Minibridge
Audit Report

Thu Jul 25 2024

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

Minibridge Audit Report

1 Executive Summary

1.1 Project Information

Description	Mini Bridge is a cross-chain bridge developed by the Chaineye team for small transfer scenarios. It is a public good tool and offers low cross-chain fees and fast confirmation times.
Type	Bridge
Auditors	ScaleBit
Timeline	Wed Jul 10 2024 - Thu Jul 25 2024
Languages	Python
Platform	Minibridge
Methods	Dependency Check, Static Analysis, Manual Review
Source Code	
Commits	

1.2 Files in Scope

The following are the directories of the original reviewed files.

Directory
https://github.com/scalebit/ClientProjects2/confirmtx.py
https://github.com/scalebit/ClientProjects2/blocksyncer.py
https://github.com/scalebit/ClientProjects2/chains_provider.py
https://github.com/scalebit/ClientProjects2/confirmmissingtx.py
https://github.com/scalebit/ClientProjects2/addmissingtx.py
https://github.com/scalebit/ClientProjects2/chains.py
https://github.com/scalebit/ClientProjects2/txprocess.py
https://github.com/scalebit/ClientProjects2/simplebase.py

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	2	1	1
Informational	1	1	0
Minor	0	0	0
Medium	1	0	1
Major	0	0	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Infinite Loop
- Infinite Recursion
- Race Condition
- Traditional Web Vulnerabilities
- Memory Exhaustion Attack
- Disk Space Exhaustion Attack
- Side-channel Attack
- Denial of Service
- Replay Attacks
- Double-spending Attack
- Eavesdropping Attack
- Business Logic Issues
- Coding Style Issues

1.5 Methodology

Our security team adopted **"Dependency Check"**, **"Automated Static Code Analysis"**, and **"Manual Review"** to conduct a comprehensive security test on the code in a manner closest to real attacks. The main entry points and scope of the security testing are specified in the **"Files in Scope"**, which can be expanded beyond the scope according to actual testing needs. The main types of this security audit include:

(1) Dependency Check

A comprehensive check of the software's dependency libraries was conducted to ensure all external libraries and frameworks are up-to-date and free of known security vulnerabilities.

(2) Automated Static Code Analysis

Static code analysis tools were used to find common programming errors, potential security vulnerabilities, and code patterns that do not conform to best practices.

(3) Manual Review

The scope of the code is explained in section 1.2.

(4) Audit Process

- Clarify the scope, objectives, and key requirements of the audit.
- Collect related materials such as software documentation, architecture diagrams, and lists of dependency libraries to provide background information for the audit.
- Use automated tools to generate a list of the software's dependency libraries and employ professional tools to scan these libraries for security vulnerabilities, identifying outdated or known vulnerable dependencies.
- Select and configure automated static analysis tools suitable for the project, perform automated scans to identify security vulnerabilities, non-standard coding, and potential risk points in the code. Evaluate the scanning results to determine which findings require further manual review.
- Based on the results of the preliminary automated analysis, develop a detailed code review plan, identifying the focus of the review. Experienced auditors perform line-by-line reviews of key components and sensitive functionalities in the code.
- If any issues arise during the audit process, communicate with the code owner in a timely manner. The code owners should actively cooperate (this may include providing

the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);

- Necessary information during the audit process will be well documented in a timely manner for both the audit team and the code owner.

2 Summary

This report has been commissioned by Minibridge with the objective of identifying any potential issues and vulnerabilities within the source code of the Minibridge repository, as well as in the repository dependencies that are not part of an officially recognized library. In this audit, we have employed the following techniques to identify potential vulnerabilities and security issues:

(1) Dependency Check

A comprehensive analysis of the software's dependency libraries was conducted using the analysis tools.

(2) Automated Static Code Analysis

The code quality was examined using a code scanner.

(3) Manual Code Review

The primary focus of the manual code review was:

- DeFiEye/minibridge

During the audit, we identified 2 issues of varying severity, listed below.

ID	Title	Severity	Status
RAW-1	`privaterpcs` Use Insecure HTTP Protocol	Informational	Fixed
TXP-1	Centralization Risk	Medium	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the Minibridge repository :

- **Users**: Individuals or institutional users who initiate cross-chain transactions. They wish to transfer assets between different blockchain networks.
- **Source Chain and Target Chain**: The two blockchain networks involved in a cross-chain transaction. The source chain is the blockchain from which the assets are transferred, and the target chain is the blockchain into which the assets are transferred.
- **Smart Contracts**: Smart contracts lock and unlock assets on the source chain and the target chain.
- **Relayers**: Responsible for transmitting cross-chain transaction information between the source chain and the target chain.

4 Findings

RAW-1 `privaterpcs` Use Insecure HTTP Protocol

Severity: Informational

Discovery Methods: Manual Review

Status: Fixed

Code Location:

rawconf.json#411

Descriptions:

In the configuration file `rawconf.json`, one of the privaterpcs uses the insecure HTTP protocol, which poses a risk of man-in-the-middle attacks.

Suggestion:

It is recommended to use the HTTPS protocol or change the RPC address to a local network address.

TXP-1 Centralization Risk

Severity: Medium

Discovery Methods: Manual Review

Status: Acknowledged

Code Location:

txprocess.py#1

Descriptions:

User funds are directly sent to the bridge, who then sends the funds to the target user on another chain.Centralization risk was identified in the bridge.

Suggestion:

It's recommended that measures be taken to reduce the centralization issue like using multi-sig.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information or assets at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information or assets at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer