



---

## Satsbridge frontend Security Review

---

### **Auditors**

Monke, Security Researcher

Sujith somraaj, Security Researcher

**Report prepared by:** Lucas Goiriz

January 29, 2025

# Contents

<b>1</b>	<b>About Spearbit</b>	<b>2</b>
<b>2</b>	<b>Introduction</b>	<b>2</b>
<b>3</b>	<b>Risk classification</b>	<b>2</b>
3.1	Impact . . . . .	2
3.2	Likelihood . . . . .	2
3.3	Action required for severity levels . . . . .	3
<b>4</b>	<b>Executive Summary</b>	<b>4</b>
<b>5</b>	<b>Findings</b>	<b>5</b>
5.1	Low Risk . . . . .	5
5.1.1	Internal IP address disclosure via misconfigured Nginx routing . . . . .	5
5.1.2	Implement DNSSEC, DMARC and DKIM to enhance DNS Security . . . . .	5
5.1.3	Response Security Headers not configured correctly . . . . .	6
5.1.4	CORS - ACAO Header Reflects Origin Request Header . . . . .	6
5.2	Informational . . . . .	7
5.2.1	Remove unused dependencies . . . . .	7
5.2.2	Add sitemap.xml and robots.txt to improve SEO . . . . .	8
5.2.3	Improve image rendering performance by adding missing attributes to img tag . . . . .	8
5.2.4	Use lazy loading to improve load times . . . . .	8
5.2.5	Deployment serving repository root, rather than built JS files . . . . .	9
5.2.6	Publicly accessible Swagger UI instance . . . . .	10
5.2.7	Nginx Routing via Substrings and 404 page differential . . . . .	10
5.2.8	Application not using compiled React . . . . .	11
5.2.9	Approve funds only if current allowance if less than required allowance . . . . .	12
5.2.10	Incorrect base URL configuration . . . . .	12
5.2.11	Failure to handle state in all transaction paths degrades UX . . . . .	13
5.2.12	Update outdated dependencies using npm update . . . . .	13
5.2.13	Validate counterparty address in client application . . . . .	14

# 1 About Spearbit

Spearbit is a decentralized network of expert security engineers offering reviews and other security related services to Web3 projects with the goal of creating a stronger ecosystem. Our network has experience on every part of the blockchain technology stack, including but not limited to protocol design, smart contracts and the Solidity compiler. Spearbit brings in untapped security talent by enabling expert freelance auditors seeking flexibility to work on interesting projects together.

Learn more about us at [spearbit.com](https://spearbit.com)

## 2 Introduction

StatsBridge is a decentralized aggregator of wrapped BTC liquidity across EVMs, allowing permissionless on/off-ramp to Bitcoin L1 and CEXes for unified BTC.

*Disclaimer:* This security review does not guarantee against a hack. It is a snapshot in time of Satsbridge frontend according to the specific commit. Any modifications to the code will require a new security review.

## 3 Risk classification

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

### 3.1 Impact

- Critical - an exploit that results in full compromise of the system or network. The attacker gains root/admin-level access. For example (but not limited to) extracting or deleting sensitive data, taking complete control over the infrastructure, or shutting down or severely degrading essential services. *Must* fix as soon as possible (if already deployed).
- High - an exploit that grants unauthorized access to sensitive data or services, potentially causing significant disruption or damage. The attacker does not gain root access but can still perform malicious activities. For example (but not limited to) unauthorized access to confidential data (as defined by the client), ability to modify, delete, or extract non-public data, or perform actions that can degrade user services significantly.
- Medium - a vulnerability that can be exploited to gain unauthorized access to the system, but with limited impact on data or service availability. For example (but not limited to) gaining the ability to bypass certain security controls (e.g., access controls, authentication), or the potential to impact service performance or availability, but not severely.
- Low - losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired.
- Optimizations - Suggestions around design or code improvements that improve performance.
- Informational - Suggestions around best practices or readability.

### 3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

### **3.3 Action required for severity levels**

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

## 4 Executive Summary

Over the course of 7 days in total, [SatsBridge](#) engaged with [Spearbit](#) to review the [satsbridge-frontend](#) protocol. In this period of time a total of **17** issues were found.

### Summary

<b>Project Name</b>	SatsBridge
<b>Repository</b>	<a href="#">satsbridge-frontend</a>
<b>Commit</b>	<a href="#">c5bbccd9</a>
<b>Type of Project</b>	Web2, Frontend
<b>Audit Timeline</b>	Jan 4th to Jan 11th

### Issues Found

<b>Severity</b>	<b>Count</b>	<b>Fixed</b>	<b>Acknowledged</b>
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	4	4	0
Optimizations	0	0	0
Informational	13	12	1
<b>Total</b>	<b>17</b>	<b>16</b>	<b>1</b>

## 5 Findings

### 5.1 Low Risk

#### 5.1.1 Internal IP address disclosure via misconfigured Nginx routing

**Severity:** Low Risk

**Context:** (No context files were provided by the reviewer)

**Description:** Satsbridge uses an Nginx server as a reverse proxy. One service that this points to is a Swagger UI instance. Upon visiting <https://satsbridge.com/doc>, a status 308 redirection occurs. Due to a misconfiguration, this redirection discloses the internal IP address and port of the Swagger instance.

**Proof of Concept:**

```
HTTP/1.1 308 PERMANENT REDIRECT
Server: nginx/1.18.0
Date: Mon, 06 Jan 2025 14:22:20 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 237
Location: https://satsbridge.com/doc/
Connection: keep-alive
Access-Control-Allow-Origin: *

<!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL: <a href="http://
10.0.0.2:8079/doc/">http://10.0.0.2:8079/doc/</a>. If not, click the link.
```

**Recommendation:** Remove public access to the Swagger UI, unless there is a need to have this service exposed.

**Satsbridge:** Fixed in commit [4431badb](#).

**Spearbit:** Fixed.

#### 5.1.2 Implement DNSSEC, DMARC and DKIM to enhance DNS Security

**Severity:** Low Risk

**Context:** (No context files were provided by the reviewer)

**Description:** The existing DNS setup for the SatsBridge front-end applications utilizes DigitalOcean nameservers configured with standard TTL settings. The A record (164.92.135.173) is set to a TTL of 3600 seconds, while the nameserver records have a TTL of 1769 seconds. However, the absence of **DNSSEC** implementation puts the domain at risk of DNS hijacking attacks, which are increasingly common among web3 frontend vulnerabilities.

Email services use Google Workspace (MX: smtp.google.com, priority 1, TTL 14400s) with SPF protection. However, missing **DKIM** and **DMARC** configurations significantly weaken email security. The SPF record (v=spf1 include:\_spf.google.com -all) implements a strict policy but lacks complementary authentication measures.

**Recommendation:**

- Consider incorporating DNSSEC, DKIM, and DMARC into your DNS setup to mitigate standard attack methods.
- Additionally, verify the locking of the domain transfer and set up non-SMS two-step authentication for the domain owner's account at the registrar.

**SatsBridge:** Moving to Cloudflare now. By the way, the pentest tool shows that DKIM is ok. Only Google checks fail.

**Spearbit:** DNSSEC, DKIM and DMARC records are found and verified.

### 5.1.3 Response Security Headers not configured correctly

**Severity:** Low Risk

**Context:** *(No context files were provided by the reviewer)*

**Description:** As noted in Telegram, the response headers are not correctly configured..

**Proof of Concept:** The server does not respond with optimal security headers.

**Recommendation:.**

- Implement a minimally permissive CORS policy to prevent CORs-based attacks.
- Add COOP policy.
- Add a strict CSP.
- Add X-Frame-Options.

We are happy to discuss further security configuration in relation to this, alongside reasoning, in the closing meeting. We highly recommended implementing these headers as they will prevent the majority of frontend attacks in the future, should a vulnerability like XSS occur.

**Satsbridge:** Fixed on the [frontend](#) implementation.

**Spearbit:** Fixed.

### 5.1.4 CORS - ACAO Header Reflects Origin Request Header

**Severity:** Low Risk

**Context:** *(No context files were provided by the reviewer)*

**Description:** There is a CORS header misconfiguration on the main HTLC API. Currently, the server returns two CORs headers, so it's not vulnerable, but the first of these two headers is still reflected from the requesting domain's `Origin` request header.


**Recommendation:** Return *only* the response header `Access-Control-Allow-Origin: https://satsbridge.com`. Duplicating headers or allowing other origins may pose unnecessary risks.

The server is correctly validating the Content-Type strictly, so CSRF is not possible here. However, additional fields in the JSON do not trigger errors, so we'd recommend handling those error cases correctly and strictly checking the data being passed in the JSON body. This is to avoid a scenario where it may be vulnerable to CSRF if the Content-Type validation is removed at some point in the future.

**Recommended Fixes:.**

- Set `Access-Control-Allow-Origin: https://satsbridge.com` and do not duplicate the ACAO header.
- Add stricter checks to the JSON body being parsed - for example, disallowing key-pair values that aren't used by the server, and ensuring that the format of existing key-pair values is checked rigorously.

**Satsbridge:** Access-Control-Allow-Origin: <https://satsbridge.com>. The duplication of headers has been removed:

▼ General	
Request URL:	<a href="https://satsbridge.com/htlc/arbitrum/in/init">https://satsbridge.com/htlc/arbitrum/in/init</a>
Request Method:	POST
Status Code:	● 200 OK
Remote Address:	[2606:4700:3034::6815:558b]:443
Referrer Policy:	strict-origin-when-cross-origin
▼ Response Headers	
Access-Control-Allow-Origin:	<a href="https://satsbridge.com">https://satsbridge.com</a>
Alt-Svc:	h3=":443"; ma=86400
Cf-Cache-Status:	DYNAMIC 
Cf-Ray:	905b2b35aab7f23d-GRU
Content-Encoding:	zstd
Content-Security-Policy:	object-src 'none'; base-uri 'none'
Content-Type:	application/json
Date:	Tue, 21 Jan 2025 23:43:51 GMT
Nel:	{"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Report-To:	{"endpoints": [{"url":"https://a.nel.cloudflare.com/report/v4?"

**Spearbit:** Fixed.

## 5.2 Informational

### 5.2.1 Remove unused dependencies

**Severity:** Informational

**Context:** [package.json#L13](#)

**Description:** The `package.json` file includes numerous dependencies that have been installed but are not used anywhere in the application. Installing unwanted dependencies will not result in issues if proper bundler settings are used.

However, installing unwanted dependencies can lead to other problems including:

- Performance implications: Bigger project sizes may result in longer build times.
- Reduced code quality: A lengthy list of dependencies, even if not in use, complicates identifying which packages are crucial for your project.

**Recommendation:** Consider removing any unused dependencies. Incorporate tools such as `depcheck` into the CI pipeline to prevent the installation of unnecessary dependencies.

**SatsBridge:** Fixed in commit [829ac2ae](#).

**Spearbit:** Verified fix.



### 5.2.2 Add `sitemap.xml` and `robots.txt` to improve SEO

**Severity:** Informational

**Context:** *(No context files were provided by the reviewer)*

**Description:** The current implementation lacks the `sitemap.xml` and `robots.txt` files in the `/public` folder. These files are essential for enhancing search engine crawling and indexing. Search engines heavily rely on these files to crawl and index websites efficiently.

Without them:

- Search engines may miss important pages.
- Crawl budget is wasted on non-essential pages.
- Site ranking potential is limited.
- Page discovery and indexing is slower.

**Recommendation:** Add the `robots.txt` and `sitemap.xml` files to improve SEO.

**SatsBridge:** Fixed in commit [e8714cfc](#).

**Spearbit:** Verified fix.

### 5.2.3 Improve image rendering performance by adding missing attributes to `img` tag

**Severity:** Informational

**Context:** [partners.tsx#L30](#)

**Description:** The current `img` tag misses a few attributes necessary to optimize the browser behavior while rendering such images. The current implementation looks like the following, where certain attributes are missing:

```
<img alt={l.alt} className={s.PartnersImage} src={l.icon} />
```

The recommended Implementation is:

```
<img
  alt={l.alt}
  className={s.PartnersImage}
  src={l.icon}
  height={100}
  width={100}
  loading="lazy"
  decoding="async"
/>
```

Adding the new attributes could reduce browser reflow operations, improve page loading, and prevent Content Layout Shifts (CLS).

**Recommendation:** Consider optimizing the image by adding `height`, `width`, `loading` and `decoding` attributes.

**SatsBridge:** Fixed in commit [47523fe8](#).

**Spearbit:** Verified fix.

### 5.2.4 Use lazy loading to improve load times

**Severity:** Informational

**Context:** [router.tsx#L6](#)

**Description:** The current implementation uses a basic React Router setup with eagerly loaded components. This approach has several implications:

- All page components are bundled together and loaded on the initial application load.
- Bundle size is more significant than necessary for the initial page load.
- No code-splitting benefits are being utilized.
- Has a significant impact on the page load times due to unnecessary memory usage from unused components.

**Recommendation:** Consider implementing lazy loading to improve performance.

```
import { lazy, Suspense } from 'react'
import { RouteObject, RouterProvider, createBrowserRouter } from 'react-router-dom'

// Lazy load components
const HomePage = lazy(() => import('~pages/home-page')).then(module => ({
  default: module.HomePage
})))
const NotFoundPage = lazy(() => import('~pages/not-found')).then(module => ({
  default: module.NotFoundPage
})))

// Loading fallback component
const LoadingSpinner = () => (
  <div className="flex items-center justify-center h-screen">
    <div className="animate-spin rounded-full h-16 w-16 border-t-2 border-b-2 border-gray-900"></div>
  </div>
)

const routes: RouteObject[] = [
  {
    element: (
      <Suspense fallback={<LoadingSpinner />>
        <HomePage />
      </Suspense>
    ),
    path: '/',
  },
  {
    element: (
      <Suspense fallback={<LoadingSpinner />>
        <NotFoundPage />
      </Suspense>
    ),
    path: '/*',
  },
]

const router = createBrowserRouter(routes)

export const Router = () => {
  return <RouterProvider router={router} />
}
```

**SatsBridge:** Fixed in commit [3ed45a55](#).

**Spearbit:** Verified fix.

### 5.2.5 Deployment serving repository root, rather than built JS files

**Severity:** Informational

**Context:** (No context files were provided by the reviewer)

**Description:** The React application currently serves the entire repository - this can be seen when visiting <https://satsbridge.com/Dockerfile>. This also causes the application to load all of the source files, slowing down application performance.

Given that no sensitive data is currently stored in the repository, this is not currently an immediate risk - but it is possible that sensitive information may be pushed in the future.

**Proof of Concept:**

- [Dockerfile](#).
- [vite.config.ts](#).
- [App.tsx](#).

**Recommendation:** This is a React application, so only the built files need to be used in production. Serving the whole repository can lead to information disclosure.

**Satsbridge:** The issue has been resolved by serving only the built files in production; see commit [fa65cd37](#).

**Spearbit:** Fixed.

### 5.2.6 Publicly accessible Swagger UI instance

**Severity:** Informational

**Context:** *(No context files were provided by the reviewer)*

**Description:** There is a Swagger UI instance at <https://satsbridge.com/doc/> and JSON available at <https://satsbridge.com/swagger.json>. This should not be public - especially considering that it documents the Lightning node proxy authentication endpoint.

**Proof of Concept:**

- [Doc](#).
- [swagger.json](#).

**Recommendation:** Remove access to Swagger UI entirely, unless there is a need for it. While this version is secure, Swagger UI has historically suffered from XSS vulnerabilities - allowing public access to Swagger UI may lead to vulnerabilities in the future.

**Satsbridge:** Fixed in commit [4431badb](#).

**Spearbit:** Fix verified.

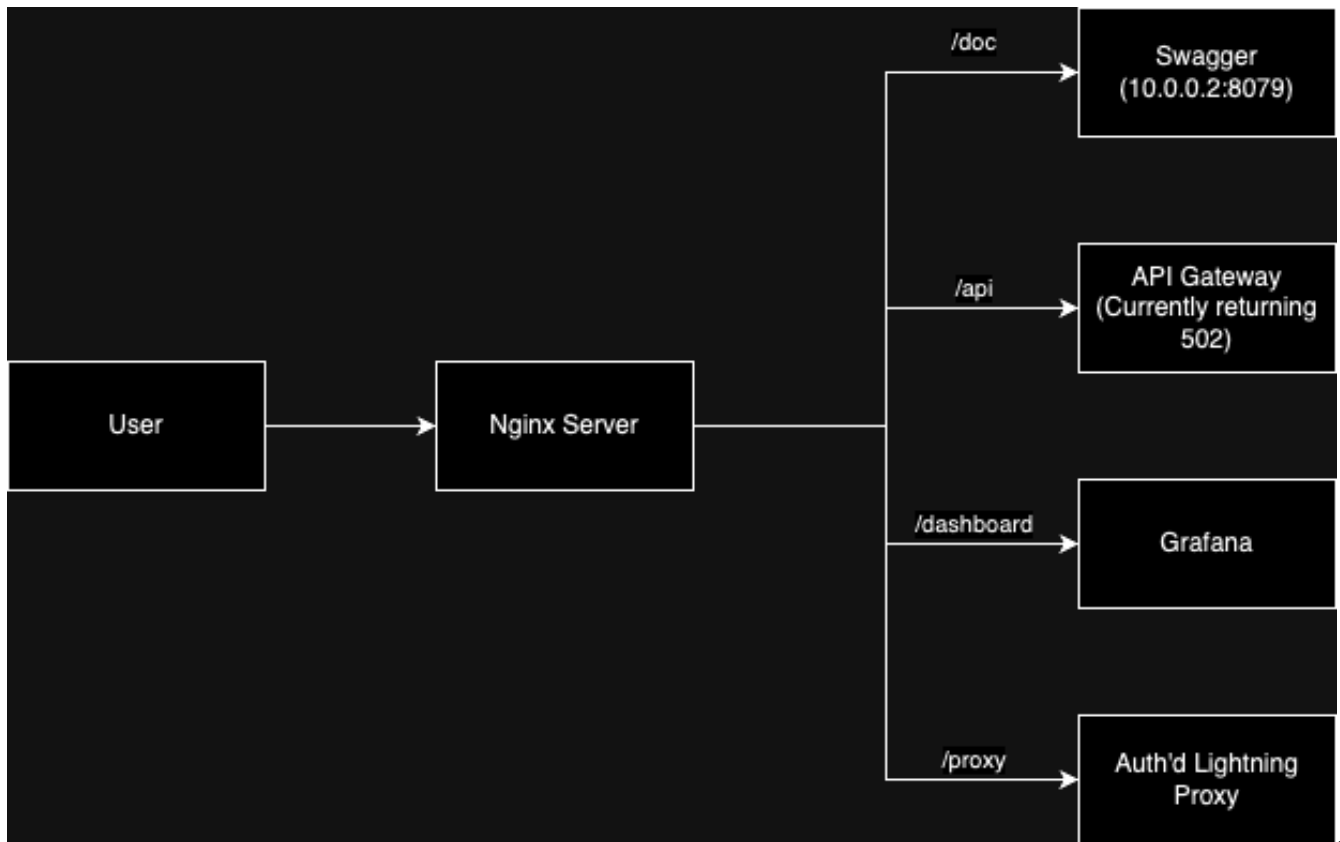
### 5.2.7 Nginx Routing via Substrings and 404 page differential

**Severity:** Informational

**Context:** *(No context files were provided by the reviewer)*

**Description:** The Nginx application is routing via substrings. This makes it much easier for attackers to fuzz the webserver for filenames, as valid substrings elicit a different 404 response due to the reverse proxy configuration. This allows an attacker to determine when a given substring is part of a valid path being served by the reverse proxy.

**Proof of Concept:**



*A rough estimate of the infrastructure layout.*

In this case, any path containing `/doc` such as `/docxyz` will cause a substring match to the `/doc` pattern, but return 404 Not Found - but this 404 page is the Nginx 404 page rather than the Satsbridge 404 page. This gives us an oracle to determine whether or not certain routes exist in the reverse proxy configuration.

**Recommendation:** Return the same 404 page universally from the reverse proxy.

**Satsbridge:** Nginx configured for unified page 404 <https://satsbridge.com/xxx>; see commit [4431badb](#).

**Spearbit:** Fix verified.

### 5.2.8 Application not using compiled React

**Severity:** Informational

**Context:** *(No context files were provided by the reviewer)*

**Description:** The application has compiled the React application, but does not use it in production. Typescript files are not valid Javascript; they should not be imported like scripts.

**Proof of Concept:**

- Compiled script that is not being used: <https://satsbridge.com/dist/assets/index-945b7326.js>.
- Imported TSX:

```

<!DOCTYPE html>
<html lang="en"> scroll
  > <script type="text/javascript"> ... </script>
  > <head> ... </head>
  > <body class="bg-dark">
... > <div id="root"> ... </div> flex == $0
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>

```

**Recommendation:** The compiled JS should be used in production, rather than source code files.

**Satsbridge:** The issue has been resolved by serving only the built files in production; see [fa65cd37](#).

**Spearbit:** Fix verified.

### 5.2.9 Approve funds only if current allowance if less than required allowance

**Severity:** Informational

**Context:** [useWbtcToLn.ts#L222](#)

**Description:** The `handleMetaMaskPayment` function within the `useWbtcToLn` hook facilitates an outgoing bridging transaction from WBTC to LNBTC. For this transaction, the user must authorize their funds to an address indicated in the return parameter of the `htlc/{network}/out/init` API call.

However, this approval call is made without checking the user's current allowance, leading to double approval, which can be expensive in networks like Ethereum..

**Recommendation:** Consider implementing an allowance check before the approval call to avoid spending unnecessary gas.

```

useEffect(() => {
  // ...
  if (walletMetaMaskBalance && (walletMetaMaskBalance?.value >= amountWbtc)) {
+    const currentAllowance = /// fetch the current allowance of wbtc to the target;
+    if(currentAllowance < amountWbtc) {
      await approveFundsWbtc(amountWbtc, network)
+    }
      setWbtcToLnWithoutAlby(3)
    // ...
  })
}

```

**SatsBridge:** Fixed in commit [b258a031](#).

**Spearbit:** Verified fix.

### 5.2.10 Incorrect base URL configuration

**Severity:** Informational

**Context:** *(No context files were provided by the reviewer)*

**Description:** The base URL configuration for the project includes a duplicate domain (<https://satsbridge.com/satsbridge.com>). Nevertheless, the requests still succeed even with this malformed URL structure. This indicates a possible configuration mistake and could lead to reduced code maintainability.

X Headers Payload Preview Response Initiator Timing Cookies	
▼ General	
Request URL:	https://satsbridge.com/satsbridge.com/htlc/arbitrum/out/init
Request Method:	POST
Status Code:	● 200 OK
Remote Address:	164.92.135.173:443
Referrer Policy:	strict-origin-when-cross-origin

**Recommendation:** Consider configuring the base URL to a single domain: <https://satsbridge.com/>.

**SatsBridge:** Fixed the issue by changing the base URL in commit [9805b9f2](#).

**Spearbit:** Verified fix.

### 5.2.11 Failure to handle state in all transaction paths degrades UX

**Severity:** Informational

**Context:** *(No context files were provided by the reviewer)*

**Description:** To enable bridging using the satsbridge application, the user flow for bridging WBTC to LNBTC is defined as follows:

```
Enter input value
> accept invoice
  > approve WBTC
    > create a new contract
      > wait for settlement
```

The user may choose to exit at any stage for various reasons. For instance, they might have created a bridging transaction with an incorrect amount and wish to cancel the flow, intending to restart it after approving the WBTC, or accidentally canceling the approval transaction. The optimal user experience is to reset the state, enabling the user to continue from that specific point.

Instead, the app currently freezes out, requiring a hard refresh. After refreshing the wallet, I noticed that connections don't persist, leading to degraded UX..

**Recommendation:** Consider implementing proper state management in all possible paths of the transaction flow.

**SatsBridge:** Thank you for your valuable feedback. We acknowledge the importance of proper state management in all transaction paths to ensure a seamless user experience. We have changed the swap process from wBTC to Lightning as part of our ongoing improvements. Specifically, a new variable has been introduced to monitor the state (status) of the swap, ensuring better control and handling of user actions across different stages.

**Spearbit:** Acknowledged.

### 5.2.12 Update outdated dependencies using `npm update`

**Severity:** Informational

**Context:** *(No context files were provided by the reviewer)*

**Description:** The existing front-end implementation features several outdated dependencies. Additionally, it shows that various packages need updates, with multiple major version upgrades accessible..

Package	Current	Wanted	Latest
@hookform/resolvers	3.9.1	3.10.0	3.10.0
@reduxjs/toolkit	2.3.0	2.5.0	2.5.0
@testing-library/react	16.0.1	16.1.0	16.1.0
@types/react	18.3.12	18.3.18	19.0.4
@types/react-dom	18.3.1	18.3.5	19.0.2
@typescript-eslint/eslint-plugin	8.16.0	8.19.1	8.19.1
@typescript-eslint/parser	8.16.0	8.19.1	8.19.1
@vitest/ui	2.1.6	2.1.8	2.1.8
@yudiel/react-qr-scanner	2.0.8	2.1.0	2.1.0
dotenv	16.4.5	16.4.7	16.4.7
ethers	6.13.4	6.13.5	6.13.5
framer-motion	11.12.0	11.16.3	11.16.3
jsdom	25.0.1	25.0.1	26.0.0
qrcode.react	3.2.0	3.2.0	4.2.0
react	18.3.1	18.3.1	19.0.0
react-dom	18.3.1	18.3.1	19.0.0
react-hook-form	7.53.2	7.54.2	7.54.2
react-redux	9.1.2	9.2.0	9.2.0
react-router-dom	7.0.1	7.1.1	7.1.1
react-toastify	10.0.6	10.0.6	11.0.2
rollup-plugin-visualizer	5.13.1	5.14.0	5.14.0
sass	1.81.0	1.83.1	1.83.1
stylelint	16.10.0	16.12.0	16.12.0
typescript	4.2.4	4.9.5	5.7.3
vite	6.0.1	6.0.7	6.0.7
vite-bundle-analyzer	0.15.2	0.15.2	0.16.0
vite-tsconfig-paths	5.1.3	5.1.4	5.1.4
vitest	2.1.6	2.1.8	2.1.8
zod	3.23.8	3.24.1	3.24.1

**Recommendation:** Use `npm audit` to find outdated dependencies and update them to latest using `npm update`.

**SatsBridge:** Fixed in commits [8844c386](#) and [ac1cc4fd](#).

**Spearbit:** Verified fix.

### 5.2.13 Validate `counterparty` address in client application

**Severity:** Informational

**Context:** *(No context files were provided by the reviewer)*

**Description:** The client application calls the <https://satsbridge.com/htlc/arbitrum/out/init> endpoint to initiate the first step of bridging from the EVM network to the Lightning Network. This provides the counterparty address where users can approve and transfer their WBTC.

Post approval, the funds are moved to the counterparty and an invoice is created on the lightening network, which gets paid and the bridging transaction gets completed.

During this process, the client application blindly trusts the counterparty address given by the API without any validation. This lack of scrutiny can be risky, as a compromised address could lead users to approve a malicious contract, resulting in the theft of their funds.

**Recommendation:** Consider keeping an address list on the client side and updating it dynamically whenever new counterparty addresses are deployed. Validate these against the API response to address the issue.

**SatsBridge:** Fixed in commits [99b189e9](#) and [fb4d8fcf](#).

**Spearbit:** Verified fix.