ICKB
Audit Report

Tue Sep 10 2024

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

ScaleBit

ICKB Audit Report

1 Executive Summary

1.1 Project Information

Description	The inflation-protected CKB (iCKB) is a Nervos L1 xUDT token responsible for protecting users against Nervos secondary issuance inflation like NervosDAO, while at the same time being a liquid asset.
Type	Token
Auditors	ScaleBit
Timeline	Tue Sep 03 2024 - Tue Sep 10 2024
Languages	Rust
Platform	CKB
Methods	Dependency Check, Static Analysis, Manual Review
Source Code	https://github.com/ickb/v1-core
Commits	454cfa966052a621c4e8b67001718c29ee8191a2

1.2 Files in Scope

The following are the directories of the original reviewed files.

Directory
https://github.com/ickb/v1-core/scripts
https://github.com/ickb/v1-core/scripts/contracts
https://github.com/ickb/v1-core/scripts/contracts/utils
https://github.com/ickb/v1-core/scripts/contracts/ickb_logic
https://github.com/ickb/v1-core/scripts/contracts/owned_owner
https://github.com/ickb/v1-core/scripts/contracts/limit_order

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	3	1	2
Informational	2	1	1
Minor	1	0	1
Medium	0	0	0
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):

- Integer overflow/underflow
- 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
- Eclipse Attack
- Sybil Attack
- Eavesdropping Attack
- Business Logic Issues
- Contract Virtual Machine Vulnerabilities
- 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.
- Design a series of fuzz testing cases aimed at testing the software's ability to handle exceptional data inputs. Analyze the issues found during the testing to determine the defects that need to be fixed.
- 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 iCKB with the objective of identifying any potential issues and vulnerabilities within the source code of the ICKB 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 dependency check tool.

(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:

- https://github.com/ickb/v1-core

During the audit, we identified 3 issues of varying severity, listed below.

ID	Title	Severity	Status
ENT-1	Busywork Attack	Informational	Acknowledged
UTI-1	Handling of LengthNotEnough Error	Informational	Fixed
ENT1-1	Confusion attack on Limit Order	Minor	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the ICKB repository :
iCKB is a protocol designed to address the illiquidity issue associated with NervosDAO by tokenizing NervosDAO deposits into a liquid iCKB token. The iCKB protocol holds all iCKB deposits and maintains a pool of these deposits, enabling anyone to use anyone else's deposit to exit the NervosDAO once it's mature.
The exchange rate between iCKB and CKB is structured to be deterministic, tracking the inflation rate from CKB secondary issuance as defined by the NervosDAO compensation formula. The iCKB protocol encourages a standard deposit size of 100,000 iCKB to enhance deposit fungibility and mitigate certain types of attacks.
Furthermore, a limit order script is implemented to address the constraints of the underlying NervosDAO and iCKB protocols, facilitating a more flexible and user-friendly experience within the iCKB ecosystem.

4 Findings

ENT-1 Busywork Attack

Severity: Informational

Discovery Methods:

Status: Acknowledged

Code Location:

scripts/contracts/ickb_logic/src/entry.rs#21-37

Descriptions:

Assuming that the user has a large corpus, he can consume all those whose maturity is near (within one hour/few days) depending on how big is his capital. Depending on the amount of capital used for the attack, this could reduce the quality of the service for everyone, as the only remaining deposits would be those whose maturity date is a bit more far away, so this could hamper the protocol fruition.

Suggestion:

This issue may be associated with the design idea of the protocol. Provided that the iCKB pool is sufficiently large, the potential impact of such an attack will be significantly reduced, while the cost of executing the attack will increase. Consequently, this type of attack is not considered to pose a substantial threat. However, due to the absence of effective remedies to fix it effectively, it counts as a security issue raised. This issue has already been discussed with the project team at GitHub issue

UTI-1 Handling of LengthNotEnough Error

Severity: Informational

Discovery Methods: Manual Review

Status: Fixed

Code Location:

scripts/contracts/utils/src/utils.rs#41,57;

scripts/contracts/ickb_logic/src/utils.rs#21

Descriptions:

It seems that the LengthNotEnough error is not being handled in a specific way. While this approach might be valid in certain contexts, it could potentially allow incomplete data to be treated as valid without additional checks.

Suggestion:

This issue has already been discussed with the project team at GitHub issue.

ENT1-1 Confusion attack on Limit Order

Severity: Minor

Discovery Methods:

Status: Acknowledged

Code Location:

scripts/contracts/limit_order/src/entry.rs#1

Descriptions:

Due to the architectural design of Nervos L1, output locks are not executed during the transaction validation process. Consequently, an attacker may create a limit order that shares the same master cell as an already existing limit order. This situation may lead to confusion for the user regarding the identification of their actual limit order. Users must exercise particular caution when melting their limit order and master cell, as selecting the incorrect limit order could result in the permanent locking of the funds associated with their original limit order.

Suggestion:

Such problems can be solved from the front end by querying to track every preceding transaction for all limit order cells until the original transaction is found to see if it is in the same transaction as the master cell.This issue has already been discussed with the project team in the dedicated GitHub issue.

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

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.