2.5 Recommendation Summary

Severity

Issues

	● High	● Medium	● Low	● Informational
Open	0	1	2	0
Resolved		1	2	
Acknowledged				
Partially Resolved				

- **Open**: Unresolved security vulnerabilities requiring resolution.
- **Resolved**: Previously identified vulnerabilities that have been fixed.
- **Acknowledged**: Identified vulnerabilities noted but not yet resolved.
- **Partially Resolved**: Risks mitigated but not fully resolved.

3.0 Checked Vulnerabilities

We examined the smart contract for widely recognized and specific vulnerabilities. Below are some of the common vulnerabilities considered.

Category	**Check Items**
Source Code Review	➔ Reentrancy Vulnerabilities ➔ Ownership Control ➔ Time-Based Dependencies ➔ Gas Usage in Loops ➔ Transaction Sequence Dependencies ➔ Style Guide Compliance ➔ EIP Standard Compliance ➔ External Call Verification ➔ Mathematical Checks ➔ Type Safety ➔ Visibility Settings ➔ Deployment Accuracy ➔ Repository Consistency
Functional Testing	➔ Business Logic Validation ➔ Feature Verification ➔ Access Control and Authorization ➔ Escrow Security ➔ Token Supply Management ➔ Asset Protection ➔ User Balance Integrity ➔ Data Reliability ➔ Emergency Shutdown Mechanism

5.0 Technical Analysis

Medium Severity Issues

Issue #1 [Resolved]

[M-1] Missing Sequencer Uptime Feed Check for L2 Price Feed Queries

Severity

MEDIUM

Location	Functions
~/authentication.ts	NA

Overview:

Authentication security heavily depends on the strength of user credentials. Weak passwords increase the likelihood of successful brute-force attacks, making it easier for attackers to compromise user accounts.

Issue Description

The existing password policy lacks strict enforcement of length and complexity, allowing passwords to be composed of only six numeric characters. This significantly reduces the search space, making password-guessing attacks more feasible.

Impact

A weak password policy increases the risk of unauthorized access. Attackers with local access could potentially brute-force weak passwords, compromising wallet security and user funds.

Recommendation

It is advised to enforce a stronger password and PIN policy to enhance security. Implementing stricter complexity and length requirements will improve account protection and reduce the risk of brute-force attacks.

Status

Issued Fixed

Low Severity Issues

Issue#2 [Resolved]

[L-1]Unremoved Source Map Files in ProductionCheck

Severity

LOW

Location	Functions
~/Global	NA

Overview

The extension's release package contains JavaScript source map (`.js.map`) files, which are primarily meant for debugging and development purposes. However, including these files in the final distribution can introduce security and privacy concerns.

Issue Description

Leaving source map files accessible may expose the extension's original source code, making it easier for attackers to analyze its structure. This could lead to the discovery of sensitive functionalities and potential security weaknesses.

Steps to Verify

1. Open the extension's installation directory (path may vary depending on the operating system).
2. Locate the `.js.map` files within `./static/js/`.

Recommendation

To enhance security, ensure that source map files are removed from the production build, preventing unauthorized access to internal code structures.

Status

Issued Fixed

Issue#3 [Resolved]

[L-2]Wallet Creation via Seed Fails

Severity

LOW

Location	Functions
~/Global	NA

Overview

The wallet setup process using a seed phrase is malfunctioning. After displaying the seed chunks, the system prompts the user to enter the first chunk but fails to recognize any provided input.

Issue Description

A bug in the validation logic prevents the wallet from accepting the seed phrase.

```
const handleSubmit = (e: React.SyntheticEvent) => {
    e.preventDefault();
    loading.setIsLoading(true);
    let target = e.target as typeof e.target & ConfirmForm;
    if (target.input.value) {
        let word = target.input.value.trim();  // Remove extra spaces
but retain case sensitivity

        let index = phrases.length;
        if (verify[index].toLowerCase() !== word.toLowerCase()) {  //
Normalize both for comparison
            loading.setIsLoading(false);
            return setError(!error);
        }
        if (index + 1 === verify.length) {
            target.input.blur();
            return handleCompletion();
        }
        setPhrases((prev) => [...prev, word]);
        loading.setIsLoading(false);
        target.input.value = '';
    }
};
```

Steps to Verify

1. Open the **Extension Wallet** and initiate a new wallet setup.
2. Set and verify a passcode.
3. Choose the **Seed** option and proceed.
4. Copy the displayed seed chunks and click **Proceed**.
5. Enter the first chunk and click **Next Word** – observe that the input is rejected.

Recommendation

Avoid converting user-inputted seed chunks to lowercase before performing the validation. The comparison should account for case sensitivity as stored in the `verify` array.

Status

Issued Fixed