



Leap Wallet

Wallet Pentest

Prepared by: Halborn

Date of Engagement: January 10th, 2022 - February 10th, 2022

Visit: [Halborn.com](https://halborn.com)

DOCUMENT REVISION HISTORY	3
CONTACTS	3
1 EXECUTIVE OVERVIEW	4
1.1 INTRODUCTION	5
1.2 AUDIT SUMMARY	5
1.3 TEST APPROACH & METHODOLOGY	5
RISK METHODOLOGY	6
1.4 SCOPE	8
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	9
3 FINDINGS & TECH DETAILS	10
3.1 (HAL-01) USER IP DISCLOSURE - HIGH	12
Description	12
Screenshots/Videos	12
Risk Level	12
Recommendation	12
Remediation Plan	12
3.2 (HAL-02) AUTO-LOCK AND LOCKING BYPASS - HIGH	13
Description	13
Screenshots/Videos	13
Risk Level	13
Recommendation	13
Remediation Plan	13
3.3 (HAL-03) USING PACKAGES WITH KNOWN VULNERABILITIES - LOW	14
Description	14

Vulnerabilities List	14
Risk Level	14
Recommendation	14
Remediation Plan	14
3.4 (HAL-04) DEPENDENCIES SHOULD BE PINNED TO EXACT VERSIONS - LOW	16
Description	16
Code Location	16
Risk Level	16
Recommendation	17
Remediation Plan	17
3.5 (HAL-05) AUTOMATICALLY GRANTED ACCESS TO DAPP - INFORMATIONAL	18
Description	18
Risk Level	18
Recommendation	18
Remediation Plan	18

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	02/08/2022	Oussama Amri
0.2	Draft Review	02/11/2022	Gabi Urrutia
1.0	Remediation Plan	02/18/2022	Afaq Abid
1.1	Remediation Plan Review	02/24/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Oussama Amri	Halborn	Oussama.Amri@halborn.com



EXECUTIVE OVERVIEW



1.1 INTRODUCTION

Leap Wallet allows users to store, send, swap and stake tokens on the Terra blockchain, sign messages and transactions, and securely manage and store their private keys and assets.

Leap Wallet engaged Halborn to conduct a security assessment on their browser wallet, beginning on January 10th, 2022 and ending on February 10th, 2022 . The security assessment was scoped to Leap Browser Extension. To begin the test, the Client's team provided the code source of the Browser Wallet Extension for Halborn to conduct security testing using tools to scan, detect, validate possible vulnerabilities found in the wallet and report the findings at the end of the engagement.

1.2 AUDIT SUMMARY

The team at Halborn was provided four weeks for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users.

In summary, Halborn identified some security risks that were mostly addressed by the [Leap Wallet team](#).

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of the pentest. While manual testing is recommended to uncover flaws in logic, process and implementation; automated testing techniques

assist enhance coverage of the infrastructure and can quickly identify flaws in it.

The following phases and associated tools were used throughout the term of the audit:

- Storing private keys, mnemonic, seed, and assets securely
- Exposure of any critical information during user interactions with the blockchain and external libraries
- Any attack that impacts funds, such as draining or manipulating of funds
- Application Logic Flaws
- Areas where insufficient validation allows for hostile input
- Exploitation of the webview to gain control of the wallet
- Application of cryptography to protect secrets
- Brute Force Attempts
- Input Handling
- Fuzzing of all input parameters
- Test for Injection (XSS/JSON/HTML)
- Web extension misconfiguration
- Technology stack-specific vulnerabilities and Code Audit
- Known vulnerabilities in 3rd party / OSS dependencies.

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.

- 4 - High probability of an incident occurring.
- 3 - Potential of a security incident in the long term.
- 2 - Low probability of an incident occurring.
- 1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

- 5 - May cause devastating and unrecoverable impact or loss.
- 4 - May cause a significant level of impact or loss.
- 3 - May cause a partial impact or loss to many.
- 2 - May cause temporary impact or loss.
- 1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
----------	------	--------	-----	---------------

- 10 - CRITICAL
- 9 - 8 - HIGH
- 7 - 6 - MEDIUM
- 5 - 4 - LOW
- 3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

The security assessment was scoped to:

Leap Wallet:

- `github/leap-wallet`
- Leap Wallet

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	2	0	2	1

LIKELIHOOD

IMPACT

		(HAL-01)		
			(HAL-02)	
	(HAL-03)			
	(HAL-04)			
	(HAL-05)			

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - USER IP DISCLOSURE	High	SOLVED - 02/22/2022
HAL-02 - AUTO-LOCK AND LOCKING BYPASS	High	SOLVED - 02/24/2022
HAL-03 - USING PACKAGES WITH KNOWN VULNERABILITIES	Low	RISK ACCEPTED
HAL-04 - DEPENDENCIES SHOULD BE PINNED TO EXACT VERSIONS	Low	SOLVED - 03/07/2022
HAL-05 - AUTOMATICALLY GRANTED ACCESS TO DAPP	Informational	ACKNOWLEDGED



FINDINGS & TECH DETAILS



3.1 (HAL-01) USER IP DISCLOSURE - HIGH

Description:

Leap Wallet allows the user to automatically fetch/show NFT images associated with the user's wallet. A highly motivated actor could craft many NFTs, send them to a victim and obtain their IP address, thereby compromising their privacy.

If malicious actors get additional information from the IP address (think geolocation, GSM operator, etc.), they can turn it into a physical risk, such as [kidnapping](#).

Screenshots/Videos:

Loom Video: [Leap Wallet - Privacy Vulnerability](#)

Risk Level:

Likelihood - 3

Impact - 5

Recommendation:

One of the solutions would be for the wallet app to require an explicit confirmation from the user to access the domain **abc.xyz** when fetching the remote image of the NFT, informing the user that this may result in an IP address leak. Another option would be to fetch the remote image in the backend or any kind of network middleware and not in the wallet app.

Remediation Plan:

SOLVED: The [Leap Wallet team](#) fixed the issue by adding the appropriate checks.

3.2 (HAL-02) AUTO-LOCK AND LOCKING BYPASS - HIGH

Description:

Leap wallet suffers from a broken locking mechanism, an attack is usually launched by taking advantage of a poorly managed locking mechanism to masquerade authenticated users.

Screenshots/Videos:

Loom Video: [Leap Wallet - Auto-Lock and Locking Bypass](#)

Risk Level:

Likelihood - 4

Impact - 4

Recommendation:

Developers should verify/validate the user's session in each wallet component. Another option would be to use a single-page application (SPA).

Remediation Plan:

SOLVED: The [Leap Wallet team](#) fixed the issue by adding the appropriate checks.

3.3 (HAL-03) USING PACKAGES WITH KNOWN VULNERABILITIES - LOW

Description:

The Leap Wallet uses third-party dependencies to delegate handling of different types of operations, for example, generation of document in a specific format, HTTP communications, parsing of data of a specific format, etc. However, the dependency presents an expected downside where the actual application's security posture now rests on it.

Vulnerabilities List:

Title	Package	Severity
Prototype Pollution	immer	Critical
Regular expression denial of service	glob-parent	High
Uncontrolled Resource Consumption	ansi-html	High

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

It is strongly recommended to perform an automated analysis of the dependencies from the birth of the project and if they contain any security problems. Developers should be aware of this and apply the necessary mitigation measures to protect the affected application.

Remediation Plan:

ACKNOWLEDGED: The Leap Wallet team updated `glob-parent` and `ansi-html` to the latest version, but the `immer` package was not updated. The Leap Wallet team acknowledged that the `immer` package is not used directly in

the extension and this is a development dependency.

3.4 (HAL-04) DEPENDENCIES SHOULD BE PINNED TO EXACT VERSIONS – LOW

Description:

The Leap Wallet contains over 20 dependencies, some of which are not locked to an exact version, but rather set to a compatible version (^x.x.x). This can potentially allow dependency attacks, as seen with the flow of events package with Copay Bitcoin Wallet.

Code Location:

Listing 1: leap-wallet/package.json

```
15 "dependencies": {
16   "@ledgerhq/hw-transport-webhid": "^6.20.0",
17   "@ledgerhq/hw-transport-webusb": "^6.20.0",
18   "@terra-money/ledger-terra-js": "1.1",
19   "@terra-money/terra.js": "2.1.23",
20   "@terra.kitchen/utils": "^1.0.6",
21   "@types/semver": "^7.3.9",
22   "axios": "0.24.0",
23   "bignumber.js": "9.0.1",
24   "classnames": "2.3.1",
25   "crypto-js": "4.1.1",
26   "date-fns": "2.26.0",
27   "extension-port-stream": "^2.0.1",
28   "extensionizer": "1.0.1",
29   "js-base64": "^3.7.2",
30   "keccak256": "1.0.6",
31   "post-message-stream": "^3.0.0",
32   "qrcode.react": "1.0.1",
33   "qs": "6.10.1",
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

Pinning dependencies to an exact version (=x.x.x) can reduce the chance of inadvertently introducing a malicious version of a dependency in the future.

Remediation Plan:

SOLVED: The **Leap Wallet team** pinned dependencies exact versions.

3.5 (HAL-05) AUTOMATICALLY GRANTED ACCESS TO DAPP - INFORMATIONAL

Description:

When a normal user **Connect** their Leap Wallet with a dApp, this connection will be saved even after restarting the browser. The Leap wallet needs to delete connections made by the user once the browser is closed. This is a reactive approach to prevent the offending dApp from being automatically granted access.

Risk Level:

Likelihood - 2

Impact - 1

Recommendation:

We recommended removing all dApp connections once the browser is terminated.

Remediation Plan:

ACKNOWLEDGED: The **Leap Wallet team** acknowledged this as the default behavior of the wallet.



THANK YOU FOR CHOOSING

 **HALBORN**





Leap Wallet – Backend Wallet Pentest

Prepared by: Halborn

Date of Engagement: February 27th, 2022 – March 15th, 2022

Visit: [Halborn.com](https://halborn.com)

DOCUMENT REVISION HISTORY	4
CONTACTS	4
1 EXECUTIVE OVERVIEW	5
1.1 INTRODUCTION	6
1.2 AUDIT SUMMARY	6
1.3 TEST APPROACH & METHODOLOGY	7
RISK METHODOLOGY	7
1.4 SCOPE	9
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	10
3 FINDINGS & TECH DETAILS	11
3.1 (HAL-01) IMPROPER VALIDATION OF INPUT PARAMETERS - HIGH	13
Description	13
Code Location	13
Risk Level	14
Recommendation	14
Reference	14
Remediation Plan	14
3.2 (HAL-02) NO RATE LIMITING ON API's - MEDIUM	16
Description	16
Results	16
Risk Level	18
Recommendation	18
Remediation Plan	19
3.3 (HAL-03) INCORRECT ERROR HANDLING - MEDIUM	20
Description	20

Code Location	20
Risk Level	21
Recommendation	21
Remediation Plan	21
3.4 (HAL-04) MULTIPLE OUTDATED PACKAGE DEPENDENCIES - MEDIUM	23
Description	23
Results	23
Risk Level	23
Recommendations	24
Remediation Plan	24
3.5 (HAL-05) MULTIPLE TLS MISCONFIGURATIONS - LOW	25
Description	25
Analysis	26
Risk Level	26
Recommendation	26
Remediation Plan	27
3.6 (HAL-06) MISSING SECURITY HEADERS - INFORMATIONAL	28
Description	28
Results	29
Risk Level	29
Recommendation	29
Reference	29
Remediation Plan	30
3.7 (HAL-07) PRESENCE OF TO-DO COMMENTS ON THE CODE - INFORMATIONAL	31
Description	31

	Code Location	31
	Risk Level	31
	Recommendation	31
	Remediation Plan	32
4	ADDITIONAL INFORMATION	33
4.1	Recommended improvements on the code	34
5	PERFORMED TESTS	34
5.1	Rate Limitation	36
	Description	36
	Goal	36
	Result	36

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	03/14/2022	Erlantz Saenz
0.2	Draft Review	03/15/2022	Gabi Urrutia
1.0	Remediation Plan	03/28/2022	Erlantz Saenz
1.1	Remediation Plan Review	03/28/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Erlantz Saenz	Halborn	Erlantz.Saenz@halborn.com



EXECUTIVE OVERVIEW



1.1 INTRODUCTION

engaged Halborn to conduct a security audit on their Leap Wallet Back-end, beginning on February 27th, 2022 and ending on March 15th, 2022 . The security assessment was scoped to the code and the production environment provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided three weeks for the engagement and assigned a full-time security engineer to audit the security of the Leap Wallet Back-end. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The following phases and associated tools were used throughout the term of the audit:

- Mapping API Content and Functionality
- API Logical Flaws
- Rate Limitations Tests
- API misconfiguration
- Input Handling
- Fuzzing of all input parameters
- Test for Injection (SQL/JSON/HTML/Command)

In summary, Halborn identified some security risks that were mostly addressed by the team.

1.3 TEST APPROACH & METHODOLOGY

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

- 5 - Almost certain an incident will occur.
- 4 - High probability of an incident occurring.
- 3 - Potential of a security incident in the long term.
- 2 - Low probability of an incident occurring.
- 1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

- 5 - May cause devastating and unrecoverable impact or loss.
- 4 - May cause a significant level of impact or loss.
- 3 - May cause a partial impact or loss to many.
- 2 - May cause temporary impact or loss.
- 1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
----------	------	--------	-----	---------------

10 - CRITICAL

9 - 8 - HIGH

7 - 6 - MEDIUM

5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

The security assessment was scoped to the git repo:

- <https://github.com/leapwallet/leap-backend> (commit ID: 920098bad25286bbd94650d6ecaba624ca4e91e8)

Additionally, the verification of the issues were performed against:

- <https://api.leapwallet.io/> (Production environment)

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	3	1	2

LIKELIHOOD

IMPACT

			(HAL-01)	
		(HAL-03) (HAL-04)	(HAL-02)	
(HAL-06)	(HAL-05)			
(HAL-07)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - IMPROPER VALIDATION OF INPUT PARAMETERS	High	SOLVED - 03/25/2022
HAL-02 - NO RATE LIMITING ON APIs	Medium	SOLVED - 03/25/2022
HAL-03 - INCORRECT ERROR HANDLING	Medium	SOLVED - 03/25/2022
HAL-04 - MULTIPLE OUTDATED PACKAGES DEPENDENCIES	Medium	PARTIALLY SOLVED AND RISK ACCEPTED
HAL-05 - MULTIPLE TLS MISCONFIGURATIONS	Low	RISK ACCEPTED
HAL-06 - MISSING SECURITY HEADERS	Informational	SOLVED - 03/25/2022
HAL-07 - PRESENCE OF TO-DO COMMENTS ON THE CODE	Informational	SOLVED - 03/25/2022



FINDINGS & TECH DETAILS



3.1 (HAL-01) IMPROPER VALIDATION OF INPUT PARAMETERS - HIGH

Description:

The code review of the Leap Wallet Back-end revealed some unsanitized input parameters. This could lead to uncontrolled errors on the application (Refer to HAL-03 - Incorrect error handling) or unexpected behaviors. Additionally, an uncontrolled parameter could leverage in an SQL injection, XSS, or other web attacks against the server and the application.

Code Location:

The line 17 of code, only verifies that the txHash and isMainnet exist on the request, but the application does not verify the data contained on the variables.

POST /txs endpoint

Listing 1: src/routes/txs/txs.ts (Line 17)

```
16 router.post('/', async (req, res) => {
17   if (req.body.txHash === undefined || req.body.isMainnet ===
    ↳ undefined) return res.status(400).json('Bad request.');
```

```
18
19   // log hash
20   logger.info(req.body.txHash);
21
22   // validate txhash
23   const isValid = await validateTxHash(req.body.txHash, req.body.
    ↳ isMainnet);
24   if (!isValid) return res.status(406).json('Invalid TxHash.');
```

```
25
26   const data = {
27     txHash: req.body.txHash,
28     createdAt: new Date(),
29     isMainnet: req.body.isMainnet,
30     type: req.body.type ?? null,
31     info: req.body.info ?? {},
```

```
32  };
33
34  try {
35      await Container.get(Transaction.token).create(data);
36  } catch (error) {
37      logger.error(error);
38      // 409 Conflict - This response is sent when a request
39      ↳ conflicts with the current state of the server.
40      return res.status(409).json('TxHash already exists.');
```

Risk Level:

Likelihood - 4

Impact - 4

Recommendation:

Sanitize all the input provided by the user on the server-side, and remove all the characters or code that could be dangerous during the process of the parameters. Sanitizing all the parameters sent by the user, the developers, would reduce the risk of an unexpected behavior.

Reference:

[SQL injection \(SQLi\)](#)

[Cross Site Scripting \(XSS\)](#)

Remediation Plan:

SOLVED: The issue was solved in the commit ID:

[69f342ff5aa100b5e82b67c9f8a39c6ac1281cb0](#)

The **Leap Wallet team** modified the code by adding user input validation.

```
18 + const schema = Joi.object({
19 +   txHash: Joi.string().alphanum().length(64).required().strict(),
20 +   isMainnet: Joi.boolean().required().strict(),
21 +   type: Joi.string()
22 +     .regex(/TRANSFER|WEB_APP|ANCHOR/)
23 +     .required()
24 +     .strict(),
25 +   info: Joi.object().required().strict(),
26 + });
27 +
28 + export enum TxCategory {
29 +   TRANSFER = 'TRANSFER',
30 +   WEB_APP = 'WEB_APP',
31 +   ANCHOR = 'ANCHOR',
32 + }
33 +
34 + export type NewTx = {
35 +   readonly txHash: string;
36 +   readonly isMainnet: boolean;
37 +   readonly type: TxCategory;
38 +   readonly info: object;
39 + };
40 +
41 router.post('/', async (req, res) => {
42 +   const { value, warning, error } = schema.validate(req.body);
43 +   if (error !== undefined || warning !== undefined) {
44 +     res.status(400).json(error);
45 +     return;
```

Figure 1: Added rate-limiting headers.

3.2 (HAL-02) NO RATE LIMITING ON API'S - MEDIUM

Description:

Several attacks to abuse the **lack of rate limiting** were launched through APIs to force the application to fall over by overwhelming network, CPU, memory or storage resources. This often results in a sluggish behavior, system crashes, or other rogue server behaviors, resulting in denial of service (DoS).

Results:

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	835	
1	1	200	<input type="checkbox"/>	<input type="checkbox"/>	844	
2	2	200	<input type="checkbox"/>	<input type="checkbox"/>	841	
3	3	200	<input type="checkbox"/>	<input type="checkbox"/>	839	

Request	Response
	<pre> 1 HTTP/1.1 200 OK 2 Date: Tue, 08 Mar 2022 18:24:22 GMT 3 Content-Type: application/json; charset=utf-8 4 Content-Length: 35 5 Connection: close 6 x-powered-by: Express 7 etag: W/"23-9esg5Ni4mkpYKbgL2BPAz0TF3kQ" 8 x-envoy-upstream-service-time: 4 9 CF-Cache-Status: DYNAMIC 10 Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct" 11 Report-To: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=KeqN73BnVncp7BpWKWz0yJzdsL NWGehlWjVG2P8GfwpS8UZWiEK6S2W3kojNXxNhmCN0hb8sd4OgaScgfkctc4LTQIohldoYnbIDvHNC01%2B9hwEGvQ3alm d98IVpYTay09Y%2BfFg%3D%3D"}],"group":"cf-nel","max_age":604800} 12 NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800} 13 Server: cloudflare 14 CF-RAY: 6e8d9b7bfa36401f-CDG 15 alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400 16 17 { "db":true, "cache":true, "api":true } </pre>

Figure 2: Lack of rate limiting allowed to execute request with no restrictions.

The images above highlighted the time when the response was received after call `/health` API endpoint. In this example, only one API call was used, however this could be applied to the rest of the API functionalities of the app.

Request	Payload	Status	Error	Timeout	Length	Comment
1660	1660					
1659	1659					
1658	1658	200			839	
1657	1657	200			837	
1656	1656	200			841	

RequestResponse

PrettyRawHexRender

1 HTTP/1.1 200 OK
2 Date: Tue, 08 Mar 2022 18:26:08 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 35
5 Connection: close
6 x-powered-by: Express
7 etag: W/"23-9esg5Ni4mkpYKbgL2BPAz0TF3kQ"
8 x-envoy-upstream-service-time: 4
9 CF-Cache-Status: DYNAMIC
10 Expect-CT: max-age=604800,
report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
11 Report-To:
{ "endpoints": [{ "url": "https://a.nel.cloudflare.com/report/v3?s=CE9nVA13VWAikeJZWdI4SIpFR%
2BrRV3MUEEo2GBLgiLHP2azBZozuhpI3j%2FR2Q1SzIcbrBKay8M2Fv0TTroMCdFaksDpu4DzxP1EcBA%2BgF%2FWpHcM
0oWm0mC07dLrp9gIlJv7P6A%3D%3D"}], "group": "cf-nel", "max_age": 604800 }
12 NEL: { "success_fraction": 0, "report_to": "cf-nel", "max_age": 604800 }
13 Server: cloudflare
14 CF-RAY: 6e8d9e12e9e23a05-CDG
15 alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400
16
17 {
 "db": true,
 "cache": true,
 "api": true
}

Figure 3: Lack of rate limiting allowed to execute request with no restrictions.

Furthermore, it is recommended to set up policies that would limit API mass use and include automated lockouts to prevent abuse.

Remediation Plan:

SOLVED: The issue was solved in the commit ID:

[0388872e9486d4a50db78aaff9b7b695eac5c797](#)

The **Leap Wallet team** added the rate-limiting control effectively.

```
HTTP/2 429
date: Fri, 25 Mar 2022 17:08:27 GMT
content-type: text/html; charset=utf-8
x-powered-by: Express
ratelimit-limit: 60
ratelimit-remaining: 0
ratelimit-reset: 31
retry-after: 60
x-envoy-upstream-service-time: 1
cf-cache-status: DYNAMIC
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
report-to: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=drbaZnFcPdTAjuVismsCAeN5Mf1jRfM5oR20ajQwpY%2FFK54ECZM1SCj4BNhVg%3D%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 6f193fa9ec7d94ee-LIS
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400

Too many requests, please try again later.
```

Figure 5: Added rate-limiting headers.

3.3 (HAL-03) INCORRECT ERROR HANDLING – MEDIUM

Description:

The lack of input validation leverage on an improper error handling. This behavior could confuse to the user of the application, giving to the end user wrong information.

Code Location:

On the line 17, it was possible to advise that the parameters were not been validating. A user could introduce malformed data on the `isMainnet` parameter and valid data on the `txHash` parameter. The line 23 would verify that the introduced `txHash` was correct, but the line 35 would trigger an error due to the data contained on the `isMainnet` parameter. The error would be caught, returning the error message `Txhash already exists`. even when the database was not containing the `txHash`.

POST /txs endpoint

Listing 2: `src/routes/txs/txs.ts` (Lines 17,34,35,36,37,39,40)

```
16 router.post('/', async (req, res) => {
17   if (req.body.txHash === undefined || req.body.isMainnet ===
  ↳ undefined) return res.status(400).json('Bad request.');
```

```
18
19   // log hash
20   logger.info(req.body.txHash);
21
22   // validate txhash
23   const isValid = await validateTxHash(req.body.txHash, req.body.
  ↳ isMainnet);
24   if (!isValid) return res.status(406).json('Invalid TxHash.');
```

```
25
26   const data = {
27     txHash: req.body.txHash,
28     createdAt: new Date(),
29     isMainnet: req.body.isMainnet,
```

```

30     type: req.body.type ?? null,
31     info: req.body.info ?? {},
32 };
33
34 try {
35     await Container.get(Transaction.token).create(data);
36 } catch (error) {
37     logger.error(error);
38     // 409 Conflict - This response is sent when a request
39     ↪ conflicts with the current state of the server.
40     return res.status(409).json('TxHash already exists.');
```

Improper error handling leverage to an invalid error message.

Risk Level:

Likelihood - 3

Impact - 3

Recommendation:

Validate the input from the user and emit the errors accordingly to situation.

Remediation Plan:

SOLVED: The issue was solved in the commit ID:

[69f342ff5aa100b5e82b67c9f8a39c6ac1281cb0](#)

The **Leap Wallet team** modified the code by adding user input validation and thus solving incorrect error handling.

```
18 + const schema = Joi.object({
19 +   txHash: Joi.string().alphanum().length(64).required().strict(),
20 +   isMainnet: Joi.boolean().required().strict(),
21 +   type: Joi.string()
22 +     .regex(/TRANSFER|WEB_APP|ANCHOR/)
23 +     .required()
24 +     .strict(),
25 +   info: Joi.object().required().strict(),
26 + });
27 +
28 + export enum TxCategory {
29 +   TRANSFER = 'TRANSFER',
30 +   WEB_APP = 'WEB_APP',
31 +   ANCHOR = 'ANCHOR',
32 + }
33 +
34 + export type NewTx = {
35 +   readonly txHash: string;
36 +   readonly isMainnet: boolean;
37 +   readonly type: TxCategory;
38 +   readonly info: object;
39 + };
40 +
41 router.post('/', async (req, res) => {
42 +   const { value, warning, error } = schema.validate(req.body);
43 +   if (error !== undefined || warning !== undefined) {
44 +     res.status(400).json(error);
45 +     return;
```

Figure 6: Validation of the user input.

3.4 (HAL-04) MULTIPLE OUTDATED PACKAGE DEPENDENCIES – MEDIUM

Description:

During the security assessment, an automated check was performed against the project dependencies. The command `yarn audit` revealed the use of multiple vulnerable or outdated dependencies.

Results:

```
root@d67808bfd418:/home/leap-backend-main/node_modules/urijs#  
grep -iE 'version' package.json  
"version": "1.19.8",
```

Figure 7: Outdated dependency found on the affected project.

high	Open Redirect in urijs
Package	urijs
Patched in	>=1.19.10
Dependency of	@stoplight/spectral-cli
Path	@stoplight/spectral-cli > @stoplight/spectral-ref-resolver > @stoplight/json-ref-resolver > urijs
More info	https://www.npmjs.com/advisories/1054109

Figure 8: Yarn audit command issue evidence.

Package	Vulnerability	Patched version
node-fetch	Exposure of Sensitive Information	>=2.6.1
node-fetch	Lack of Size Validation	>=2.6.1
urijs	Open Redirect	>=1.19.10
urijs	Protocol Validation Bypass	>=1.19.9

Risk Level:

Likelihood - 3

Impact - 3

Recommendations:

Apply the security patches recommended by the project maintainers. Regarding the urijs vulnerabilities could be solved applying the patches automatically with the command `npm audit fix`.

Remediation Plan:

PARTIALLY SOLVED: The `Leap Wallet team` solved multiple outdated dependencies. However, multiple outdated packages were still outdated. The team accepted the risk of this finding.

3.5 (HAL-05) MULTIPLE TLS MISCONFIGURATIONS - LOW

Description:

Several misconfigurations were identified within the application's SSL/TLS configuration that could compromise the security of communications. While attacks that target weak cipher suites or algorithms are complex to execute, with the steady progress of computational power these attacks become easier to achieve over time. As such, it is recommended to configure the connection to comply with security best practices.

Many protocols may be used in establishing a secure connection. In the past, various vulnerabilities have existed surrounding legacy protocols and their associated cipher suites. Newer versions of these secure communication protocols and modern browsers address the security vulnerabilities identified in older versions. For example, the POODLE (Padding Oracle On Downgraded Legacy Encryption) vulnerability allowed information to be extracted from communication headers and the DROWN attack allowed SSLv2 traffic to be decrypted. Neither of these attacks continue to pose a threat in modern browsers. However, if a user accesses the application with an outdated browser, these vulnerabilities may remain exploitable.

Encryption ciphers are used to protect communication channels between a client and a web server and are negotiated when a client initially connects to the server. This negotiation involves agreeing on a cipher suite and a combination of protocols, encryption algorithms, key lengths, and hashing algorithms supported by both parties. To support a wide range of browsers, web servers typically support ciphers of various strengths. However, some encryption algorithms do not provide adequate security, due to issues such as implementation flaws or inadequate key lengths. Additionally, permitting insecure cipher suites puts users with outdated software at greater risk, as an attacker could downgrade a user's connection to an insecure cipher.

The SSL/TLS cipher suites support a variety of key lengths. Only suites with sufficiently long keys should be permitted, as short keys may allow an attacker to perform a brute-force attack to decrypt the traffic.

Analysis:

Listing 3

```

1 Testing protocols via sockets except NPN+ALPN
2
3 SSLv2      not offered (OK)
4 SSLv3      not offered (OK)
5 TLS 1      offered (deprecated)
6 TLS 1.1    offered (deprecated)
7 TLS 1.2    offered (OK)
8 TLS 1.3    offered (OK): final
9 NPN/SPDY   h2, http/1.1 (advertised)
10 ALPN/HTTP2 h2, http/1.1 (offered)
11
12 Testing cipher categories
13
14 NULL ciphers (no encryption)                not offered (OK)
15 Anonymous NULL Ciphers (no authentication)  not offered (OK)
16 Export ciphers (w/o ADH+NULL)                not offered (OK)
17 LOW: 64 Bit + DES, RC[2,4] (w/o export)      not offered (OK)
18 Triple DES Ciphers / IDEA                    not offered (OK)
19 Obsolete: SEED + 128+256 Bit CBC cipher      offered
20 Strong encryption (AEAD ciphers)             offered (OK)
21

```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

As the application functionality is hosted across multiple servers, the SSL/TLS configuration of each should be reviewed and updated to comply

with security best practice and to minimize the risk of existing and future SSL/TLS vulnerabilities.

- Remove support for both TLSv1.0 and TLSv1.1, and use instead TLSv1.3.
- The use of weak cipher suites identified above should be removed.
- Key lengths of 256 bits and above should be used for symmetric encryption algorithms and a length of 4096 bits should be used for RSA algorithms. For asymmetric keys generated with ECC algorithms, the minimum recommended key size is 512 bits.

Remediation Plan:

RISK ACCEPTED: The Leap Wallet team accepted the risk of this finding.

3.6 (HAL-06) MISSING SECURITY HEADERS - INFORMATIONAL

Description:

We have identified that there are important security Headers missing in the Backend APIs. These headers used by client browser and enhance the security for end users against common attacks. Although these are only APIs but in advance scenario, it is still a chance to exploit the vulnerabilities so it is still recommended to follow the security best practices and implement these headers.

Important missing security headers; **Strict-Transport-Security**, **X-Frame-Options**, **X-Content-Type-Options** and **Content-Security-Policy** response headers.

- **Strict-Transport-Security (HSTS)** HTTP Strict Transport Security is an important security header to implement, as it makes the browser to only be communicated over HTTPS. Recommended value "Strict-Transport-Security: max-age=31536000; includeSubDomains".
- **X-Frame-Options** tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site, you can defend against attacks like clickjacking. Recommended value "X-Frame-Options: SAMEORIGIN".
- **X-Content-Type-Options** stops a browser from attempting to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".
- **Content-Security-Policy** is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets.

```
* Connection state changed (MAX_CONCURRENT_STREAMS == 256)!
< HTTP/2 200
< date: Tue, 08 Mar 2022 16:18:23 GMT
< content-type: application/json; charset=utf-8
< content-length: 35
< x-powered-by: Express
< etag: W/"23-9esg5Ni4mkpYKbgL2BPAz0TF3kQ"
< x-envoy-upstream-service-time: 4
< cf-cache-status: DYNAMIC
< expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
< report-to: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=NX%2FJCviW%2F0icVw%2BuTdaoKv0mofqLaiZXNgilNcWgNQw8aFjLc0q0fi8RDu7oP%2Beix1oLd0%2F5ACk6PBuNmUz4cYpXVyS%2FhMBvXN%2BlsYXFfbnUIUqiPAxPDyl08yodYJmnRkKtyQ%3D%3D"}],"group":"cf-nel","max_age":604800}
< nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
< server: cloudflare
< cf-ray: 6e8ce2ebe94299ae-CDG
< alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400
<
* Connection #0 to host api.leapwallet.io left intact
{"db":true,"cache":true,"api":true}* Closing connection 0
```

Figure 9: Lack of security headers.

Results:

Risk Level:

Likelihood - 1

Impact - 2

Recommendation:

It is recommended to define `Strict-Transport-Security`, `X-Content-Type-Options=nosniff`, `Content-Security-Policy` and `X-Frame-Options` response headers with proper policies.

Reference:

`Strict-Transport-Security`
`X-Content-Type-Options`
`X-Frame-Options`
`Content Security Policy`
`Content-Type`

Remediation Plan:

SOLVED: The issue was solved in the commit ID:

[0388872e9486d4a50db78aaff9b7b695eac5c797](https://github.com/leapwallet/leapwallet/commit/0388872e9486d4a50db78aaff9b7b695eac5c797)

The Leap Wallet team added the security headers in the server responses.

```
→ curl -ki https://api.leapwallet.io
HTTP/2 200
date: Fri, 25 Mar 2022 16:05:16 GMT
content-type: text/html; charset=utf-8
x-powered-by: Express
ratelimit-limit: 60
ratelimit-remaining: 59
ratelimit-reset: 41
strict-transport-security: max-age=31536000; includeSubDomains
x-frame-options: SAMEORIGIN
x-content-type-options: nosniff
content-security-policy: default-src 'none'
x-envoy-upstream-service-time: 17
cf-cache-status: DYNAMIC
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/
report-to: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=zu%
0nhi9WBhf2A2dt56SXPIIrtNZykdteHtQWbTtNug%3D%3D"}],"group":"cf-nel","max_age":6048
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 6f18e3197d9d94f8-LIS
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400
```

Figure 10: Added security headers.

3.7 (HAL-07) PRESENCE OF TO-DO COMMENTS ON THE CODE – INFORMATIONAL

Description:

At least one TO-DO comment was found on the code. From the security perspective, the use of these comments does not imply a security risk. However, it could mean that the developed application did not reach an appropriate level of maturity to be on a production environment.

Code Location:

Listing 4: src/routes/txs/validateTxHash.ts (Lines 15,16)

```
14 export default async function validateTxHash(txHash: string,  
  ↳ isMainnet: boolean): Promise<boolean> {  
15   // TODO: fix this with message queue  
16   await sleep(1000);  
17  
18   const baseUrl = isMainnet ? TERRA_API_BASE_URL.MAINNET :  
  ↳ TERRA_API_BASE_URL.TESTNET;
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:


Review all the comments on the code and ensure that this situation does not affect or offer any risk to the security of the application.

Remediation Plan:

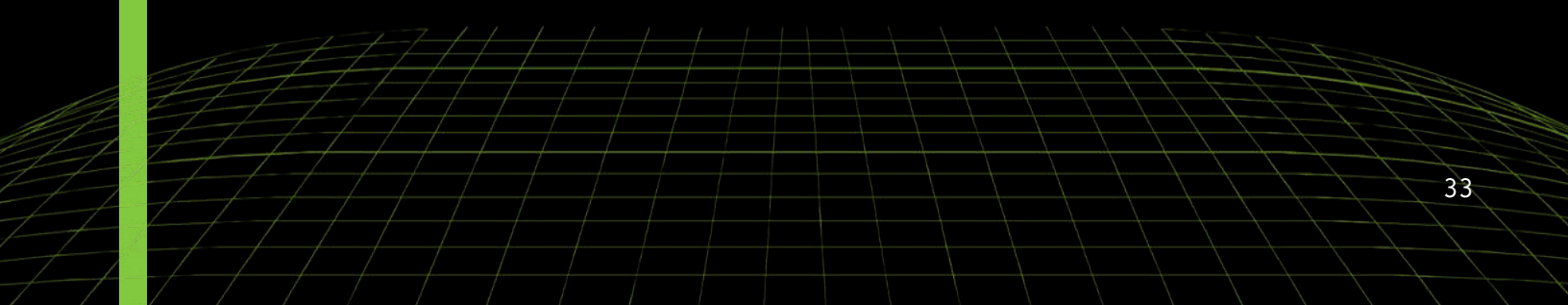

SOLVED: The issue was solved in the commit ID:

[2049821249ceeb7dbec2222b515a09b30752a74b](#)

The **Leap Wallet team** removed the file containing the TO-DO comment.



ADDITIONAL INFORMATION



4.1 Recommended improvements on the code

During the code review phase of the application, it could be possible to advise an incorrect functionality of the application on the `GET /txs` functionality. The team was alerted about the issue, and the recommendation proposed by the security analyst was pushed to the code and to the production application. (Commit: `d725de8aa5077c28afa4d794931e47caaa7d275e`)

```

@@ -10,7 +10,7 @@ const router = express.Router();
10
11 router.get('/:txHash', async
12   (req, res) => {
13     const txHash =
14       req.params.txHash;
15     - return
16       res.json(Container.get(Transaction.token).read(txHash));
17   });
18
19 router.post('/', async (req, res)
20   => {
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Figure 11: Recommended commit introduced on the code.



PERFORMED TESTS



5.1 Rate Limitation

Description:

Rate limits define the maximum number of requests that can be sent in a given time frame. By implementation of rate limitation which protect your infrastructure from being overwhelmed by malicious requests and make your infrastructure available for legitimate users.

Goal:

Determine the security measures against the number of requests in a short time frame.

Result:

The vulnerable endpoints already reported above.



THANK YOU FOR CHOOSING

 **HALBORN**

