coinspect

You build, we defend.

flare

Source Code Audit

Governance Poll Frontend

November 2024

Governance Poll Frontend - Wallet Integration

Source Code Audit

Version: v241112 Prepared for: Flare November 2024

Security Assessment

1. Executive Summary

In **November 2024**, **Flare** engaged Coinspect to perform a Source Code Audit of the Governance Poll Frontend code in charge of wallets integration. The purpose of this review is to identify any potential risks for users when initiating staking and voting related transactions with the various supported wallets.

	✓ Solved	⚠ Caution Advised	✗ Resolution Pending
High	0	0	0
Medium	0	0	0
Low	0	0	3
No Risk	0	2	0
Total	**0**	**2**	**3**

The assessment identified three low-risk issues related to using type 0 transactions on Ledger devices, which result in wasted fees, relying on a non-production-ready Ledger library, and using the `eth_sign` method, which could lead to users unknowingly signing wallet-draining transactions.

2. Summary of Findings

This section provides a concise overview of all the findings in the report grouped by remediation status and sorted by estimated total risk.

2.1 Findings with pending resolution

These findings indicate potential risks that require some action. They must be addressed with modifications to the codebase or an explicit acceptance as part of the project's known security risks.

Id	Title	Risk
GPF-001	Reliance on eth_sign unnecessarily exposes users to signing draining transactions	Low
GPF-002	Ledger users incurring unnecessary transaction fees due to the use of type 0 transactions	Low
GPF-003	Flare Ledger application is not production ready	Low

2.2 Finding where caution is advised

Issues with risk in this list have been addressed to some extent but not fully mitigated. Any future changes to the codebase should be carefully evaluated to avoid exacerbating these issues or increasing their probability.

Findings with a risk of `None` pose no threat, but document an implicit assumption which must be taken into account. Once acknowledged, these are considered solved.

Id	Title	Risk
GPF-004	Usage of dangerouslySetInnerHTML	None

GPF-005	Governance voting related transactions execution at risk due to fixed gasPrice	None

3. Scope

The scope was set to be the source code at https://gitlab.com/flarenetwork/governance-poll-fe/src/wallets on commit **abb8f59a59ef0a737b05a41dbfa22f7c30605e9e**.

4. Assessment

The present assessment focused on the portion of code from the Governance Poll Frontend in charge of interacting with the different supported wallets to enable users to delegate their tokens to FTSO data providers and participate in governance voting.

Specifically, the frontend integrates with extension wallets and wallets supporting Wallet Connect via the `web3-react` component, and Ledger devices via the two different Ledger applications supporting Flare transactions (https://github.com/zondax/ledger-flare and https://github.com/ava-labs/ledger-avalanche-js).

In reviewing the in-scope code, Coinspect primarily focused on ensuring that users are not at risk of inadvertently signing arbitrary transactions or transaction hashes. Additionally, Coinspect verified that transactions are created with the correct gas price to avoid overpaying or underpaying, as the latter could prevent transactions from being executed. Potential transaction replays were also considered as part of the threat model.

All transactions except those signed using Ledger devices are generated via the `typechain-ethers` library which automates the calculation of the transaction fees based on the current base and priority fees.

4.1 Security assumptions

Coinspect consultants identified the following assumptions:

- Transactions processed by the code within scope, generated by out-of-scope code, are correctly formed.
- The Ledger applications referenced in the previous section are secure and function properly.

4.2 Testing

No tests were identified in the repository containing the code under scope. Coinspect recommends developing a unit testing suite to ensure reliability and verify that code operates as expected.

5. Detailed Findings

GPF-001

Reliance on eth_sign unnecessarily exposes users to signing draining transactions

Status	Risk
Resolution Pending	**Low**
✗	🟥⬜⬜
	Impact **Low**
Resolution	Likelihood **Low**
Open	

Location

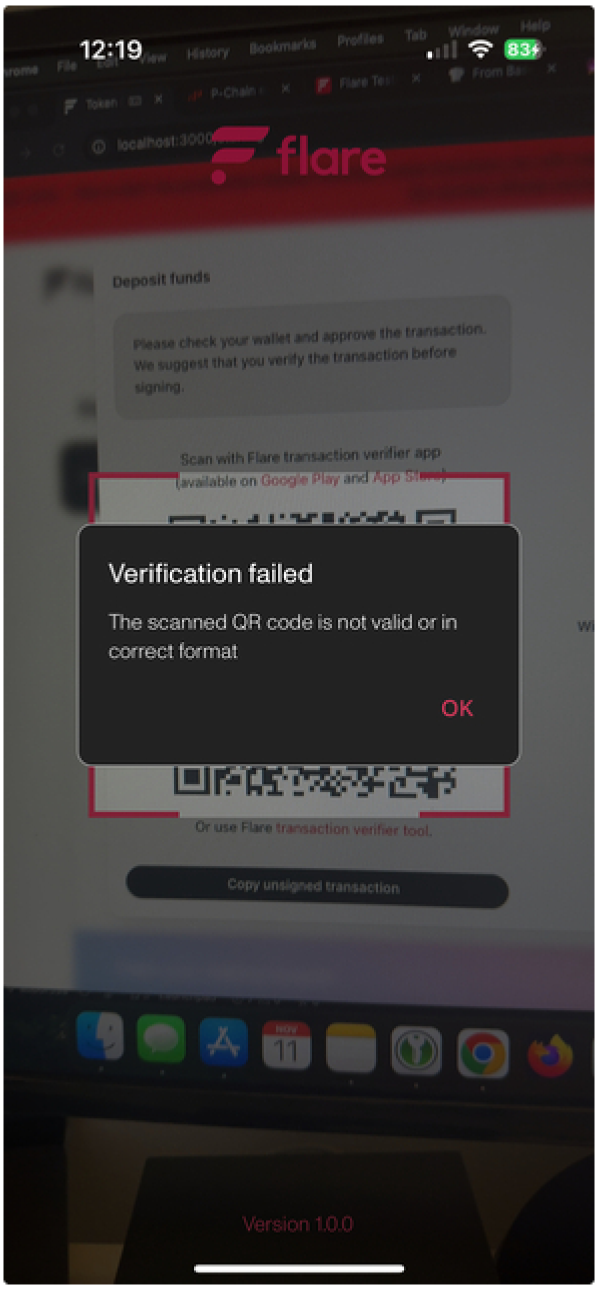
```
governance-poll-fe/src/staking/generalFunctions.ts:12
```

Description

The `eth_sign` method enables signing any arbitrary hash, potentially including one for a wallet-draining transaction.

The frontend generates import and export transactions between the C and P chains, which require the user's signature via the software wallet's `eth_sign` method or the `ledgerSignHash` function when using the Avalanche Ledger app. In contrast, the Flare Ledger app enables users to verify transactions directly on the Ledger device screen, eliminating the need to blindly sign hashes.

Although Flare offers a mechanism for users to verify transactions through the Flare transaction verifier mobile app —which successfully worked during testing with Ledger— consultants were unable to verify transactions using a browser extension wallet (Metamask), as demonstrated in the screenshot below.



Please check your wallet and approve the transaction. We suggest that you verify the transaction before signing.

Scan with Flare transaction verifier app (available on Google Play and App Store)

Verification failed

The scanned QR code is not valid or in correct format

Or use Flare transaction verifier tool.

Recommendation

Consider only allowing these transactions via the Flare Ledger app. Otherwise, ensure that users can verify transactions and hashes in detail before signing, regardless of the wallet used.

The Flare transaction verifier app should issue a more prominent warning when unable to parse a QR code, as this could indicate suspicious activity.

Status

Open.

GPF-002

Ledger users incurring unnecessary transaction fees due to the use of type 0 transactions

Status
Resolution Pending

Resolution
Open

Risk
Low

Impact
Low

Likelihood
Low

Location

`governance-poll-fe/src/wallets/connectors/ledger/ledgerConnector.ts`

Description

Ledger users are required to sign type 0 transactions, which results in them paying higher fees than necessary and thereby wasting funds.

The `finalizeAndConvertEvmTx` function, responsible for crafting type 0 transactions to be signed via Ledger, sets the `gasPrice` as `maxFeePerGas - maxPriorityFeePerGas`.

```
export async function finalizeAndConvertEvmTx(
    ...
): Promise<IFinalizeEvm> {
    ...
   let feeEstimate = await estimateEIP1559Fee(supportedChainId);
   let maxFeePerGas = feeEstimate[0];
```

```
    let maxPriorityFeePerGas = feeEstimate[1];
     ...
    if (txType == 0) {
       let common = Common.custom({ chainId });
       let gasPrice = maxFeePerGas - maxPriorityFeePerGas;
```

From the snippet below, the `maxFee` equals twice the `baseFee` plus the `priorityFee`.

```
export async function estimateEIP1559Fee(supportedChainId:
AllSupportedChainsType): Promise<[bigint, bigint]> {
   let priorityFee = (gasUsedRatio / 4 < 0.9 ? priorityFee10 :
priorityFee50) / BigInt(100);
   let maxFee = BigInt(2) * baseFee + priorityFee;
   return [maxFee, priorityFee];
}
```

This results in the `gasPrice` being set at 2x the `baseFee`. As highlighted in this post, the `gasPrice` for legacy pre-1559 transactions includes both the `baseFee` and the `priorityFee`.

Recommendation

Add support for EIP-1559 Type 2 transactions on Ledger devices.

Status

Open.

GPF-003

Flare Ledger application is not production ready

Status

Resolution Pending

Resolution

Open

Risk

Low

Impact

Low

Likelihood

Low

Location

```
governance-poll-fe/src/wallets/connectors/ledger/ledgerConnector.ts
```

Description

According to the Flare Ledger application documentation at https://github.com/Zondax/ledger-flare, its usage in production environments is not recommended.

The scope of the present assessment does not include reviewing these external applications, and its current status is therefore unknown.

Using experimental or unvetted applications poses a security risk, as they may contain unpatched vulnerabilities and lack rigorous testing, making them more susceptible to attacks.

Recommendation

Consider not using the application until it has undergone a thorough audit. Alternatively, ensure its risks and limitations are clearly documented.

Status

Open.

GPF-004

Usage of dangerouslySetInnerHTML

Status
Caution Advised

Resolution
Open

Risk
None

Impact
Recommendation

Likelihood
–

Location

```
governance-poll-fe/src/wallets/images/MetamaskIcon.tsx:21
```

Description

The use of dangerouslySetInnerHTML presents a potential for Cross-Site Scripting (XSS) if it is ever provided with arbitrary or untrusted input.

The `dangerouslySetInnerHTML` function can directly insert HTML content into the DOM without escaping, enabling Cross-Site Scripting (XSS) attacks if malicious code is included.

Note however this issue has been deemed informational, as the component in question does not take any dynamic or user-controlled input. As a result, there is no current path for exploitation.

```
dangerouslySetInnerHTML={{
    __html:
        '\n\t.st0{fill:#E2761B;stroke:#E2761B;stroke-
linecap:round;stroke-
linejoin:round;}\n\t.st1{fill:#E4761B;stroke:#E4761B;stroke-
linecap:round;stroke-
linejoin:round;}\n\t.st2{fill:#D7C1B3;stroke:#D7C1B3;stroke-
linecap:round;stroke-
linejoin:round;}\n\t.st3{fill:#233447;stroke:#233447;stroke-
linecap:round;
         ...'
}}
```

Recommendation

Consider removing the `dangerouslySetInnerHTML` function.

Status

Open.

GPF-005

Governance voting related transactions execution at risk due to fixed gasPrice

Status
Caution Advised

Resolution
Open

Risk
None

Impact
Recommendation

Likelihood
-

Location

```
governance-poll-fe/src/wallets/images/MetamaskIcon.tsx:21
```

Description

The `delegate` and `undelegate` calls to the `GovernanceVotePower` contract may fail due to a fixed `gasLimit` transaction parameter. If the gas requirements for executing these functions increase over time, the transactions will not succeed.

Currently, these transactions are capped at `300,000` gas units, as shown below.

```
transaction: govContractFlare.delegate(utils.getAddress(toAddress), {
from: fromAddress, gasLimit: 300_000, ...defaultTxParamsFlare }),
```

It is worth noting that the `GovernanceVotePower` contract was outside the scope of this assessment, and without access to it, Coinspect's consultants were unable to evaluate the potential risk associated with this issue.

Recommendation

The `gasLimit` parameter should be dynamically calculated before each call. Alternatively, ensure that these calls will not demand additional gas in the future.

Status

Open.

6. Disclaimer

The contents of this report are provided "as is" without warranty of any kind. Coinspect is not responsible for any consequences of using the information contained herein.

This report represents a point-in-time and time-boxed evaluation conducted within a specific timeframe and scope agreed upon with the client. The assessment's findings and recommendations are based on the information, source code, and systems access provided by the client during the review period.

The assessment's findings should not be considered an exhaustive list of all potential security issues. This report does not cover out-of-scope components that may interact with the analyzed system, nor does it assess the operational security of the organization that developed and deployed the system.

This report does not imply ongoing security monitoring or guaranteeing the current security status of the assessed system. Due to the dynamic nature of information security threats, new vulnerabilities may emerge after the assessment period.

This report should not be considered an endorsement or disapproval of any project or team. It does not provide investment advice and should not be used to make investment decisions.