://bit5.com/activity>

Steps to Reproduce

1. Go to any NFT and place your bid.
2. Go to Activity tab and cancel you bid
3. After canceling, your bid activity would still be displayed and won't be removed from the database.

Recommendation

Ensure that the frontend is properly synchronized with the backend when canceling orders. When an order is canceled, update the frontend to reflect the updated status and remove the canceled order from the user interface.

Status

Resolved



Closing Summary

In this report, we have considered the security of the BIT5 Nft Marketplace, Lending page, and corresponding API's along with WebSocket. We performed our audit according to the procedure described above.

Some issues of High, low and Informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Dapp audit is not a security warranty, investment advice, or an endorsement of the BIT5 Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multi-step process. One audit cannot be considered enough. We recommend that the BIT5 Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies.

We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.



850+
Audits Completed



\$16B
Secured



800K
Lines of Code Audited



Follow Our Journey





Audit Report

May, 2023

For

Bit 5



QuillAudits

📍 Canada, India, Singapore, UAE, UK

🌐 www.quillaudits.com

✉️ audits@quillhash.com