DeHacker

Code Security Assessment

ZEBEC-RUST

March 11th, 2025

Contents

CONTENTS 1
SUMMARY 2
ISSUE CATEGORIES 3
OVERVIEW 4
PROJECT SUMMARY 4
VULNERABILITY SUMMARY 4
AUDIT SCOPE 5
FINDINGS 6
MAJOR 7
SRC-02 | Centralization Related Risks and Upgradability 7
DESCRIPTION 7
RECOMMENDATION 8
MEDIUM 10
MIG-01 | Missing ZBC and ZBCN Token Decimal Conversion 10
DESCRIPTION 10
RECOMMENDATION 10
MINOR 11
INS-01 | Missing ZBC and ZBCN Token Decimal Conversion 11
DESCRIPTION 11
RECOMMENDATION 13
INFORMATIONAL 14
INI-02 | Assignee And Assigning Time For zbcn_mint Mint Authority Role 14
DESCRIPTION 14
RECOMMENDATION 15
INFORMATIONAL 16
LIB-01 | Potential Front-Running Risk with Initialize Instructions 16
DESCRIPTION 16
RECOMMENDATION 16
INFORMATIONAL 17
MIG-02 | ZBCN Max Minted Amount 17
DESCRIPTION 17
RECOMMENDATION 17
INFORMATIONAL 18
MIG-03 | Meaningless Variable Name 18
DESCRIPTION 18
RECOMMENDATION 18
INFORMATIONAL 19
SRC-01 | ZBC and ZBCN Token Price 19
DESCRIPTION 19
RECOMMENDATION 20
DISCLAIMER 21
APPENDIX 22
ABOUT 23

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best prac tices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	Zebec-Rust
Platform	Rust
Website	https://zebec.io
Type	DeFi
Language	Solana
Codebase	Zebec Github Repo

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	1	0	0	1	0	0
Medium	1	0	0	0	0	1
Minor	1	0	0	0	0	1
Informational	5	0	0	1	0	4
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
INI	Zebec-protocol/migration-program	9ab06f0fecfe6ada46db679376bf9fcc75903b8133fecc41c2141e1f6be7b795

Findings

ID	Title	Severity	Status
SRC-02	Centralization Related Risks And Upgradability	Major	Acknowledged
MIG-01	Missing ZBC And ZBCN Token Decimal Conversion	Medium	Resolved
INS-01	Missing Mint Authority Validation	Minor	Resolved
INI-02	Assignee And Assigning Time For zbcn_mint Mint Authority Role	Informational	Resolved
LIB-01	Potential Front-Running Risk With Initialize Instructions	Informational	Acknowledged
MIG-02	ZBCN Max Minted Amount	Informational	Resolved
MIG-03	Meaningless Variable Name	Informational	Resolved
SRC-01	ZBC And ZBCN Token Price	Informational	Resolved

MAJOR

SRC-02 | Centralization Related Risks and Upgradability

Issue	Severity	Location	Status
Centralization	Major	Programs/migration-programs/src/instructions/migrate_token.rs: 114~115; programs/migration-programs/src/lib.rs: 25, 29	Acknowledged

Description

In program migration_program, the role admin has privilege over the following instructions:

- emergency_pause(): maliciously pause the ZBC migration service.
- update_admin(): grant admin role to malicious account.

In program migration_program, the ZBCN role mint::authority has privilege over the following instructions:

- mint_to(): mint ZBCN to any account.

Any compromise to the privileged roles may allow the hacker to take advantage of this authority. The hackers may manipulate the cross-chain transfers. Such actions could result in financial losses for both the project and its users.

Also, the Solana platform allows for the possibility of upgrading its programs, with the default upgrade authority being the entity responsible for deployment. In situations where the program has upgradability features and the account of the upgrade authority becomes compromised, there is the potential for an unauthorized and malicious update to the program.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We recommend carefully managing the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination mitigate by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, mitigate by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Recommendation

Permanent:

Renouncing the ownership or removing the function can be considered fully resolved.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

MEDIUM

MIG-01 | Missing ZBC and ZBCN Token Decimal Conversion

Issue	Severity	Location	Status
Logical Issue	Medium	programs/migration-programs/src/instructions/migrate_token.rs: 97-98, 114~115	Resolved

Description

Based on the code logic below, it's clear that the value of the amount argument provided is used for both burning ZBC and minting ZBCN. Notably, there is no code logic for decimal conversion. Consequently, the minted amount of ZBCN does not match the expected value.

migrate_token.rs#L97

```
// Burn ZBC tokens.
    burn(burn_ctx, amount)?;
```

migrate_token.rs#L114

```
// Mint equivalent ZBCN tokens that were burned.
    mint_to(mint_ctx, amount)?;
```

Recommendation

Recommend refactoring the code to ensure accurate conversion of token decimals during the migration process.

MINOR

INS-01 | Missing ZBC and ZBCN Token Decimal Conversion

Issue	Severity	Location	Status
Logical Issue	Minor	programs/migration-programs/src/instructions/init_config.rs:19~28, 32; programs/migration-programs/src/instructions/migrate_token.rs: 39~48	Resolved

Description

Based on the comment at L27, the mint_authority account has been assigned the role of mint authority for the ZBCN token. However, a current flaw in the code is the lack of a check to verify whether the zbcn_mint mint authority truly corresponds to the mint_authority PDA.

And after executing init_config() instruction, migrate_pda.zbcn_mint and migrate_pda.mint_authority cannot be updated.

Description

init_config.rs#L19

```
#[account(
        init,
        payer = admin,
        space = 8,
        seeds = [b"zbcn_mint"],
        bump
    )]
    /// CHECK: seeds has been checked
    /// This account has mint authority for ZBCN tokens.
    pub mint_authority: AccountInfo<'info>,
    /// ZBC (new) mint account.
    pub zbc_mint: Account<'info, Mint>,
    /// ZBCN (old) mint account.
    pub zbcn_mint: Account<'info, Mint>,
```

A similar issue also exists in migrate_token.rs.

migrate_token.rs#L39

```
/// ZBCN (new) mint account.
    pub zbcn_mint: Account<'info, Mint>,
    #[account(
        mut,
        seeds = [b"zbcn_mint"],
        bump
    )]
    /// CHECK: seeds has been checked
    /// This account has mint authority for ZBCN tokens.
    pub mint_authority: AccountInfo<'info>,
```

Recommendation

Recommend refactoring the code to ensure that the zbcn_mint token mint authority account is mint_authority PDA.

INFORMATIONAL

INI-02 | Assignee And Assigning Time For zbcn_mint Mint Authority Role

Issue	Severity	Location	Status
Logical Issue	Medium	programs/migration-programs/src/instructions/migrate_token.rs: 97-98, 114~115	Resolved

Description

According to the comment at L27, the mint_authority account has been designated the mint authority role for the ZBCN token. However, the current codebase lacks the necessary logic to assign the mint authority role for the ZBCN token to the mint_authority account.

```
#[account(
        init,
        payer = admin,
        space = 8,
        seeds = [b"zbcn_mint"],
        bump
    )]
    /// CHECK: seeds has been checked
    /// This account has mint authority for ZBCN tokens.
    pub mint_authority: AccountInfo<'info>,
    /// ZBC (new) mint account.
    pub zbc_mint: Account<'info, Mint>,
    /// ZBCN (old) mint account.
    pub zbcn_mint: Account<'info, Mint>,
```

Recommendation

The audit team would like to ask Zebec team regarding the design logic behind the granting of the ZBCN mint authority role.

INFORMATIONAL

LIB-01 | Potential Front-Running Risk with Initialize Instructions

Issue	Severity	Location	Status
Volatile Code	Informational	Programs/migration-programs/src/lib.rs: 17~19	Acknowledged

Description

The program migration_program can be initialized via init_config instruction and set up accounts with sensitive information for the corresponding program.

However, a malicious party can invoke init_config instructions, which will affect the program.

Recommendation

Recommend that the team review the process for deployment and initialization of programs. To make it more difficult for frontrunners to take advantage of the situation to gain ownership of contracts, the audit team recommends using extra gas during the initialization phase to make it costlier and less likely for front-running attacks to succeed(although the chance is low on Solana).

Depending on how the future implementations of the contract are structured, it would also be beneficial to restrict access to the initialization instructions to ensure that only intended users have access to the function.

INFORMATIONAL

MIG-02 | ZBCN Max Minted Amount

Issue	Severity	Location	Status
Logical Issue	Informational	programs/migration-programs/src/instructions/migrate_token.rs: 84~88	Resolved

Description

According to the logic outlined in the following code, it is evident that only 10 billion ZBCN tokens are minted via the migration-program, implying that only 10 billion ZBC tokens undergo migration.

migrate_token.rs#L84

```
// The migration program is one way ZBC -> ZBCN. The migration program will stop once 10,000,000,000 ZBCN tokens are minted.
    require!(
        migrate_pda.total_migrated + amount <= 10_000_000_000 * 10u64.pow(zbcn_decimals as u32),
        MigrationError::MaxMigrationReached
    );
```

Recommendation

The audit team would like to request confirmation from the Zebec team regarding whether the maximum supply of ZBC does not exceed 10 billion.

INFORMATIONAL

MIG-03 | Meaningless Variable Name

Issue	Severity	Location	Status
Coding Style	Informational	programs/migration-programs/src/instructions/migrate_token.rs: 103, 111	Resolved

Description

The following CPI is utilized for minting ZBCN, however, the variable name mint_zbc fails to accurately convey the meaning of MintTo struct instance.

```
let mint_zbc = MintTo {
        mint: zbcn_mint.to_account_info(),
        to: sender_zbcn_ata.to_account_info(),
        authority: mint_authority.to_account_info(),
    };

    let mint_ctx = CpiContext::new_with_signer(
        token_program.to_account_info(),
        mint_zbc,
        mint_authority_seed,
    );
```

Recommendation

Recommend using mint_zbcn to replce mint_zbc.

INFORMATIONAL

SRC-01 | ZBC and ZBCN Token Price

Issue	Severity	Location	Status
Design Issue	Informational	programs/migration-programs/src/instructions/migrate_token.rs: 97-98, 114~115; programs/migration-programs/sec/lib.rs: 21	Resolved

Description

Based on the code logic below, it's evident that the provided amount argument value is used for both burning ZBC and minting ZBCN. This implies that the conversion rate between ZBC and ZBCN tokens is 1:1 when they have the same decimal precision.

migrate_token.rs#L97

```
// Burn ZBC tokens.
    burn(burn_ctx, amount)?;
```

migrate_token.rs#L114

```
// Mint equivalent ZBCN tokens that were burned.
    mint_to(mint_ctx, amount)?;
```

The holders of ZBC token assets accesses the migrate_token() function to convert a specified amount, rather than their entire holding, of ZBC to ZBCN. Therefore, the duration of the token migration is not predetermined. Throughout the migration process, the prices of ZBC and ZBCN may differ. In such instances, a 1:1 amount conversion could potentially reduce the value of the holder's assets if the price of ZBCN is lower than that of ZBC.

Description

lib.rs#L21

```
pub fn migrate_token(ctx: Context<MigrateToken>, amount: u64) -> Result<()> {
    migrate_token::handler(ctx, amount)
}
```

Recommendation

The audit team would like to request confirmation from the Zebec team regarding whether the prices of both ZBC and ZBCN remain consistent throughout the entire token migration process.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of 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 sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege
Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style
Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code
Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue
Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum
Cosmos
Eos
Substrate

TECH STACK

Python
Solidity
Rust
c++

CONTACTS

https://dehacker.io
https://twitter.com/dehackerio
https://github.com/dehacker/audits_public
https://t.me/dehackerio
https://blog.dehacker.io/

March 11th 2025