QuillAudits

Audit Report
July, 2024

For

Table of Content

Table of Content

Overview

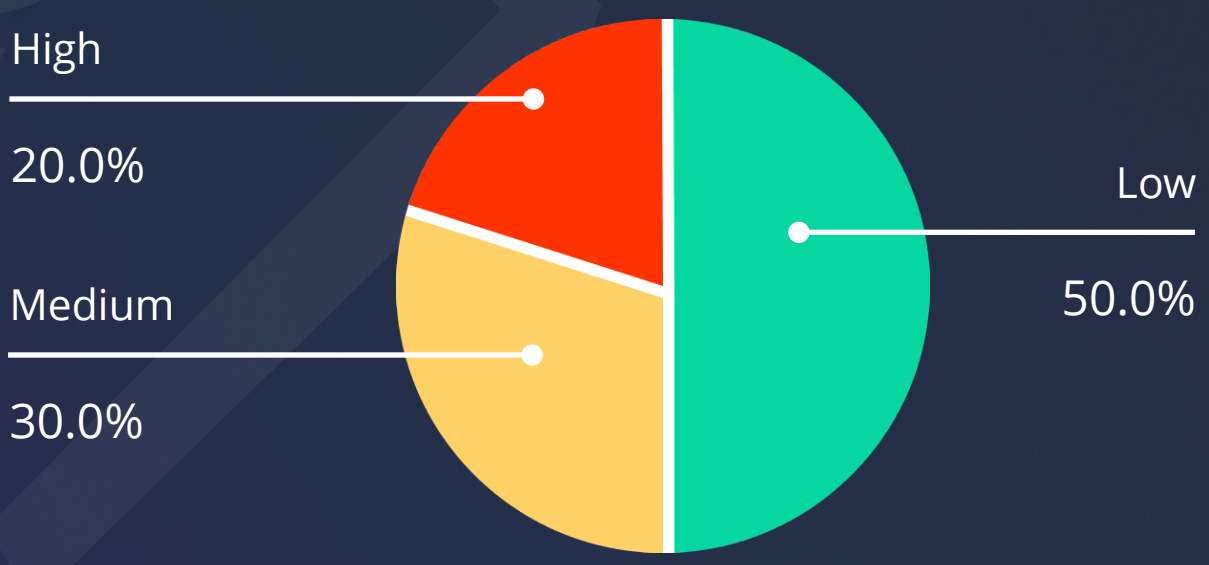
Overview	T Wallet is a cryptocurrency wallet. Below are some of the Features • Secure storage of Bitcoin (BTC) and potentially other cryptocurrencies • User control over private keys • Decentralized operation on a peer-to-peer network • Quick transactions using QR codes, NFC, or Bitcoin URLs
Scope of Audit	The scope of this pentest was to analyse T Wallet(Wave) for quality, security, and correctness.
Timeline	1st July 2024 - 9th July 2024
Updated Code Received	18th July 2024
Final Review	19th July 2024 - 22nd July 2024
In Scope	https://github.com/SOC-Single-Contracts/WAVE_BACKEND
Commit	bc23db2bbec48f1bcf878f81335d3cf11d3663be
Fixed In	NA

Number of Issues per Severity

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	2	3	5	0

Security Issues



Checked Vulnerabilities

We scanned the application for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- 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 Request Forgery
- Broken Authentication and Session Management
- Insufficient Transport Layer Protection
- Broken Access Controls
- Insecure Cryptographic Storage
- Insufficient Cryptography
- Insufficient Session Expiration
- Information Leakage
- Third-Party Components
- Malware
- Denial of Service (DoS) Attacks
- Cross-Site Scripting (XSS)
- Security Misconfiguration
- Unvalidated Redirects and Forwards

And more...

Techniques and Methods

Throughout the pentest of applications, 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
- Client-side and business logic testing

Tools and Platforms used for Pentest:

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

Issue Categories

Every issue in this report has been assigned with 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 web app can be exploited. Issues on this level are critical to the web app'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 web app 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 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 four 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.

Issues Found

High Severity Issues

1. Private Key Leaked in Server Logs for ETH Wallet

Description

A critical security issue has been identified in the /eth-create-wallet endpoint where private keys and mnemonics are being logged in plaintext to the server logs. This can lead to severe security vulnerabilities, as private keys are sensitive information that should be kept confidential and secure. If an attacker gains access to these logs, they could potentially control users' Ethereum wallets and steal their funds

Vulnerable URL

https://github.com/SOC-Single-Contracts/WAVE_BACKEND/blob/bc23db2bbec48f1bcf878f81335d3cf11d3663be/controllers/ethereum.js#L365

Steps to Reproduce

1. s to Reproduce:
2. Deploy the WAVE_BACKEND server.
3. Send a request to the /eth-create-wallet endpoint to trigger wallet creation. Observe the server logs and note that the private key is logged due to the console.log statement on line 365.

POC

```
#!/bin/bash

URL="http://localhost:8080/eth-create-wallet"
REQUESTS=10000

for I in $(seq 1 $REQUESTS); do
    curl -X POST $URL -H "Content-Type: application/json" &
done

wait
```

Impact

Storing private keys in plain text poses a critical security risk. If an attacker gains access to the database, they could:

- Steal and misuse the private keys to authorize transactions.
- Compromise user funds and perform unauthorized operations.
- Damage the reputation of the service and erode user trust.

Recommendation

Remove the console.log statement on line 365 of the controllers/ethereum.js file to ensure that private keys are not logged. Implement secure logging practices to avoid leaking sensitive information in the future.

Status

Resolved

2. Hardcoded Secrets, Api Keys, Private keys and more

Description

The T Wallet project contains several instances of hardcoded credentials, such as private keys, API keys, and API secrets, within the source code. Hardcoding sensitive information in source code poses significant security risks, including unauthorized access and potential data breaches.

Vulnerable URL

1. .env
2. controllers/btc.js
3. tronbridge.js#L20
   controllers/binance.js
   And More

Steps to Reproduce

1. Navigate to the respective URLs provided above.
2. Review the code and configuration files for the presence of hardcoded credentials.
3. Identify the hardcoded credentials, such as private keys, API keys, or API secrets.

Impact

Hardcoded credentials in source code can lead to several security issues, including:
Unauthorized access to sensitive systems and data.
Compromise of user accounts and associated data.
Increased risk of data breaches and exploitation by malicious actors.
Loss of trust and reputational damage for the organization.

Recommendation

Remove Hardcoded Credentials: Replace hardcoded credentials with secure references to environment variables or secret management services.
Use Environment Variables: Store sensitive credentials in environment variables, which can be securely managed and accessed by the application.

Status

Resolved

Medium Severity Issues

3. Lack of Rate Limiting

Description

Multiple endpoint in the T Wallet project lacks appropriate rate limiting, allowing attackers to create an excessive number of wallets or accounts in a short period. This vulnerability can lead to resource exhaustion, service disruption, and potential abuse of system resources.

Steps to Reproduce

Use the provided exploit code to send 10,000 requests to the /eth-create-wallet endpoint.

Observe the creation of 10,000 wallets or accounts without any rate limiting.

Exploit Code

```
#!/bin/bash

URL="http://localhost:8080/eth-create-wallet"
REQUESTS=10000

for I in $(seq 1 $REQUESTS); do
    curl -X POST $URL -H "Content-Type: application/json" &
done

wait
```

Impact

Resource Exhaustion: Creating a large number of wallets or accounts in a short time can lead to database resource exhaustion, slowing down or crashing the service.
Denial of Service (DoS): Uncontrolled requests can overwhelm the server, leading to denial of service for legitimate users.
Abuse of System Resources: Malicious actors can exploit this vulnerability to create multiple accounts for spam or fraudulent activities.

Recommendation

Implement Rate Limiting: Use middleware to limit the number of requests a user can make to the /eth-create-wallet endpoint within a given time period.

Monitor and Alert: Set up monitoring and alerting to detect unusual spikes in account creation activity.

- Use logging and monitoring tools to track the number of requests to sensitive endpoints.
- Set up alerts to notify administrators of potential abuse.

Status

Resolved

4. Lack of Authentication in API Endpoints

Description

The API endpoints in the WAVE_BACKEND project lack proper authentication mechanisms, allowing anyone to access the API without providing valid authentication tokens. This vulnerability can lead to unauthorized access, data breaches, and potential abuse of system functionalities.

Steps to Reproduce

1. Send a request to any API endpoint without including an authentication token.
2. Observe that the request is successfully processed without any authentication checks.

Impact

Unauthorized Access: Malicious users can access API endpoints without authentication, leading to potential data theft and manipulation.
Abuse of System Functionalities: Attackers can exploit API functionalities to perform actions such as creating wallets, transferring funds, or manipulating system resources without authorization.

Recommendation

Implement Authentication Middleware: Use authentication middleware to validate tokens before processing API requests.

Use Strong Authentication Methods: Implement strong authentication methods such as OAuth2.0, JWT, or API keys to secure access to the API.

Rate Limiting and Monitoring: Implement rate limiting and monitoring to detect and prevent excessive or suspicious API requests.

Status

Resolved

5. Id param issue in get_Notification_By_Id

Description

In API endpoint of get_Notification_By_Id it asks for a parameter (id) and that is a hash value. But while calling if you add multiple id parameters it would react to the last one whereas normally it should react to the first one only.

Steps to Reproduce

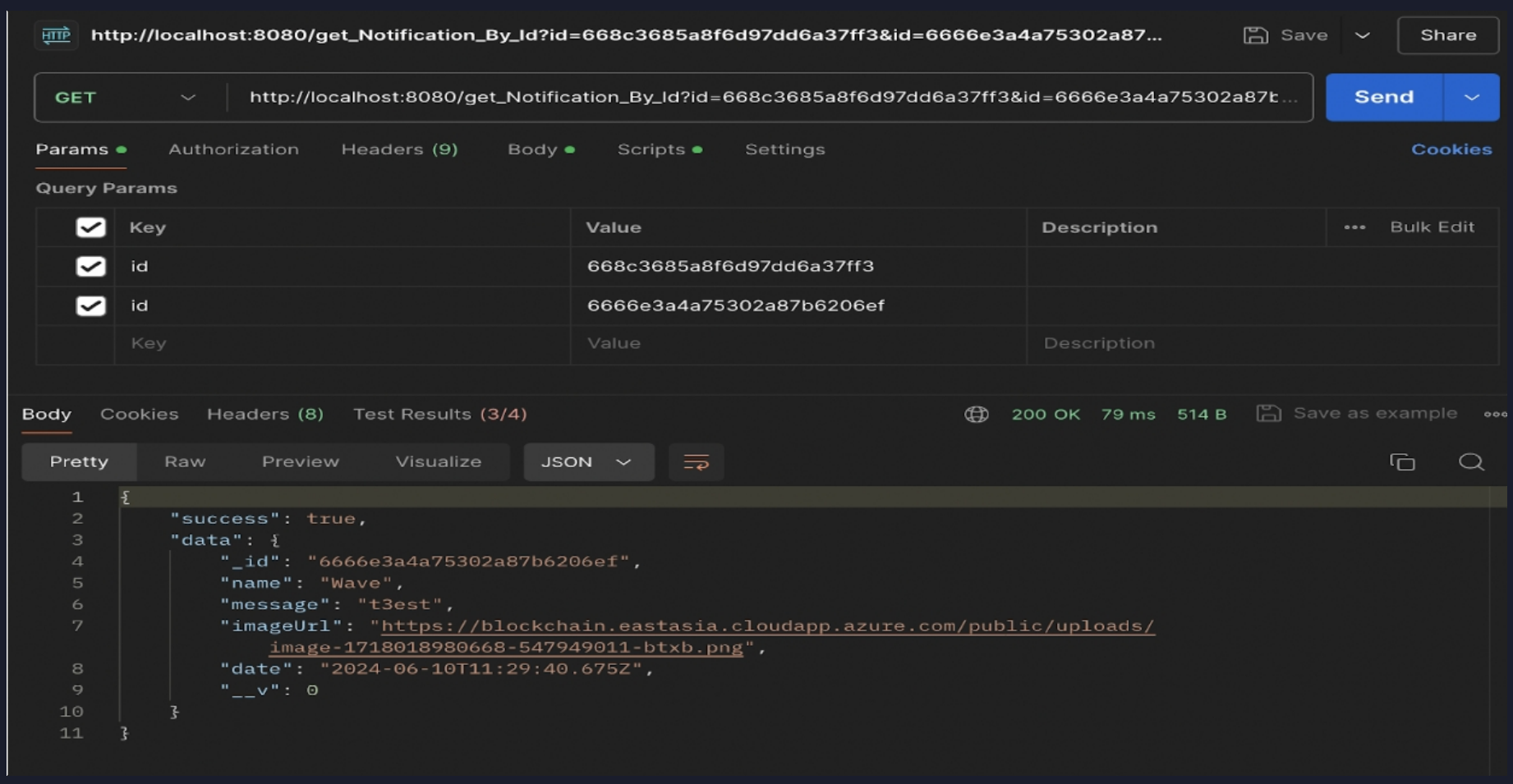
Send a request to below API endpoint without including an authentication token.

http://localhost:8080/get_Notification_By_Id?id=668c3685a8f6d97dd6a37ff3&id=6666e3a4a75302a87b6206ef

Recommendation

Here it should take the first param value and show me response and should ignore the second but here the first one is being ignored.

POC



Status

Resolved

Low Severity Issues

6. uniswap_tokens Endpoint takes too long to load

Description

In API endpoint of uniswap_tokens it takes too long to fetch data and give an output. If this endpoint is cause multiple times concurrently it can cause a lot of stress on server and also result in Denial of Service (DOS attack).

Steps to Reproduce

Send a request to below API endpoint without including an authentication token.

http://localhost:8080/uniswap_tokens

Status

Resolved

7. Unknown error in btc-importaccount

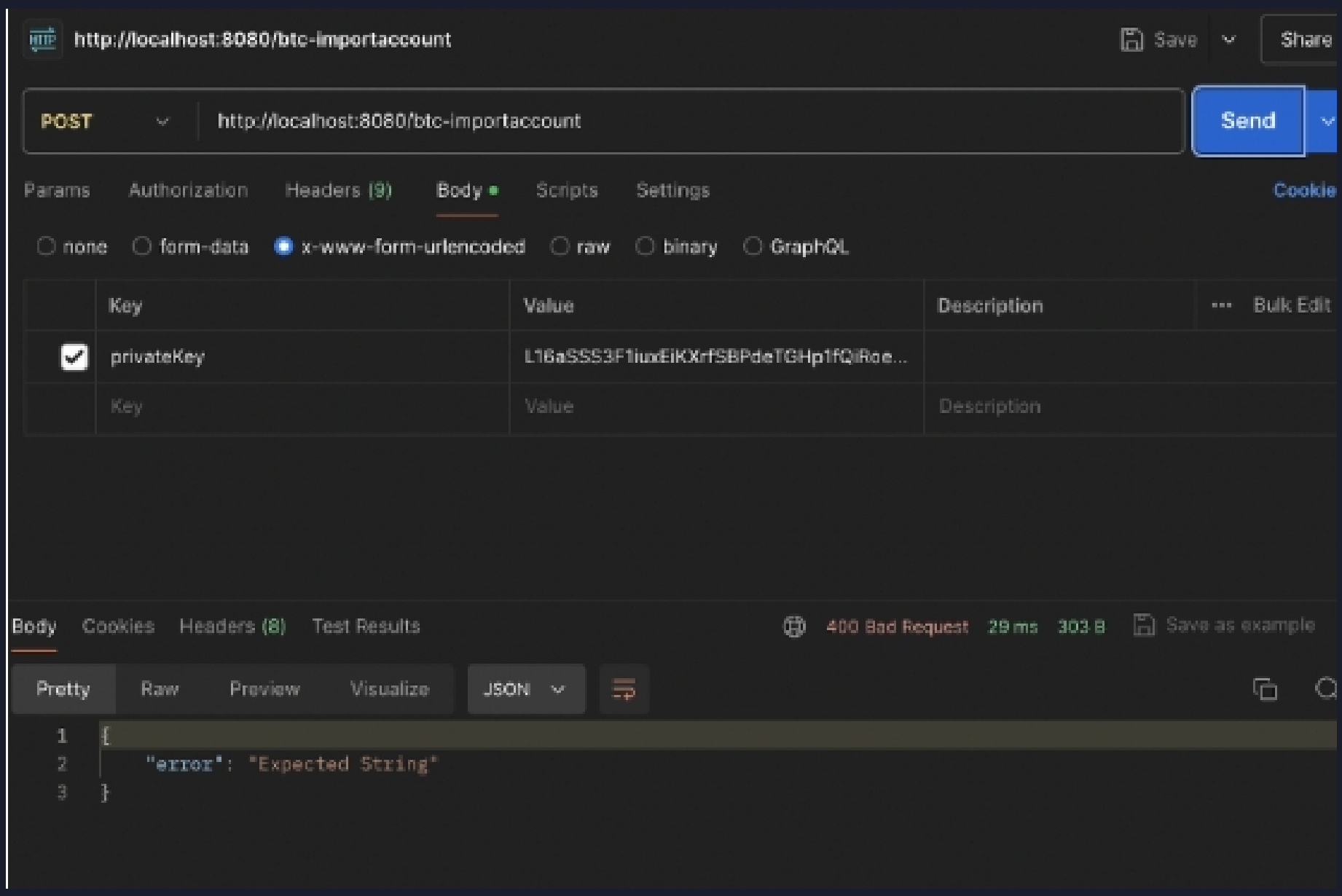
Description

In API endpoint of btc-importaccount it request a request body with content privatekey and the outcome can be 200 , 400 or 500 status code. But if you import key from other wallet it shows a weird response that is not documented for

Steps to Reproduce

Send a request to below API endpoint without including an authentication token.
http://localhost:8080/btc-importaccount

POC



Status

Resolved

8. Cleartext Transmission of Sensitive Information

Description

The T Wallet project transmits sensitive information over cleartext HTTP instead of secure HTTPS. This practice exposes sensitive data, including authentication tokens, personal user information, and transaction details, to potential interception by malicious actors during transmission.

Vulnerable URL

WAVE_BACKEND/app.js Line 248

Steps to Reproduce

Access the application via an HTTP URL (e.g., http://example.com).

Monitor the network traffic using tools like Wireshark or Burp Suite.

Observe that sensitive information is being transmitted in cleartext.

Recommendation

Change the code on line 248 from const server = http.createServer(app); to const server = https.createServer(app);

Status

Resolved

9. Sensitive Cookie in HTTPS Session Without ‘Secure’ Attribute

Description

The T Wallet project sets sensitive cookies in an HTTPS session without using the Secure attribute. This practice makes the cookies vulnerable to being sent over insecure channels if an attacker tricks the user into accessing the HTTP version of the site. The Secure attribute ensures that cookies are only sent over HTTPS, providing an additional layer of security for sensitive information.

Vulnerable URL

WAVE_BACKEND/controllers/user.js Line 111

Impact

Cookie Theft: Without the Secure attribute, cookies can be transmitted over an unencrypted HTTP connection, potentially exposing them to interception by attackers.
Data Integrity and Confidentiality: Lack of secure cookie transmission can compromise the integrity and confidentiality of user data.

Recommendation

Set the Secure Attribute: Ensure that the Secure attribute is set for all cookies containing sensitive information. This ensures that the cookies are only transmitted over HTTPS connections.
Add secure: true in res.cookie on line 112

Status

Resolved

10. Solana account import with mnemonic

Description

While trying to add a phantom account on the backend with mnemonic it shows a complete different public and private key then the account it was provided for.

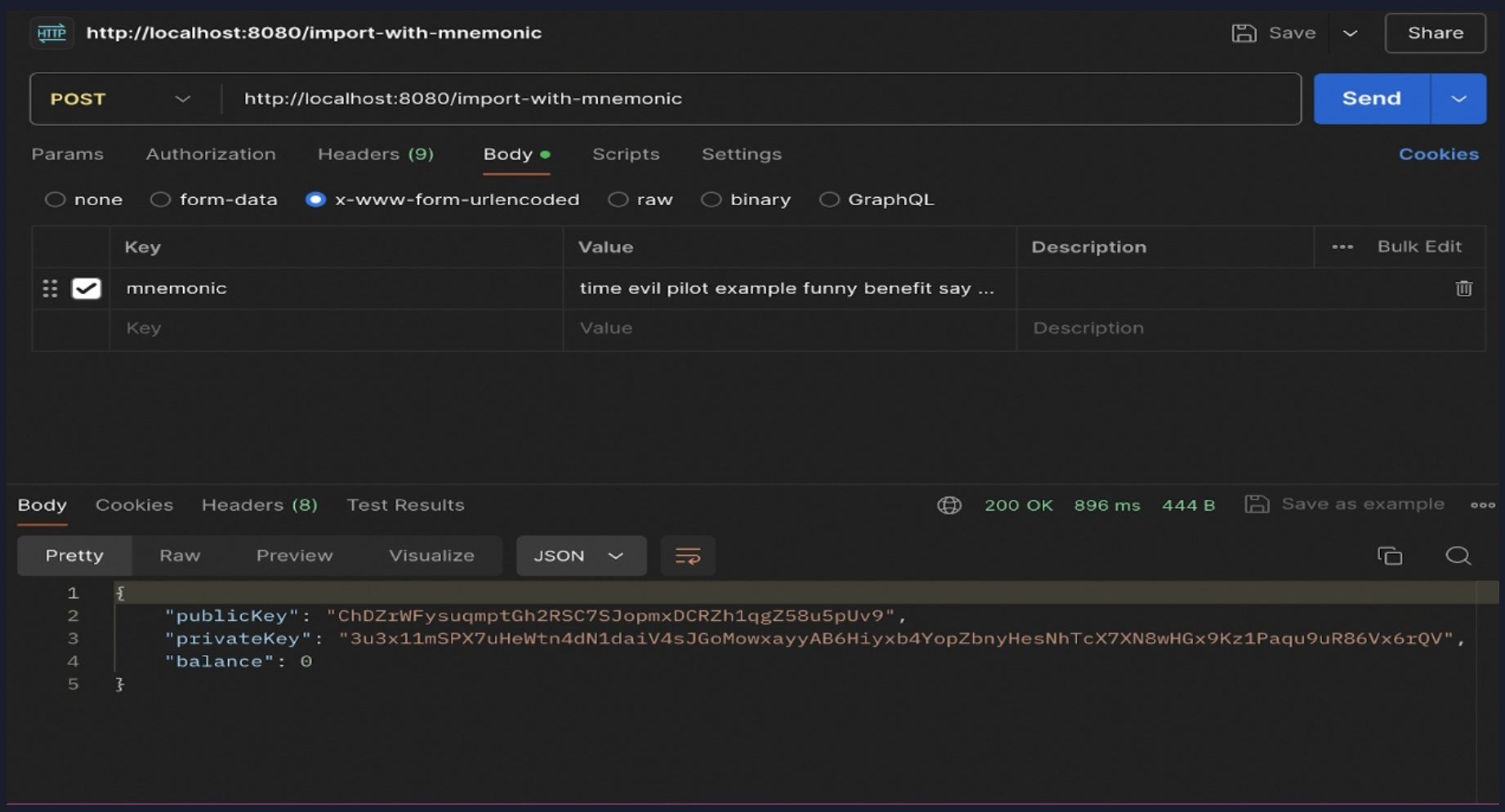
Vulnerable Code

https://github.com/SOC-Single-Contracts/WAVE_BACKEND/blob/main/controllers/wallet.js#L155

Steps to Reproduce

Send a Post Request to http://localhost:8080/import-with-mnemonic with body content of mnemonics .
The response contains a public and private key of a different account than it was intended for .

POC



Status

Resolved

Closing Summary

In this report, we have considered the security of T Wallet. 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 he End, T Wallet Team Resolved all Issues.

Disclaimer

QuillAudits Dapp security audit provides services to help identify and mitigate potential security risks in T Wallet Platform. 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 T Wallet Platform. 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 Platform 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 T Wallet 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.

1000+
Audits Completed

$30B
Secured

1M+
Lines of Code Audited

Follow Our Journey

Audit Report
July, 2024

For

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com