QuillAudits

Audit Report
May, 2023

For

Bit5

Table of Content

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 Medium

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

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 aving 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

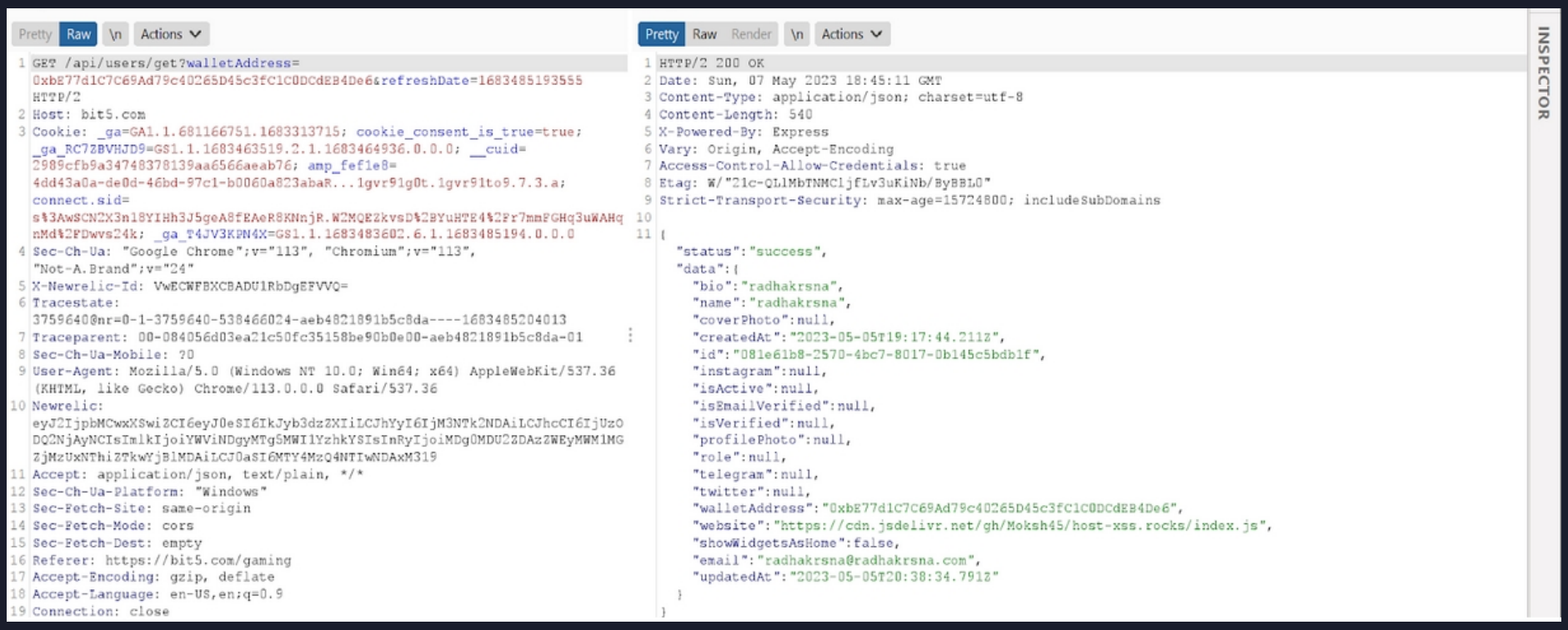
1. User's personal information such as Email Addresses are not found anywhere in the profile of any user
2. However, the endpoint /api/users/get?walletAddress= is leaking such information about any user.
3. Request:

```
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:
eyJ2IjpbMCwxXSwiZCI6eyJ0eSI6IkJyb3dzZXIiLCJhYyI6IjM3NTk2NDAiLCJhcCI6IjUzODQ2NjAyNCIsImlkIjoiY
zA2NDcyOTMwYmJmNTljMiIsInRyIjoiNWFhNTU0YzFmOWZhNjc1OGJjNTE1MGU0MjI5M2JkMDAiLCJ0aSI
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
2. address.
   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

GET /api/users/get?walletAddress=
0xbE77d1C7C69Ad79c40265D45c3fC1C0DCdEB4De6&refreshDate=1683485193555
HTTP/2
Host: bit5.com
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_fef1e8=
4dd43a0a-de0d-46bd-97c1-b0060a823abaR...1gvr91g0t.1gvr91to9.7.3.a;
connect.sid=
s%3AwSCN2X3n18YIHh3J5geA8fEAeR8KNnjR.W2MQE2kvsD%2BYuHTE4%2Fr7mmFGHq3uWAHq
nMd%2FDwvs24k; _ga_T4JV3KPN4X=GS1.1.1683483602.6.1.1683485194.0.0.0
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-aeb4821891b5c8da----1683485204013
Traceparent: 00-084056d03ea21c50fc35158be90b0e00-aeb4821891b5c8da-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:
eyJ2IjpbMCwxXSwiZCI6eyJ0eSI6IkJyb3dzZXIiLCJhYyI6IjM3NTk2NDAiLCJhcCI6IjUzO
DQ2NjAyNCIsImlkIjoiYWViNDgyMTg5MWI1YzhkYSIsInRyIjoiMDg0MDU2ZDAzZWEyMWM1MG
ZjMzUxNThiZTkwYjBlMDAiLCJ0aSI6MTY4MzQ4NTIwNDAxM319
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/gaming
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
Pretty Raw Render \n Actions

HTTP/2 200 OK
Date: Sun, 07 May 2023 18:45:11 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 540
X-Powered-By: Express
Vary: Origin, Accept-Encoding
Access-Control-Allow-Credentials: true
Etag: W/"21c-QL1MbTNMCljfLv3uKiNb/ByBBL0"
Strict-Transport-Security: max-age=15724800; includeSubDomains

{
  "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"
  }
}
```

INSPECTOR

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.

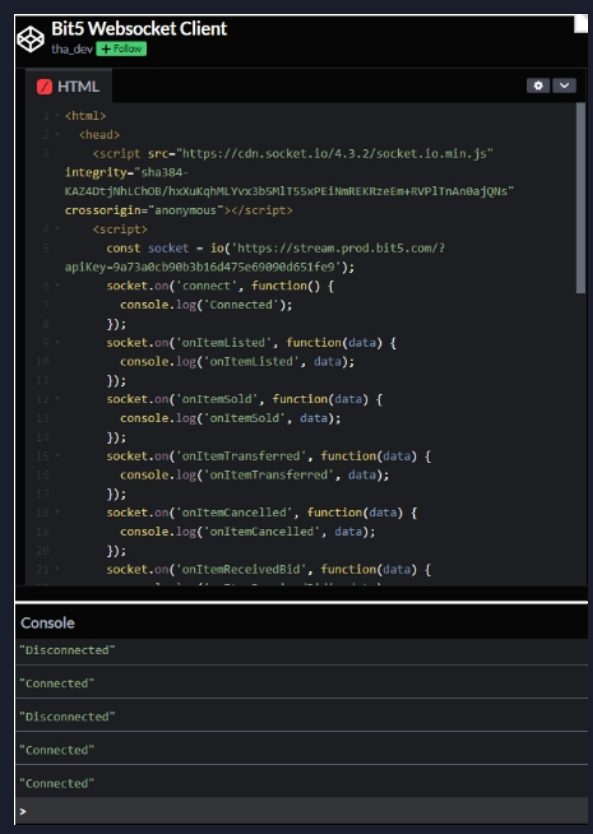
POC

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

Ozcanl 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  }
```



Status

Resolved

Medium Severity Issues

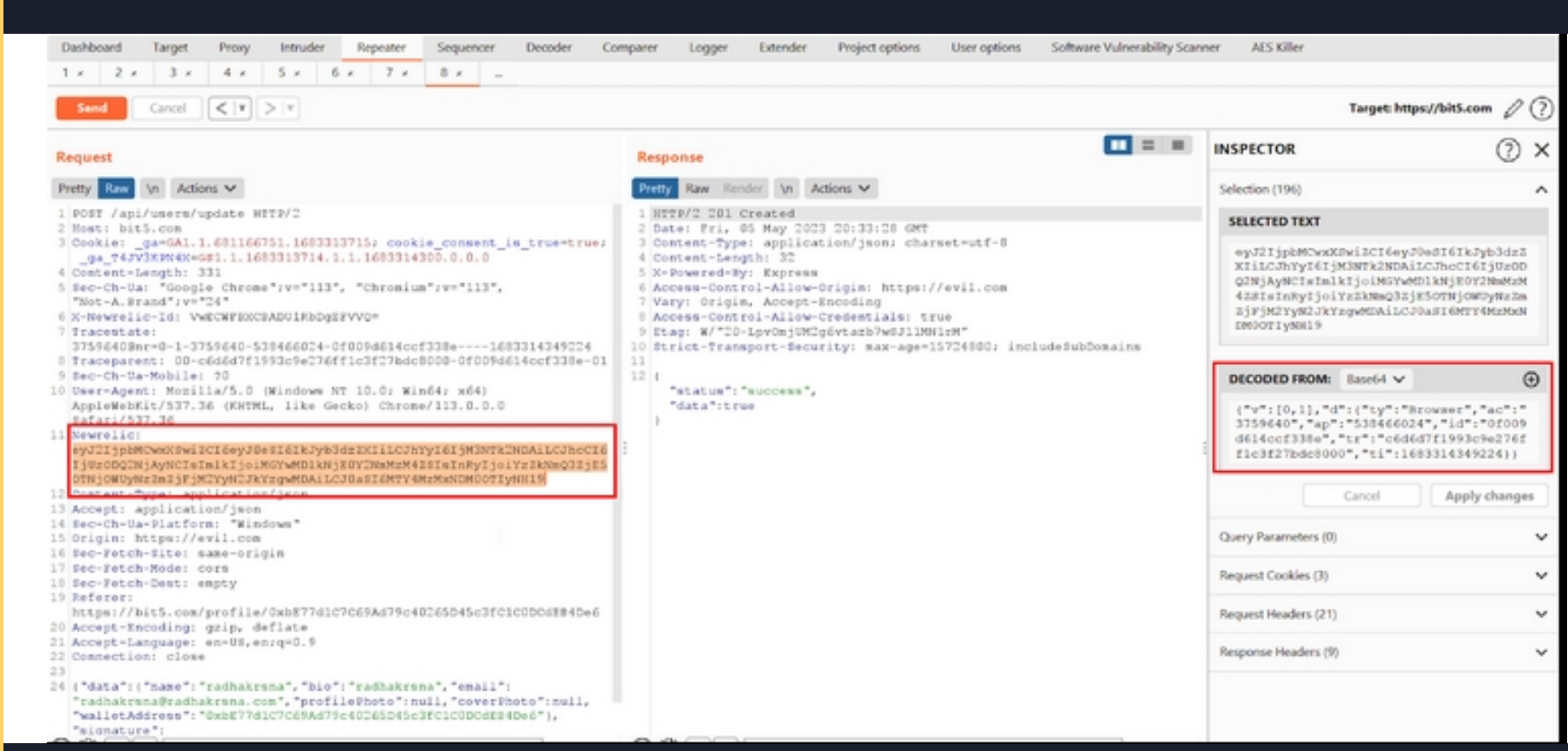
4. Issues in JWT token being used

Description

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

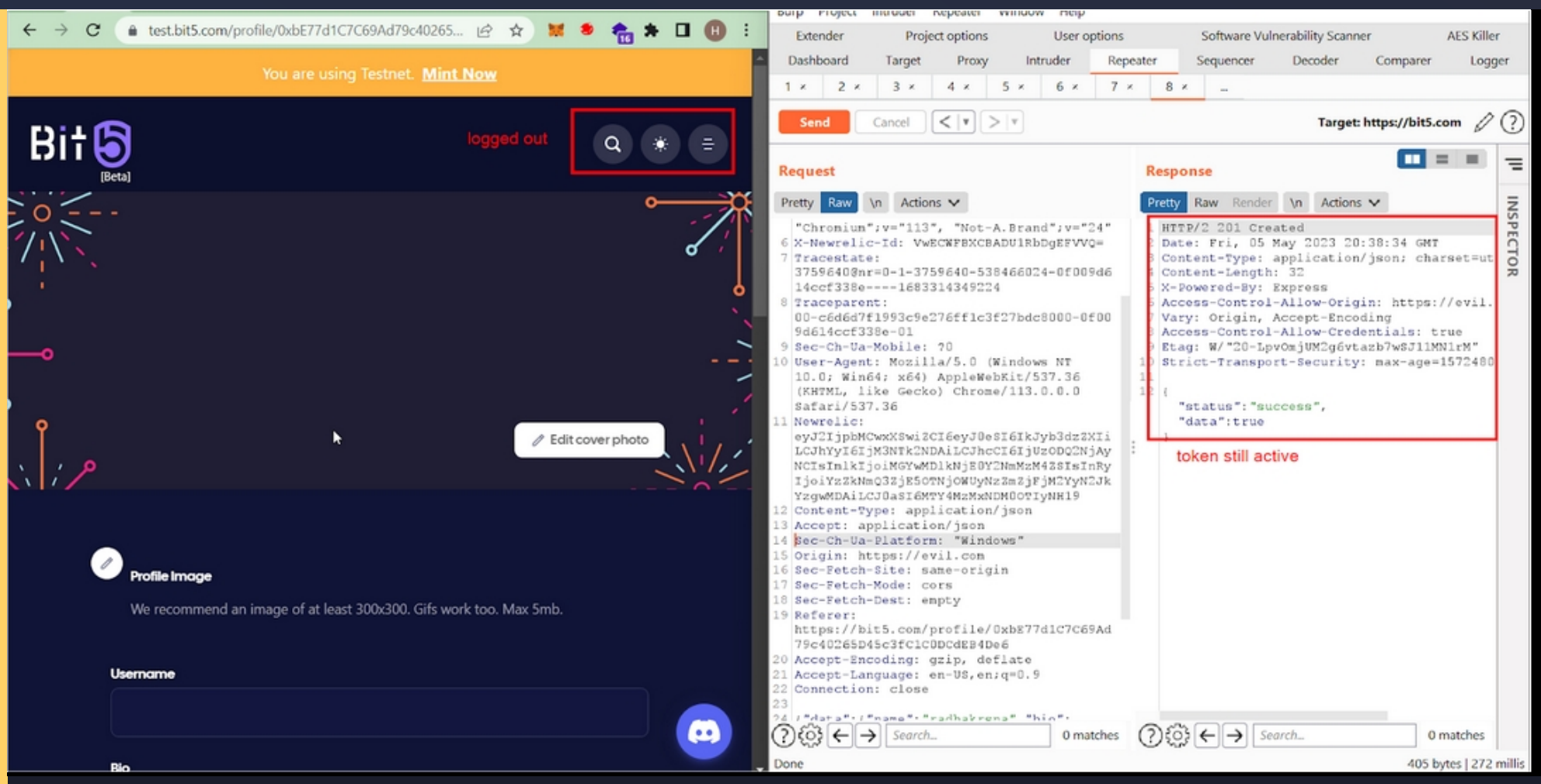
Issue1.a

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 relics 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.



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.



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:
eyJ2IjpbMCwxXSwiZCI6eyJ0eSI6IkJyb3dzZXIiLCJhYyI6IjM3NTk2NDAiLCJhcCI6IjUzODQ2NjAyNCIsImlkIjoi
MGYwMDlkNjE0Y2NmMzM4ZSIsInRyIjoiYzZkNmQ3ZjE5OTNjOWUyNzZmZjFjM2YyN2JkYzgwMDAiLCJ0aS
I6MTY4MzMxNDM0OTIyNH19
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
```

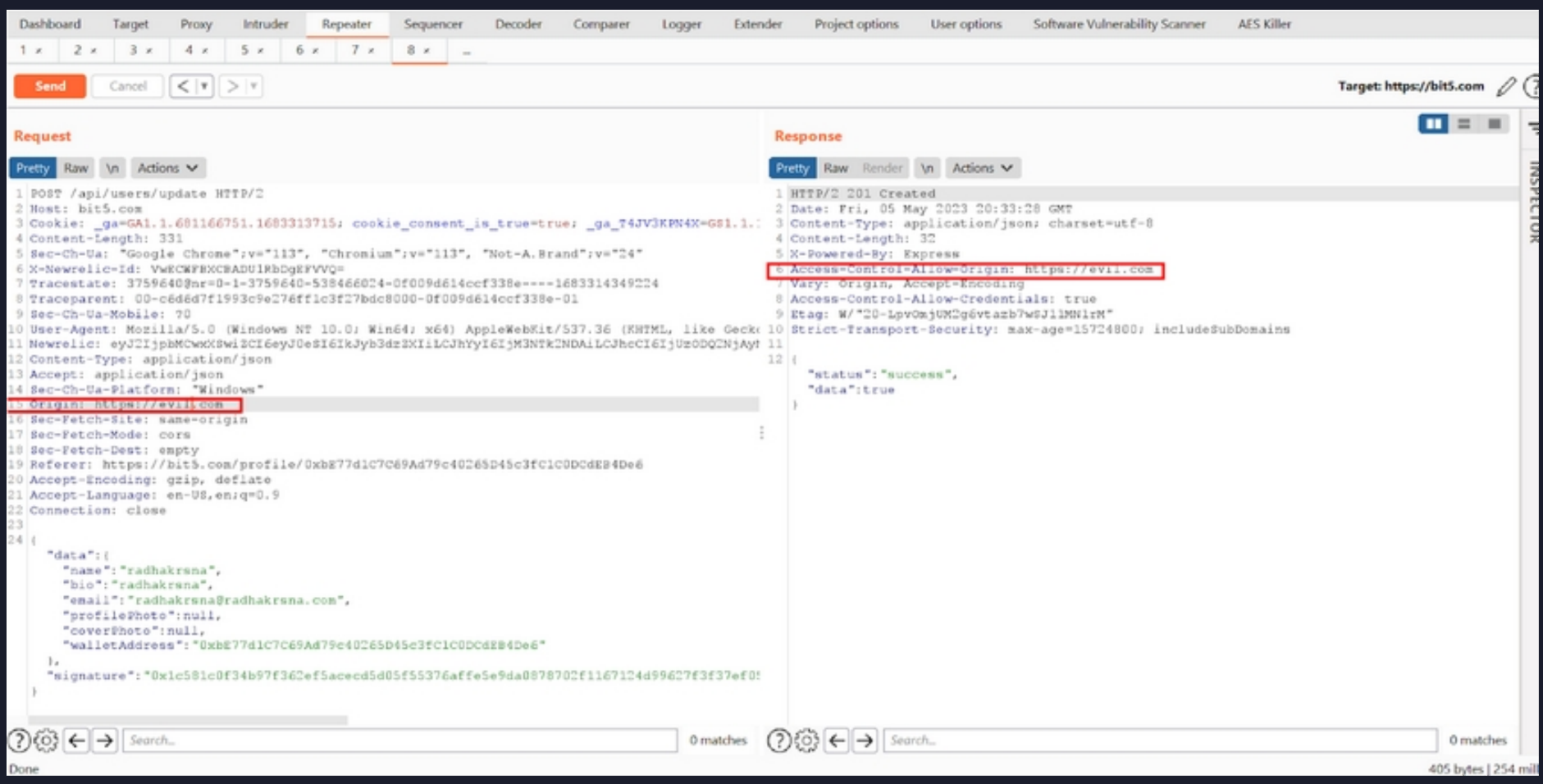
Referer: https://bit5.com/profile/0xbE77d1C7C69Ad79c40265D45c3fC1C0DCdEB4De6
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{"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



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

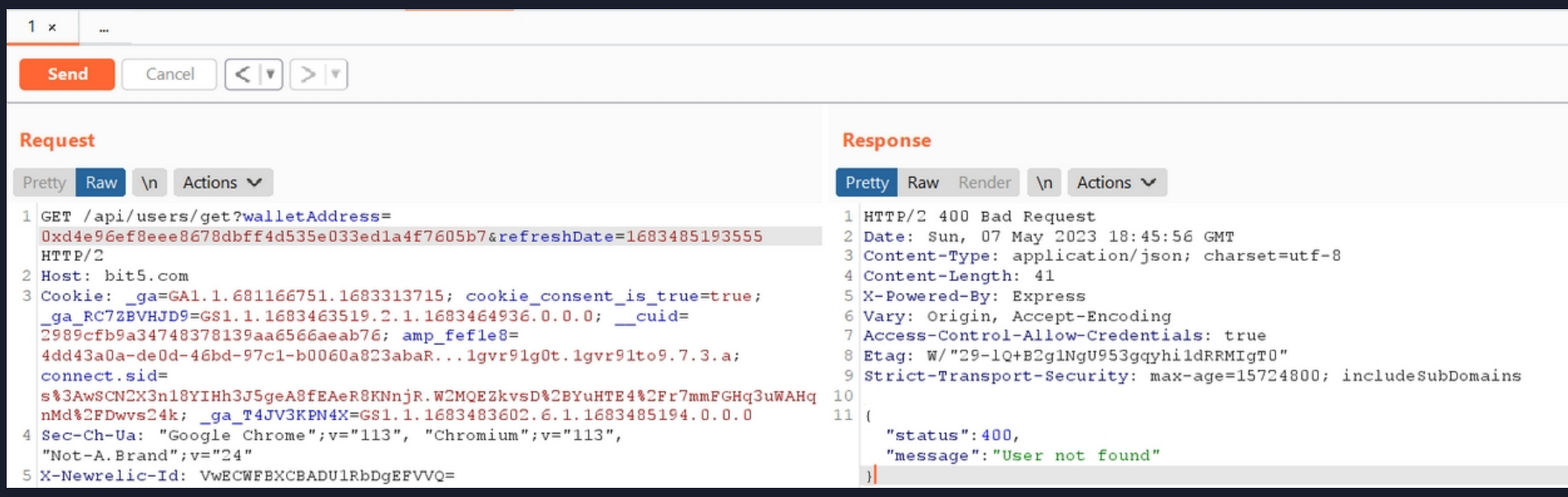
1. Visit *here.*
2. If the user with address exists on your platform, it will list the user’s details such as username, bio, wallet address, website, etc.
3. If the user does not exists, the response will be display - {"status":400,"message":"User not found"} error

Recommendation

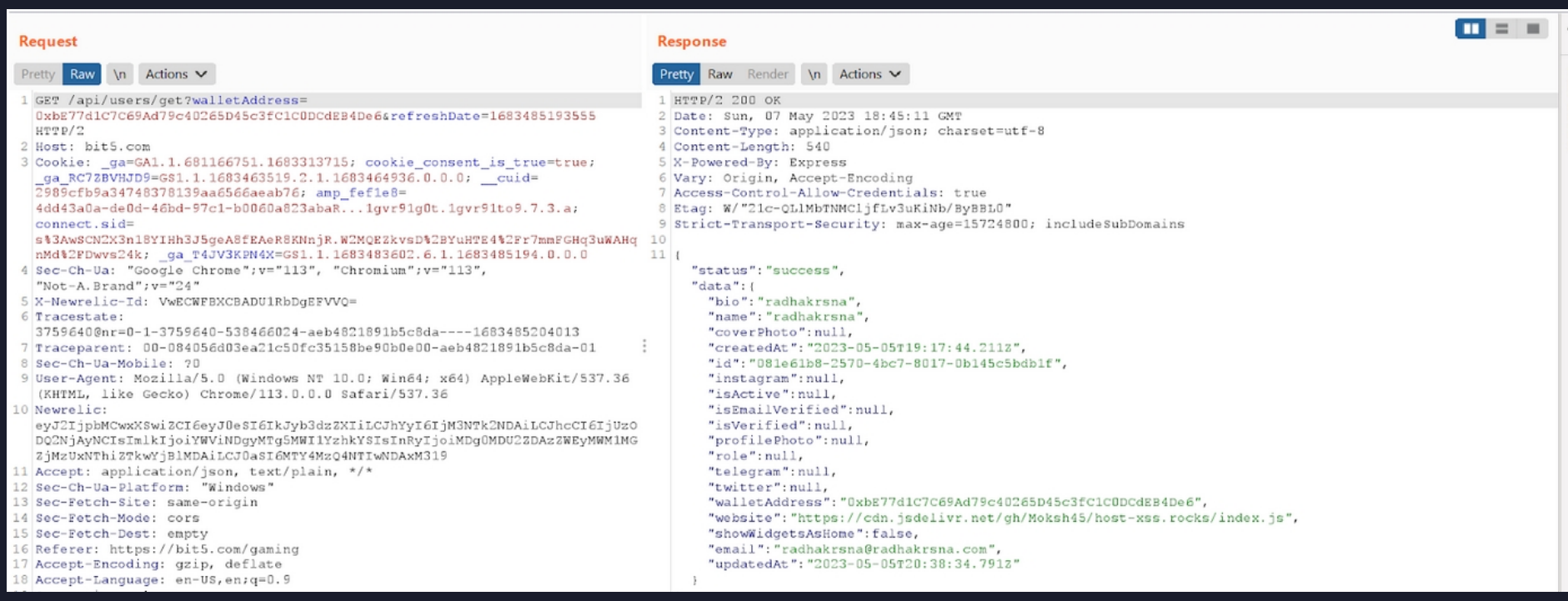
1. Implement generic messages instead of user details on the endpoint to avoid user enumeration.
2. You can also implement rate limiting mechanisms to restrict the number of requests that can be made to the endpoint within a specific time period. This helps prevent automated enumeration attempts and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

POC

1.User not found error



2. User successfully found



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*

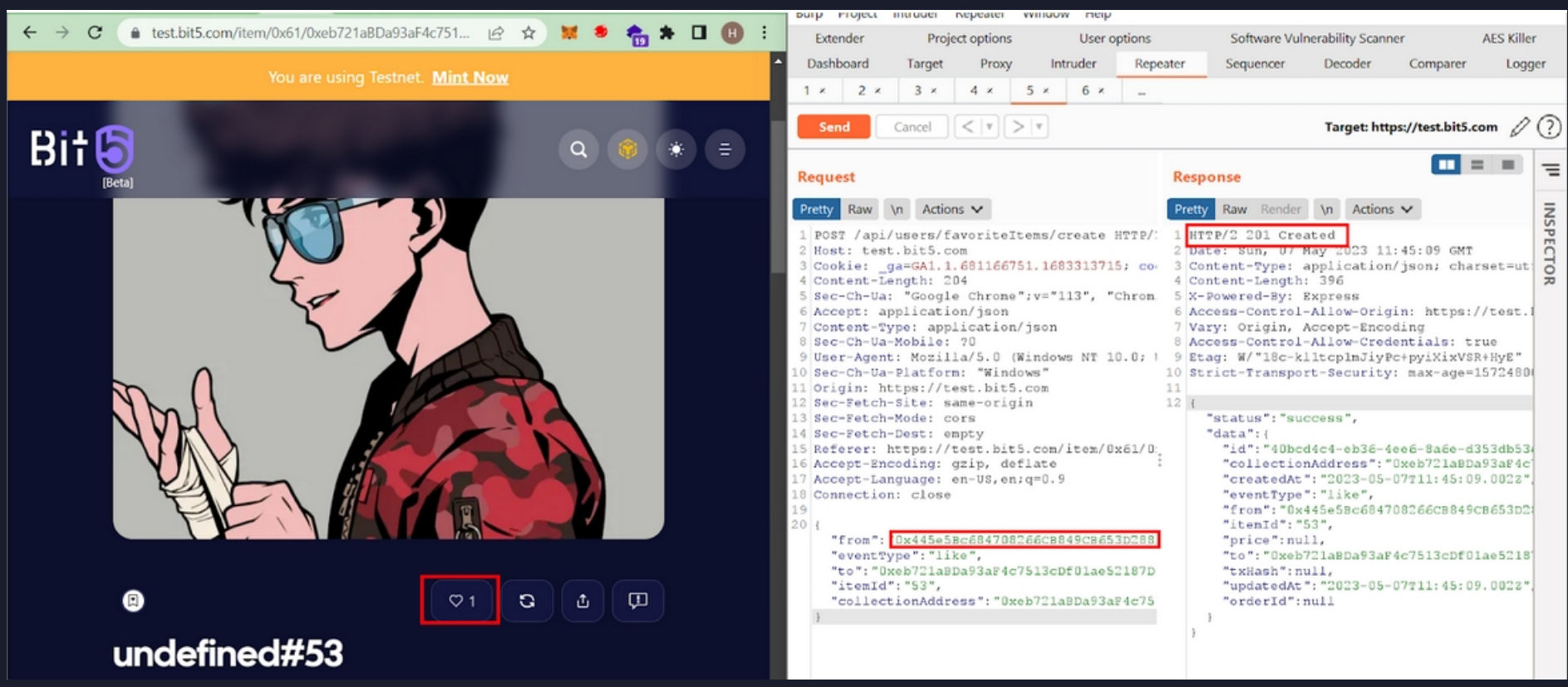
Steps to Reproduce

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



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*

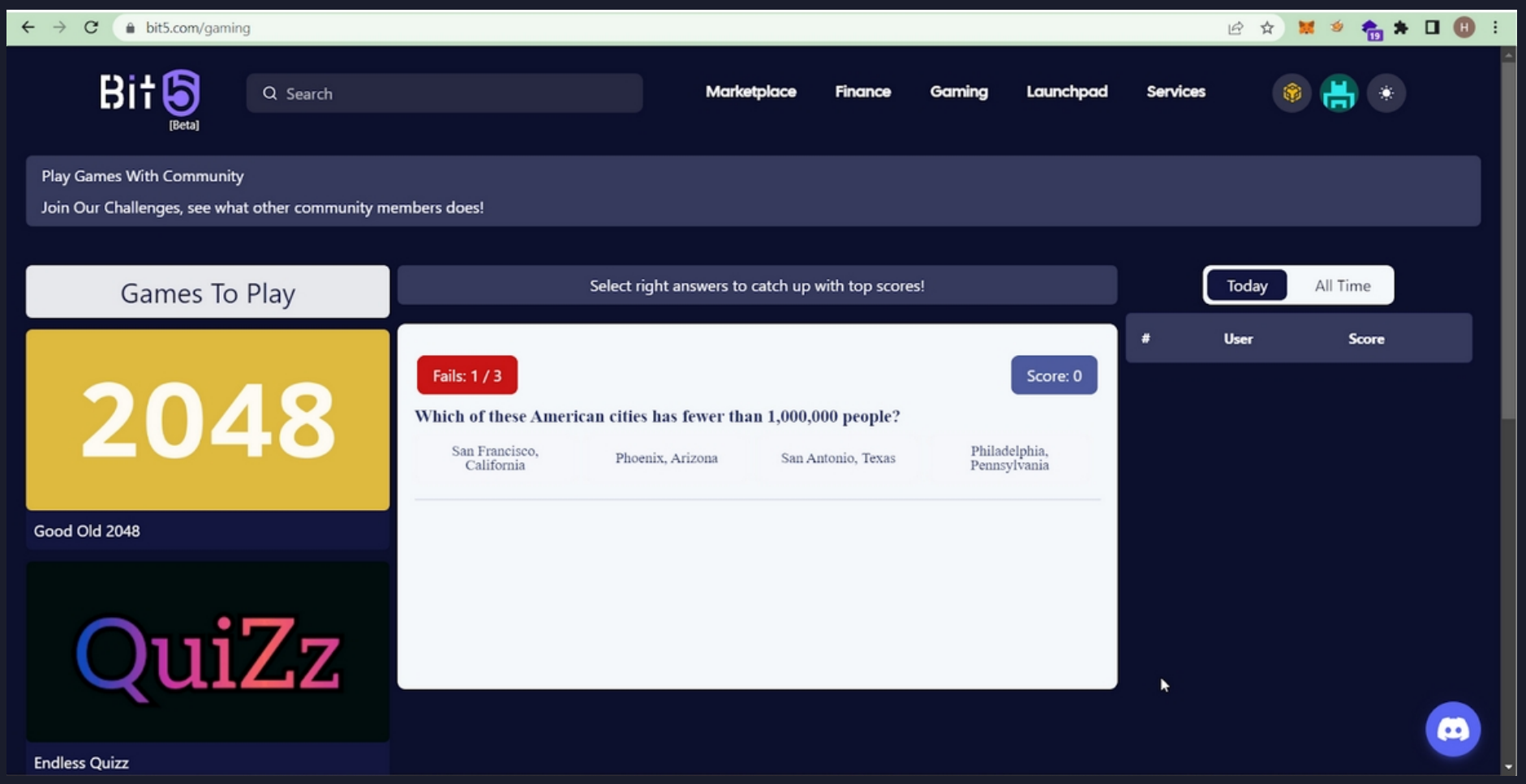
Steps to Reproduce

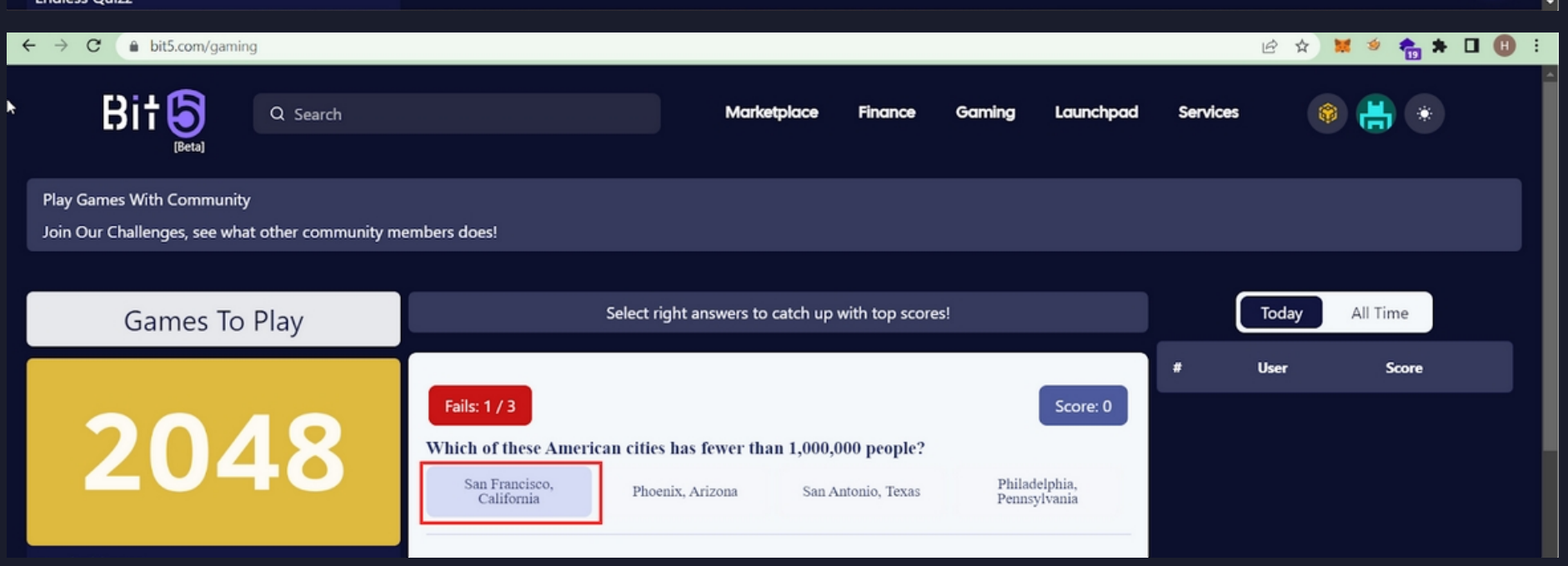
1. Go to https://bit5.com
2. Login inside your account
3. Go to Gaming
4. To reproduce this issue, start a Quiz game
5. Play the game
6. Visit this link in a new browser tab - https://bit5.com/game-events/delete-session. You will find the following JSON data as response - {"status":"success","data":true} 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.

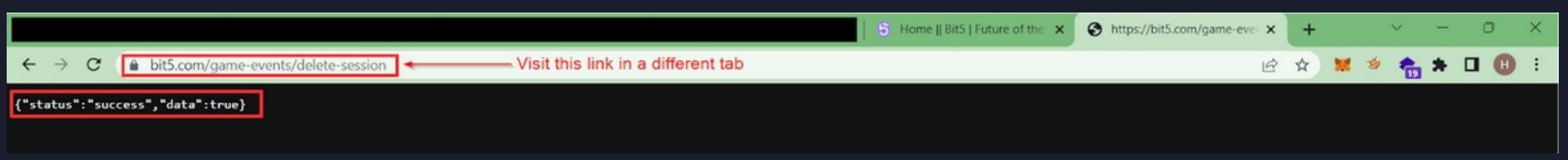
Recommendation

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







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*

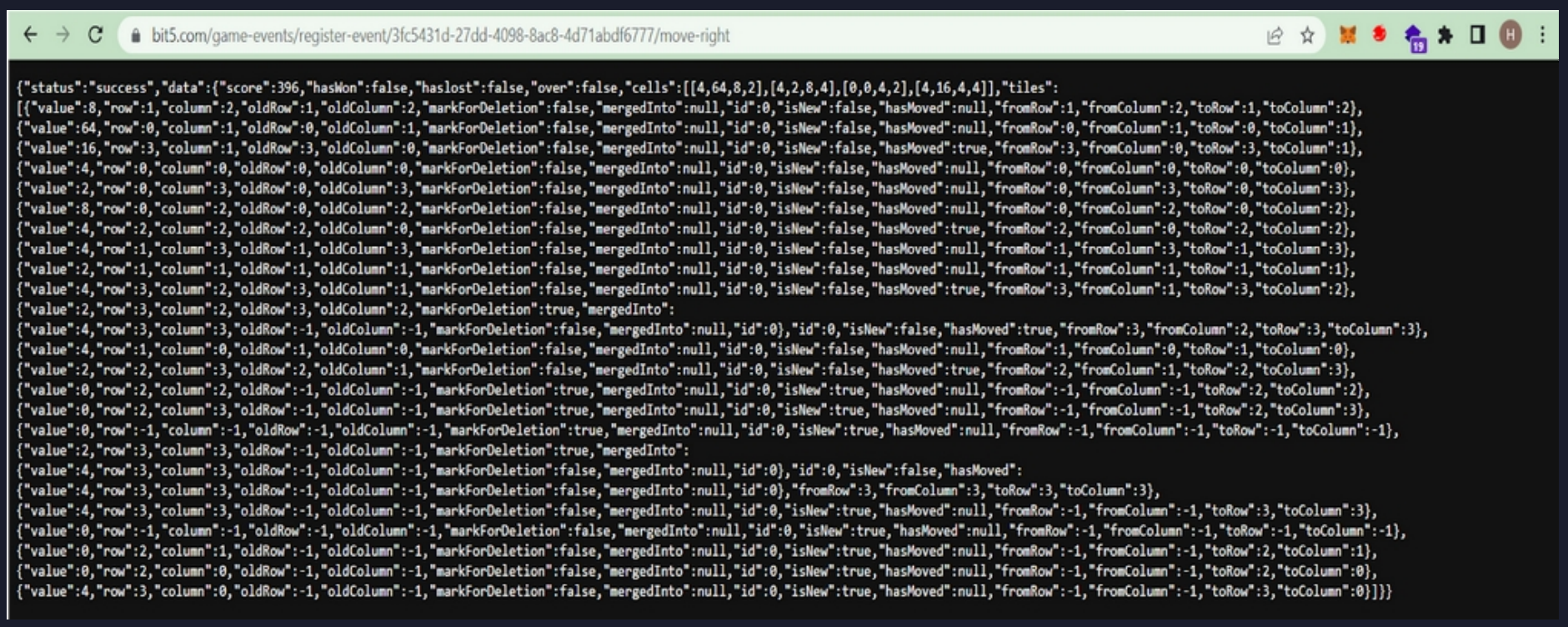
Steps to Reproduce

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

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*:

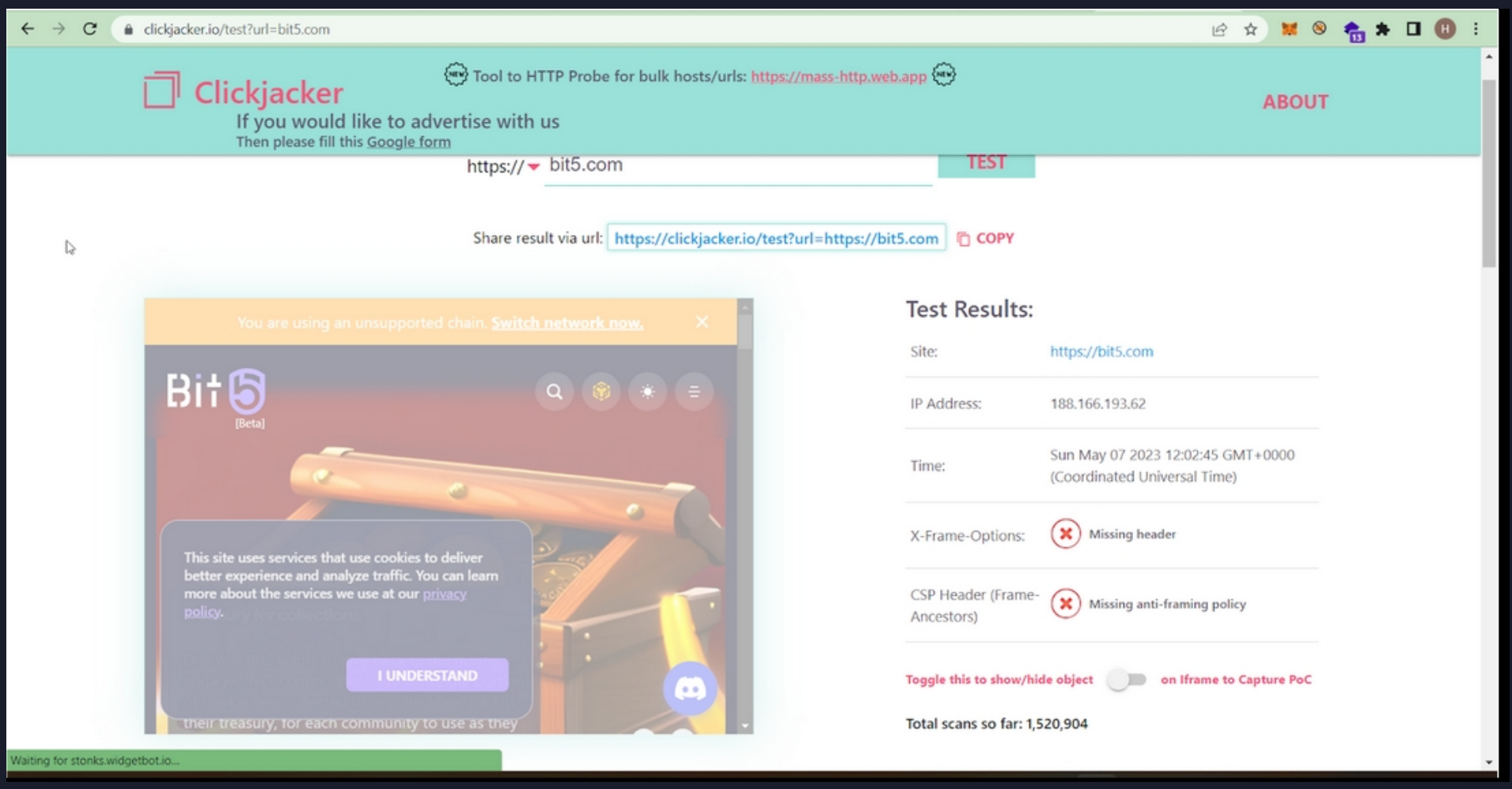
```
<html>
  <body>
    <h1>Malicious Website</h1>
    <iframe src="https://bit5.com"></iframe>
  </body>
</html>
```

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



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

Bit5

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com