QuillAudits

AUDIT REPORT

January 2025

For

Dabba

Table of Content

Executive Summary

Project Name	Dabba Network
Project URL	*https://dabba.com/*
Overview	Dabba Inc has deployed thousands of hotspots across India at retail, residential, and commercial locations serving users with super fast, super cheap internet. Demand for data in India is roughly doubling every 18 months and Dabba is building a new data-only network for a data-hungry generation.
Audit Scope	The Scope of the Audit is to analyse the security, code quality and correctness of the Source Code and Web App (Explorer) of dabba network.
Contracts in Scope	*https://new-explorer-dabba-com.vercel.app/* (Source Code) *quillaudits.new-explorer.wifidabba.com* 53bfc81224dd995f3d523149312a87cde417a2e6 (Source Code) quillaudits-wifidabba-api b3169f320088c0b0a55bc2915ea6eaf943a0cdd8
Review 1	20th December 2024 - 1st January 2025
Updated Code Received	7th January 2025
Final Review	10th January 2025

Number of Issues per Severity

High	3 (30%)
Medium	5 (50%)
Low	2 (20%)
Informational	0 (0%)

Severity

Issues

	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	**3**	**5**	**2**	0
Acknowledged	0	0	0	0
Partially Resolved	0	0	0	0

Checked Vulnerabilities

- Improper Authentication
- Improper Resource Usage
- Improper Authorization
- Insecure File Uploads
- Insecure Direct Object References
- Client-Side Validation Issues
- Rate Limit
- Input Validation
- Injection Attacks
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery
- Security Misconfiguration
- 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
- Denial of Service (DoS) Attacks
- Malware
- Third-Party Components

And More..

Techniques and Methods

Throughout the pentest of application, 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, Acunetix etc.
- Platform testing and configuration
- Error handling and data validation testing
- Encryption-related protection testing

Tools and Platforms used for Pentest:

Burp Suite	**Acunetix**	**Nmap**
DNSenum	**Neucli**	**Metasploit**
Dirbuster	**Nabbu**	**Horusec**
SQLMap	**Turbo Intruder**	**Postman**
Netcat	**Nessus**	And Many more..

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

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 Severity Issues

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 Severity Issues

Low-level severity issues can cause minor impact and 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 four 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.

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.

High Severity Issues

1. Unsanitized input in NoSQL query

Resolved

Description

Using unsanitized data in SQL queries exposes your application to SQL injection attacks. This vulnerability arises when user input, request data, or any externally influenced data is directly passed into a SQL query function without proper sanitization.
Here the parameters - hotspotId is not sanitized and could lead to SQL injection vulnerability.

Vulnerable Location

quillaudits-wifidabba-api/src/routes/explorer/hotspots.js:686

Recommendation

Do not include raw, unsanitized user input in NoSQL queries. This practice can lead to NoSQL injection vulnerabilities.

```
const User = require("../models/user")
 const newUser = new User(req.body); // unsafe
```

Do sanitize all input data before using it in NoSQL queries. Ensuring data is properly sanitized can prevent NoSQL injection attacks.

```
 const User = require("../models/user");

 username = req.params.username;
 User.findOne({ name: username.toString() });
```

POC

src > routes > explorer > JS hotspots.js > hotspotUptime > query

hotspotId Aa ab .* ? of 19

```
666  async function hotspotUptime(hotspotId, days = 7) {
670      const query = `
671          SELECT
672              uptime_date,  LEAST((SUM(heartbeats) /  200 * 100), 100) uptime_percentage
673          FROM
674              (
675                  SELECT
676                      dabba_id,
677                      DATE(created_at) uptime_date,
678                      LEAST(COUNT(created_at), 200) AS heartbeats
679                  FROM
680                      dabba_heartbeats
681                  WHERE
682                      uptime_date >= '${formatDateToYYYYMMDD(startDate)}'
683                          AND
684                      uptime_date < '${formatDateToYYYYMMDD(endDate)}'
685                          AND
686                      dabba_id = '${hotspotId}'
687                  GROUP BY
688                      uptime_date, dabba_id
689                  ORDER BY
690                      uptime_date ASC
691              ) dabba_uptime
692          GROUP BY
693              uptime_date
694          `;
695
696      const result = await clickhouseClient.query({
697          query,
698      });
699      const resultJSON = await result.json();
```

2. Hard-coded Secrets

Resolved

Description

It is identified that API Keys, Passwords, Secrets are hard-coded in the .env environment file. Hard-coding secrets in a project opens them up to leakage.

Vulnerable Location

quillaudits-wifidabba-api/.env.example

Recommendation

Do not hard-code secrets in committed code. Instead, use environment variables and a secret management system.

POC

```
$ .env.example
31  JWT_PRIVATE_KEY=
32  JWT_EXPIRATION_SECONDS=2592000
33  BCRYPT_SALT_ROUNDS=10
34
35  #Generic token
36  WD_TOKEN="01b6c5f3-b124-44af-9a18-787ee174b57d"
37
38  #Signup token expiration in seconds
39  PASSWORD_RESET_TOKEN_EXPIRATION=1800
40
41  #Invoice details
42  INVOICE_SERIAL_TEXT="#INV-"
43  INVOICE_START_NUMBER=1000
44
45  #Razorpay payment gateway keys
46  RAZORPAY_KEY_ID=
47  RAZORPAY_KEY_SECRET=
48  #NOTE: Set random key same as that of from Razorpay website webook.
49  RAZORPAY_WEBHOOK_SECRET=
50
51  #Stripe payment gateway keys
52  STRIPE_PUBLISH_KEY=
53  STRIPE_SECRET_KEY=
54  STRIPE_BANDWIDTH_PAYMENT_WEBHOOK_SECRET=whsec_yuql5uwRpKscYC0hX5sd3sfD6kpQYPkq
55  STRIPE_DABBA_PAYMENT_WEBHOOK_SECRET=whsec_yuql5uwRpKscYC0hX5sd3sfD6kpQYPkq
56
57  #Login attempts
58  MAX_LOGIN_ATTEMPTS=8
59  MAX_LOGIN_ATTEMPTS_BLOCK_HOURS=4
60
61  #Expiry is 30 minutes
62  SMS_TOKEN_EXPIRATION_IN_SECONDS=1800
```

3. Loss of funds due to misconfigured data handling

Resolved

Description

The following API endpoint is fetched while staking coins - https://test-b2b-api.wifidabba.com/api/dbt-staking/stake
Here, the parameter amount and durationInDays both accept scientific values. The application has restrictions to limit users to stake only up to 1 year (365 days). However, this could be bypassed using scientific numbers. On supplying a huge scientific value such as 1e9, the application processes the requests and deducts the amount from the wallet but does not stake it leading to loss of user's funds.

The API Endpoint
https://test-b2b-api.wifidabba.com/api/explorer/my-hotspots/payments/lco-maintenance-fee-topup to top up maintenance services. Here the application again accepts scientific numbers instead of integers and successfully processes the request. Hence, an attacker can pay lower and get higher epochs in return. Currently the top up feature is not working properly. This vulnerability is included based on analyzing the API response.

Vulnerable Location

https://test-b2b-api.wifidabba.com/api/dbt-staking/stake

https://test-b2b-api.wifidabba.com/api/explorer/my-hotspots/payments/lco-maintenance-fee-topup

Impact

Loss of user funds.
Users can cover more epoch points by paying less.

Recommendation

Properly verify the input from the user end. Only accept integers as valid input.

POC

Request

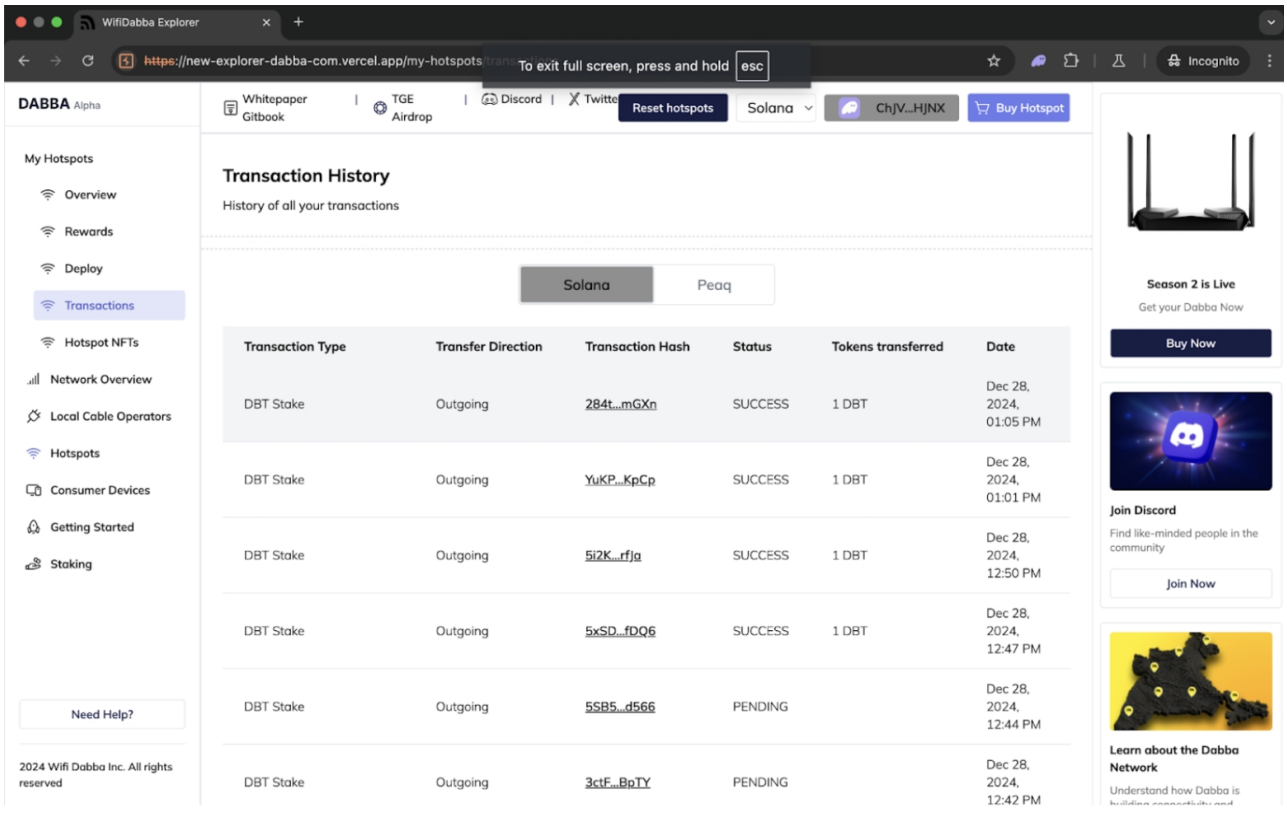
Pretty Raw Hex

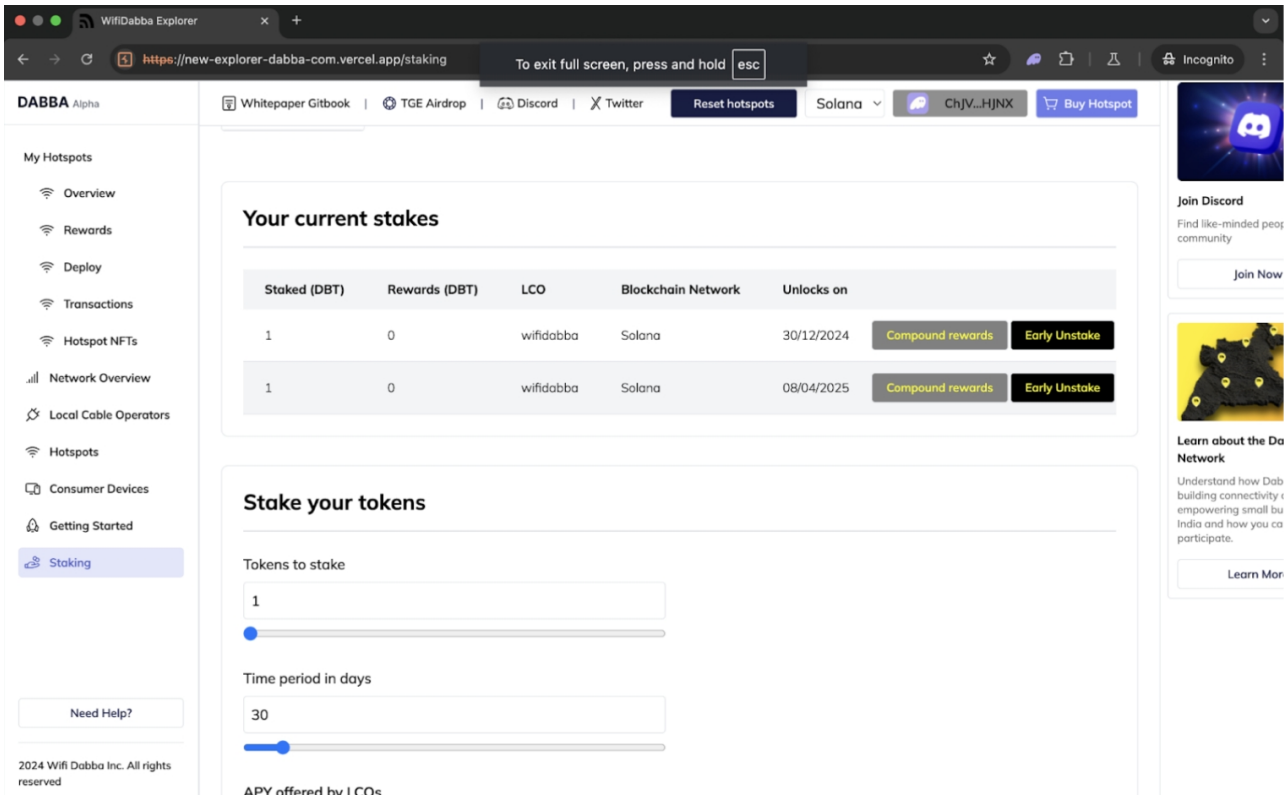
```
POST /api/dbt-staking/stake HTTP/1.1
Host: test-b2b-api.wifidabba.com
Content-Length: 197
Sec-Ch-Ua-Platform: "macOS"
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzZjMDJmOTM5MmVhNDUwZD
Q3NTc4MmUiLCJyb2xlcyI6WyJDVVNUT01FUiJdLCJpYXQiOjE3MzUzNjk1MzMsImV4cCI6M
TczNzk2MTUzM30.hrknzvBwht-u-gVI_9gFUrxwXD0ilP1IsOKdUF2IG7E
Accept-Language: en-GB,en;q=0.9
Sec-Ch-Ua: "Chromium";v="131", "Not_A Brand";v="24"
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140
Safari/537.36
Accept: */*
Origin: https://new-explorer-dabba-com.vercel.app
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://new-explorer-dabba-com.vercel.app/
Accept-Encoding: gzip, deflate, br
Priority: u=4, i
Connection: keep-alive

{
    "signature":
    "YuKPUL3KUp5sFDsXHiy6fWCEKAoKZJ5m5ziEDPp3VVGdT5NESZizGFYQga1WpKtPw
    WXLDdY7wgv3tWEtQ6iKpCp",
    "amount":"1000000000",
    "durationInDays":1e9,
    "lcoCode":"wifidabba",
    "blockchainNetwork":"Solana"
}
```

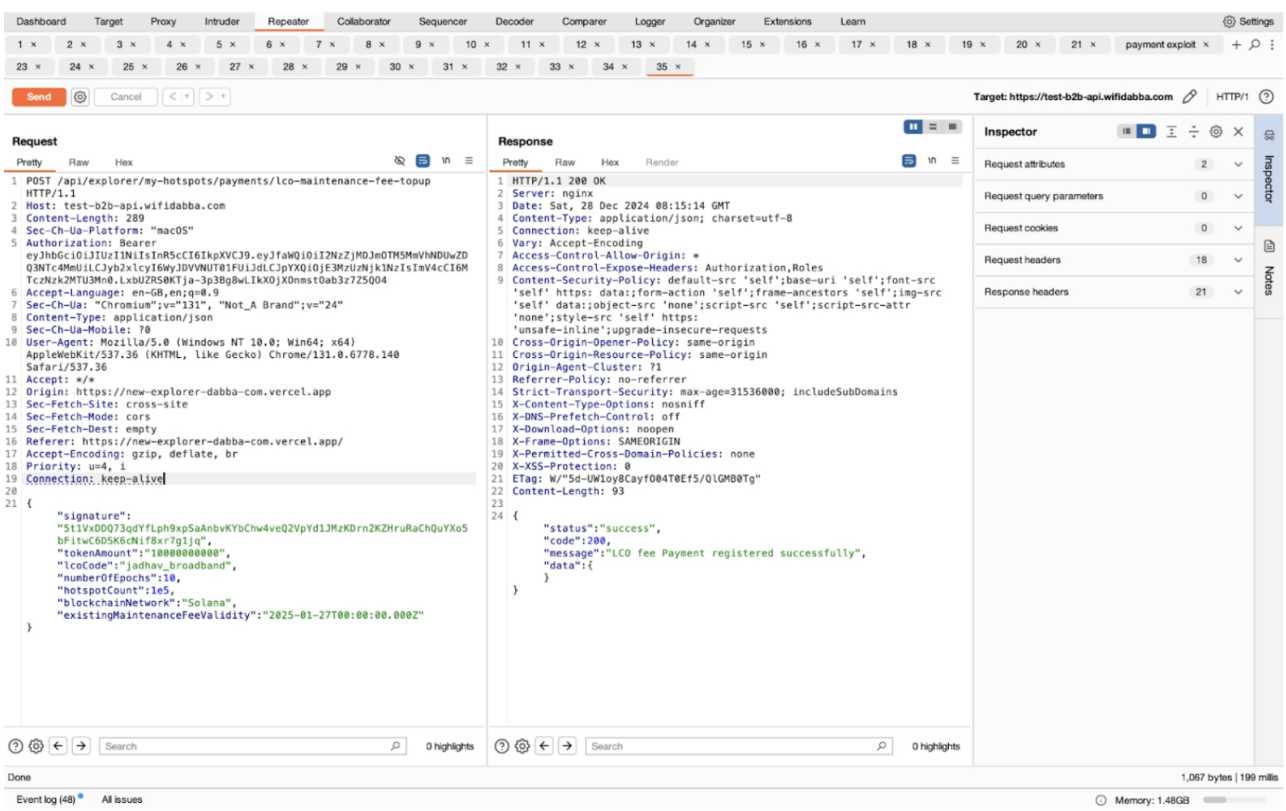
Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Server: nginx
Date: Sat, 28 Dec 2024 07:31:07 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Vary: Accept-Encoding
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Authorization,Roles
Content-Security-Policy: default-src 'self';base-uri 'self';font-src
'self' https: data:;form-action 'self';frame-ancestors 'self';img-src
'self' data:;object-src 'none';script-src 'self';script-src-attr
'none';style-src 'self' https:
'unsafe-inline';upgrade-insecure-requests
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Resource-Policy: same-origin
Origin-Agent-Cluster: ?1
Referrer-Policy: no-referrer
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-DNS-Prefetch-Control: off
X-Download-Options: noopen
X-Frame-Options: SAMEORIGIN
X-Permitted-Cross-Domain-Policies: none
X-XSS-Protection: 0
ETag: W/"47-QFzPS5zYII/xpDAAMQ44QwUstJs"
Content-Length: 71

{
    "status":"success",
    "code":200,
    "message":"Stake successfull",
    "data":{
    }
}
```




WifiDabba Explorer
https://new-explorer-dabba-com.vercel.app/staking
To exit full screen, press and hold esc
Incognito
DABBA Alpha
Whitepaper Gitbook
TGE Airdrop
Discord
Twitter
Reset hotspots
Solana
ChJV...HJNX
Buy Hotspot
My Hotspots
Overview
Rewards
Deploy
Transactions
Hotspot NFTs
Network Overview
Local Cable Operators
Hotspots
Consumer Devices
Getting Started
Staking
Need Help?
2024 Wifi Dabba Inc. All rights reserved
Your current stakes
Staked (DBT)
Rewards (DBT)
LCO
Blockchain Network
Unlocks on
1
0
wifidabba
Solana
30/12/2024
Compound rewards
Early Unstake
1
0
wifidabba
Solana
08/04/2025
Compound rewards
Early Unstake
Stake your tokens
Tokens to stake
1
Time period in days
30
APY offered by LCOs
Join Discord
Join Now
Learn about the Da Network
Learn Mor


Dashboard
Target
Proxy
Intruder
Repeater
Collaborator
Sequencer
Decoder
Comparer
Logger
Organizer
Extensions
Learn
Settings
payment exploit
Send
Cancel
Target: https://test-b2b-api.wifidabba.com
HTTP/1
Request
Pretty
Raw
Hex
POST /api/explorer/my-hotspots/payments/lco-maintenance-fee-topup HTTP/1.1
Host: test-b2b-api.wifidabba.com
Content-Length: 289
Sec-Ch-Ua-Platform: "macOS"
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzZjMDJmOTM5MmVhNDUwZDQ3NTc4MmUiLCJyb2xlcyI6WyJDVVNUT01FUiJdLCJpYXQiOjE3MzUzNjk1NzIsImV4cCI6MTczNzk2MTU3Mn0.LxbUZRS0KTja-3p3Bg8wLIkXOjX0nmstOab3z7Z5Q04
Accept-Language: en-GB,en;q=0.9
Sec-Ch-Ua: "Chromium";v="131", "Not_A Brand";v="24"
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140 Safari/537.36
Accept: */*
Origin: https://new-explorer-dabba-com.vercel.app
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://new-explorer-dabba-com.vercel.app/
Accept-Encoding: gzip, deflate, br
Priority: u=4, i
Connection: keep-alive
{
"signature":"5t1VxDDQ73qdYfLph9xpSaAnbvKYbChw4veQ2VpYd1JMzKDrn2KZHruRaChQuYXo5bPitwC6DSK6cNif8xr7g1jq",
"tokenAmount":"10000000000",
"lcoCode":"jadhav_broadband",
"numberOfEpochs":10,
"hotspotCount":1e5,
"blockchainNetwork":"Solana",
"existingMaintenanceFeeValidity":"2025-01-27T00:00:00.000Z"
}
Response
Pretty
Raw
Hex
Render
HTTP/1.1 200 OK
Server: nginx
Date: Sat, 28 Dec 2024 08:15:14 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Vary: Accept-Encoding
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Authorization,Roles
Content-Security-Policy: default-src 'self';base-uri 'self';font-src 'self' https: data:;form-action 'self';frame-ancestors 'self';img-src 'self' data:;object-src 'none';script-src 'self';script-src-attr 'none';style-src 'self' https: 'unsafe-inline';upgrade-insecure-requests
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Resource-Policy: same-origin
Origin-Agent-Cluster: ?1
Referrer-Policy: no-referrer
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-DNS-Prefetch-Control: off
X-Download-Options: noopen
X-Frame-Options: SAMEORIGIN
X-Permitted-Cross-Domain-Policies: none
X-XSS-Protection: 0
ETag: W/"5d-UW1oy8CayfO04T0Ef5/QlGMB0Tg"
Content-Length: 93
{
"status":"success",
"code":200,
"message":"LCO fee Payment registered successfully",
"data":{
}
}
Inspector
Request attributes
Request query parameters
Request cookies
Request headers
Response headers
Search
0 highlights
Done
1,067 bytes | 199 millis
Event log (48)
All issues
Memory: 1.48GB

4. Unsanitized user input in dynamic HTML insertion (XSS) Resolved

Description

Unsanitized user input in dynamic HTML insertion can lead to Cross-Site Scripting (XSS) attacks. This vulnerability arises when user-provided data is directly inserted into the DOM without proper sanitization, potentially allowing attackers to execute malicious scripts.

The application directly parses the value of html parameter into the setHTML. Any insecure input value in the html parameter from the user will be directly executed by the setHTML function leading to XSS attack.

Vulnerable URL

quillaudits.new-explorer.wifidabba.com-main/src/components/common/Map.jsx:176

Impact

Do use an HTML sanitization library to clean user input before inserting it into the HTML. This step helps prevent XSS attacks by removing or neutralizing any potentially harmful scripts.

import sanitizeHtml from 'sanitize-html';
const html = `<strong>${user.Input}</strong>`;
document.body.innerHTML = sanitizeHtml(html);

POC

```
map.on("click", "hotspot-layer", (e) => {
  const coordinates = e.features[0].geometry.coordinates[0][0].slice();
  const properties = e.features[0].properties;
  const html = userId ? `<div style="padding-bottom:0px" class="pb-0 min-w-[200px] flex flex-col gap-2 justify-between items-center">
        <span class="text-xs font-bold">${properties.name}</span>
        <hr class="w-full"/>
        <span class="text-sm">${properties.bandwidthConsumedInGb?.toFixed(2)} GB</span>
        <hr class="w-full"/>
      </div>` :
      `<div style="padding-bottom:0px" class="pb-0 min-w-[200px] flex flex-col gap-2 justify-between items-center">
        <span class="text-xs font-bold">${properties.name}</span>
        <hr class="w-full"/>
        <span class="text-sm">LCO: ${properties.lco}</span>
        <hr class="w-full"/>
        <span class="text-sm">Total Hotspots: ${properties.totalHotspots}</span>
        <hr class="w-full"/>
        <span class="text-sm">Bandwidth consumed: </span>
        <span class="text-sm">${properties.bandwidthConsumedInGb?.toFixed(2)} GB</span>
        <a href="/map/${properties.id}" class="text-white text-center text-xs w-full py-2 px-4 rounded bg-[#7284FF]">Visit Location</a>
      </div>
      `

  const newPopup = new mapboxgl.Popup({ closeButton: false })
    .setLngLat(coordinates)
    .setHTML(html)
    .addTo(map);

  setPopup(newPopup);
});
```

5. Unsanitized dynamic input in regular expression

Resolved

Description

Creating regular expressions from dynamic input can lead to a vulnerability known as Regular Expression Denial of Service (ReDoS). This issue arises because some regular expressions can be processed with exponential time complexity. When attackers exploit this, it can significantly drain CPU resources, effectively causing a denial of service.

Vulnerable Endpoint

quillaudits-wifidabba-api-develop

/src/services/coupon.service.js:6
/src/services/customer.service.js:28
/src/services/customer.service.js:37

Recommendation

Do validate all dynamic and user-supplied input against a strict safelist of allowed characters before using it in regular expressions. This step helps prevent attackers from injecting malicious patterns.
Do restrict the length of input that can be processed. Limiting input size is a straightforward way to mitigate many ReDoS vulnerabilities.
Do implement timeouts for regular expression evaluation to avoid excessive resource consumption. This can be achieved using JavaScript environments or libraries that allow setting execution time limits.
Do simplify complex regular expressions to reduce the risk of catastrophic backtracking. Breaking down expressions into simpler parts makes them safer and more manageable.
Do not directly concatenate user input into regular expressions. This practice can introduce unsafe patterns and lead to vulnerabilities.

```
var dynamicRegex = new RegExp('^' + userInput); // unsafe
```

POC

src > services > JS coupon.service.js > findByCoupon > coupon

```
1   const { couponRepository } = require("../repository");
2
3   async function findByCoupon(couponCode) {
4       const currentDate = new Date().toISOString();
5
6       couponCode = new RegExp(`^${couponCode}$`, "i");
7
8       const coupon = await couponRepository.findOne({
9           code: couponCode,
10          start_date: { $lte: currentDate },
11          end_date: { $gte: currentDate },
12          deleted_at: null,
13      });
14
15      if (!coupon) {
16          throw new Error("COUPON_NOT_FOUND");
17      }
18
19      return coupon;
20  }
```

src > services > JS customer.service.js > ...

```
10  async function isValidUserToken(customerId, token) {
20      if (!customer) {
21          throw new Error("CUSTOMER_NOT_FOUND_EXCEPTION");
22      }
23
24      return customer;
25  }
26
27  async function findByEmailWithoutError(email) {
28      email = new RegExp(`^${email}$`, "i");
29
30      return await customerRepository.findOne({
31          email: email,
32          deleted_at: null,
33      });
34  }
35
36  async function findByEmail(email) {
37      email = new RegExp(`^${email}$`, "i");
38
39      const customer = await customerRepository.findOne({
40          email: email,
41          deleted_at: null,
42      });
43
44      if (!customer) {
45          throw new Error("CUSTOMER_NOT_FOUND");
46      }
```

6. Usage of insecure HTTP connection

Resolved

Description

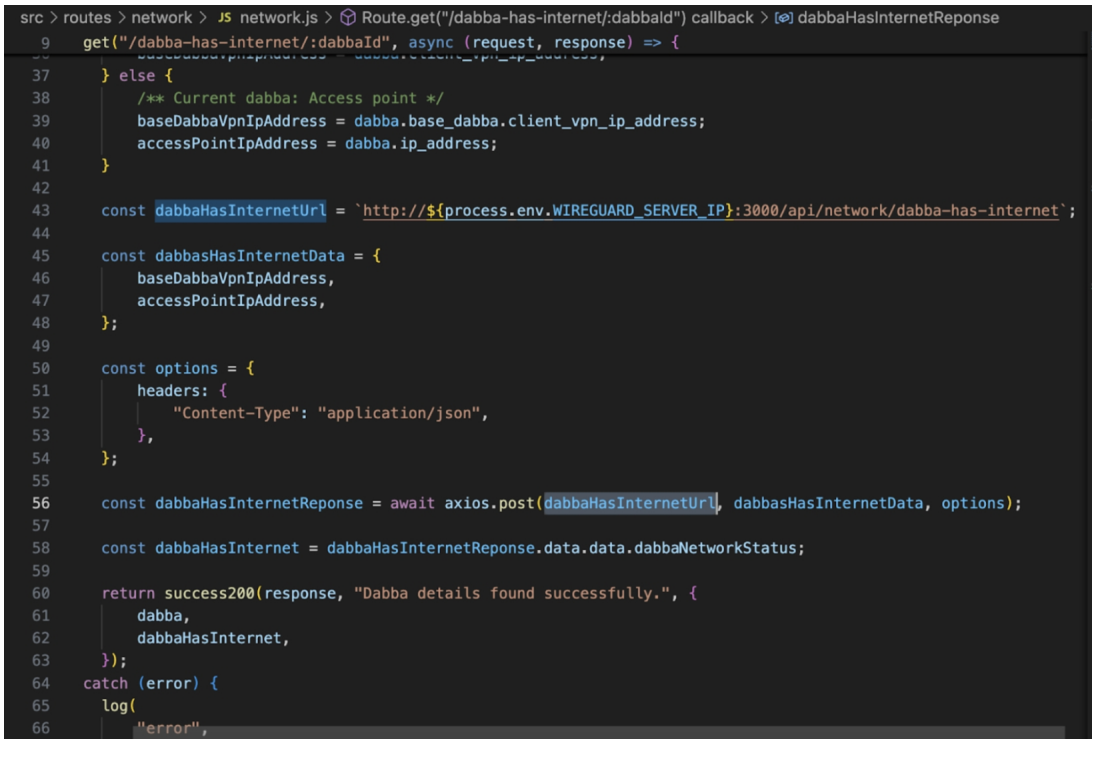
Your application is at risk when it connects to APIs using insecure HTTP connections. This vulnerability occurs because HTTP does not encrypt data, making it susceptible to interception and alteration. Ensure that your application uses HTTPS, which encrypts data in transit, for all connections.

Vulnerable Endpoint

quillaudits-wifidabba-api-develop

/src/routes/network/network.js:56
/src/routes/network/network.js:104

POC



Recommendation

- Do not use HTTP for outgoing connections or API calls. This practice leaves your data vulnerable to eavesdropping and tampering.

  const response = axios.get('http://insecure-api.com') // unsafe

- Do ensure all external connections, especially API calls, use HTTPS to protect data in transit.

  const response = axios.get('***https://secure-api.com***')

7. Unsanitized user input in format string

Resolved

Description

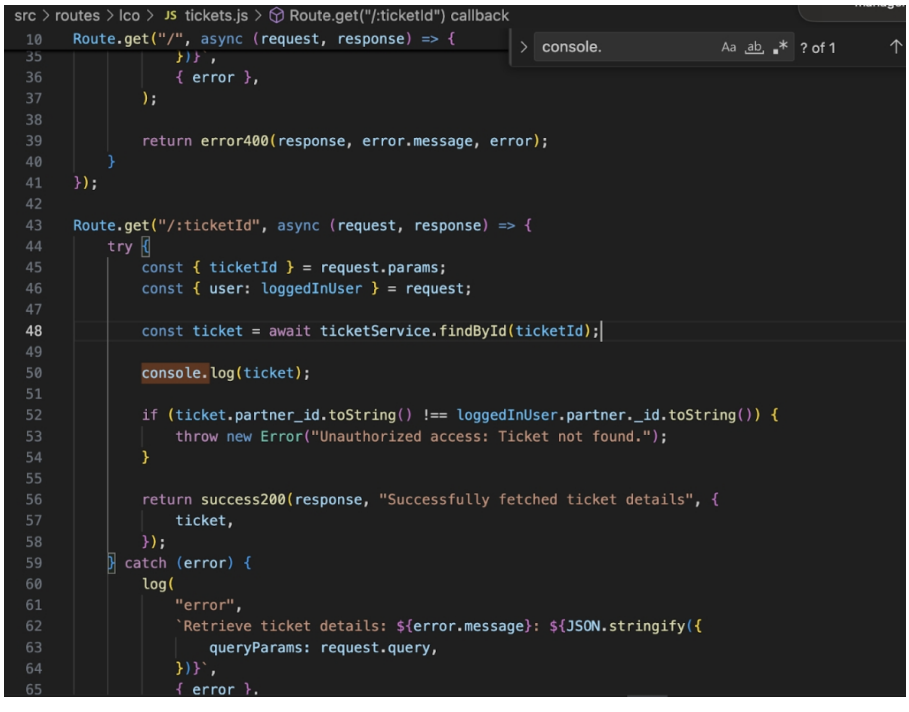
Including user input directly in a format string can lead to security vulnerabilities. This issue arises when an attacker manipulates the format specifiers in the user input, resulting in misleading or fabricated messages.

Vulnerable Endpoint

quillaudits-wifidabba-api-develop

/src/routes/lco/tickets.js:50

POC



Recommendation

- Do not incorporate user input directly into format strings. This approach can be exploited by attackers to manipulate output or execute malicious code.

  console.log(`The value was ${req.params.value}`); // unsafe

- Do use a literal format string and pass user input as additional arguments. This method safely incorporates user input without exposing the application to format string vulnerabilities.

  console.log(“The value was %s”, req.params.value);

8. Information Disclosure

Resolved

Description

The API Endpoint https://test-b2b-api.wifidabba.com/api/explorer/my-hotspots/unclaimed-hotspots?network=Etherum discloses unwanted information about WiFi Hotspots about to be deployed on other networks (in-progress). This information includes - customer_id, order_id, name, currency, when the value of network parameter is changed to any other value except Solana.

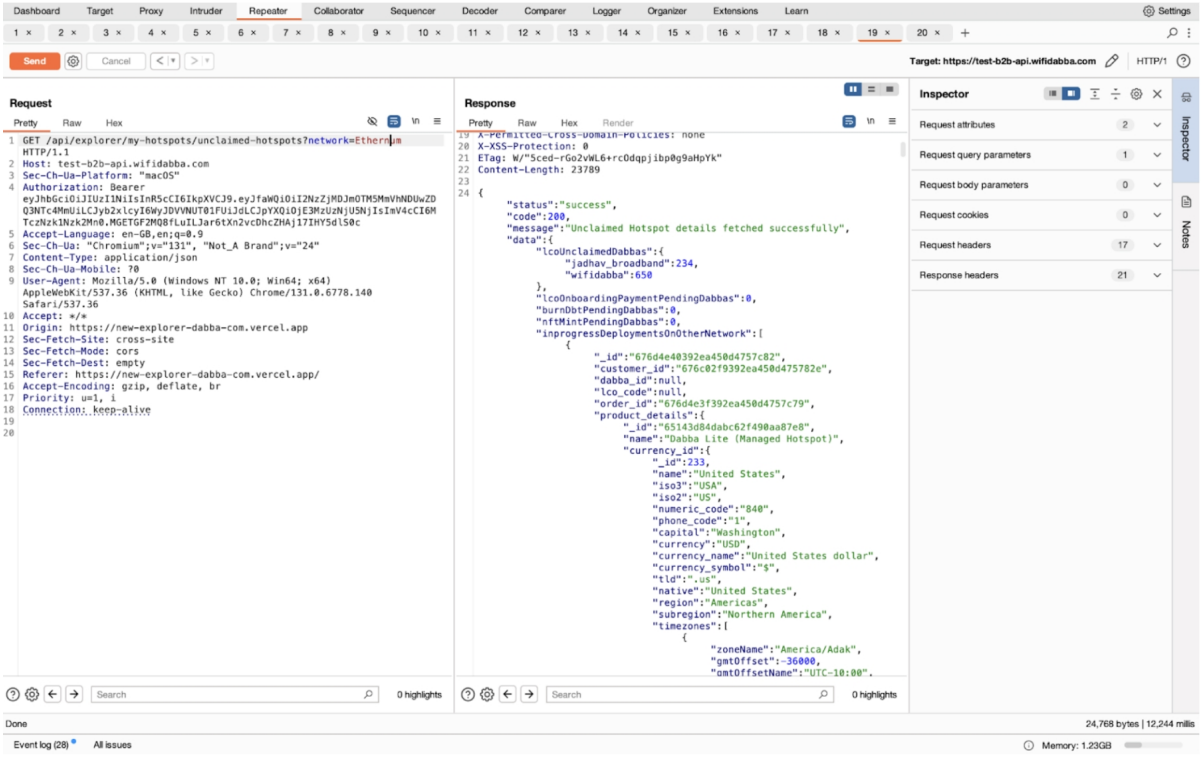
Send a GET request to https://test-b2b-api.wifidabba.com/api/explorer/my-hotspots/unclaimed-hotspots?network=Etherum

Check the response disclosing information about product, customer, currency, and in-progress deployments of WiFi on other networks.

Vulnerable Endpoint

https://test-b2b-api.wifidabba.com/api/explorer/my-hotspots/unclaimed-hotspots?network=Etherum

POC



Recommendation

Limit disclosure of information as much as possible. Implement error handling on the API endpoint to showcase a simple error string when any wrong input is entered.

Low Severity Issues

9. Unsanitized input in NoSQL query

Resolved

Description

Using unsanitized data in NoSQL queries exposes your application to NoSQL injection attacks. This vulnerability arises when user input, request data, or any externally influenced data is directly passed into a NoSQL query function without proper sanitization.

Vulnerable Location

/src/routes/explorer/hotspots.js:429
/src/routes/explorer/maps.js:44
/src/routes/explorer/maps.js:257
/src/routes/lco/bandwidth-plans.js:42
/src/routes/lco/bandwidth-plans.js:128
/src/routes/payments/payments.js:19
/src/services/bandwidth-plan.service.js:6
/src/services/bandwidth-plan.service.js:65
/src/services/customer.service.js:11
/src/services/customer.service.js:65
/src/services/customer.service.js:209
/src/services/dabba-heartbeat.service.js:5
/src/services/dabba-model.service.js:5
/src/services/dabba-provisioning-status.service.js:5
/src/services/dabba.service.js:120
/src/services/dabba.service.js:750
/src/services/partner.service.js:6
/src/services/partner.service.js:20
/src/services/product.service.js:5
/src/services/ticket.service.js:119
/src/services/user.service.js:71
/src/services/user.service.js:255
/src/services/user.service.js:338

Recommendation

Do sanitize all input data before using it in NoSQL queries. Ensuring data is properly sanitized can prevent NoSQL injection attacks.

```
const User = require("../models/user");
username = req.params.username;
User.findOne({ name: username.toString() });
```

10. Clickjacking

Resolved

Description

The application is vulnerable to clickjacking attacks. Clickjacking occurs when an attacker tricks a user into clicking on something different from what they perceive by embedding a legitimate web page within an iframe, often with opacity or style manipulation. This allows attackers to potentially carry out unauthorized actions on behalf of users without their knowledge, such as submitting forms, clicking on buttons, or navigating to malicious sites.

The lack of protection against clickjacking allows attackers to embed the web page inside an iframe, exposing the application to this attack vector.

Vulnerable Location

https://clickjacker.io/test?url=https://new-explorer-dabba-com.vercel.app/

Recommendation

Add the X-Frame-Options HTTP header to instruct the browser to prevent the page from being loaded in an iframe.

Example header values:

X-Frame-Options: DENY: Completely prevents the page from being embedded in an iframe.

X-Frame-Options: SAMEORIGIN: Allows embedding only if the request comes from the same origin as the application.

Use Content Security Policy (CSP) Frame Ancestors Directive:

Implement a Content Security Policy (CSP) that includes the frame-ancestors directive. This provides more granular control over which domains are allowed to embed the application.

Closing Summary

In this report, we have considered the security of the WifiDabba. We performed our audit according to the procedure described above.

Some issues of High, medium, low, and Informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.In the End, Wifi Dabba Network team Resolved all Issues.

Disclaimer

QuillAudits Pentest security audit provides services to help identify and mitigate potential security risks in Wifi Dabba. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Wifi Dabba. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the Wifi Dabba Team to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

6+ Years of Expertise	**1M+** Lines of Code Audited
$30B+ Secured in Digital Assets	**1K+** Projects Secured

Follow Our Journey

AUDIT REPORT

January 2025

For

Dabba

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillhash.com