CERTIK

Preliminary Comments

# Zebec Program v2

Nov 19th, 2021

# Table of Contents

## Summary

This report has been prepared for Zebec Protocol to discover issues and vulnerabilities in the source code of the Zebec Program v2 project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in purely informational findings. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

# Overview

## Project Summary

| | |
|---|---|
| Project Name | Zebec Program v2 |
| Platform | Solana |
| Language | Rust |
| Codebase | https://github.com/Zebec-protocol/zebec-program-v2/ |
| Commit | f64184d2160708244a5f3a151b8b09e5dd5c3e74 |

## Audit Summary

| | |
|---|---|
| Delivery Date | Nov 22, 2021 |
| Audit Methodology | Static Analysis, Manual Review |
| Key Components | |

## Vulnerability Summary

| Vulnerability Level | Total | Pending | Declined | Acknowledged | Partially Resolved | Resolved |
|---|---|---|---|---|---|---|
| ● Critical | 0 | 0 | 0 | 0 | 0 | 0 |
| ● Major | 0 | 0 | 0 | 0 | 0 | 0 |
| ● Medium | 0 | 0 | 0 | 0 | 0 | 0 |
| ● Minor | 0 | 0 | 0 | 0 | 0 | 0 |
| ● Informational | 15 | 15 | 0 | 0 | 0 | 0 |
| ● Discussion | 0 | 0 | 0 | 0 | 0 | 0 |

## Audit Scope

| ID | Repo | Commit | File | SHA256 Checksum |
|---|---|---|---|---|
| ERR | Zebec-protocol/zebec-program-v2 | f64184d | src/error.rs | d7d8feddcebd62457c16976ac38d4b497d42ef16e79630049dfd265c5d4b6ff8 |
| INS | Zebec-protocol/zebec-program-v2 | f64184d | src/instruction.rs | 4fa1de848c9788370a485bf8594cbc9f3a829e3adde645414d6ae5d4495891bd |
| LIB | Zebec-protocol/zebec-program-v2 | f64184d | src/lib.rs | ea4f4fed482154130ccaa3db687ad1749aae3f3b6c309d1986201176eb71b622 |
| PRO | Zebec-protocol/zebec-program-v2 | f64184d | src/processor.rs | 1be37ef5f7c853f85478656e710440d8be96d55d132bf2a84a83edb41798ec75 |
| STA | Zebec-protocol/zebec-program-v2 | f64184d | src/state.rs | 9540dea542ef0c66368af2fdd245f8bfa56ad6d9fbcb0ee9dd551986f20fe71d |
| UTI | Zebec-protocol/zebec-program-v2 | f64184d | src/utils.rs | 7e712ad755562af348280ea286a6fd27db2168ca8dab3a23cac7218b0d071ada |

## Findings

15
Total Issues

| | |
|---|---|
| 🟥 **Critical** | **0** (0.00%) |
| 🟧 **Major** | **0** (0.00%) |
| 🟨 **Medium** | **0** (0.00%) |
| 🟨 **Minor** | **0** (0.00%) |
| 🟦 **Informational** | **15** (100.00%) |
| 🟩 **Discussion** | **0** (0.00%) |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| PRO-01 | Code reuse can be streamlined through member function | Coding Style | ● Informational | ⓘ Pending |
| PRO-02 | Decrementation can be simplified | Mathematical Operations | ● Informational | ⓘ Pending |
| PRO-03 | Decrementation can be simplified | Mathematical Operations | ● Informational | ⓘ Pending |
| PRO-04 | Unnecessary re-slicing | Language Specific | ● Informational | ⓘ Pending |
| PRO-05 | Unnecessary re-slicing | Language Specific | ● Informational | ⓘ Pending |
| PRO-06 | Decrementation can be simplified | Mathematical Operations | ● Informational | ⓘ Pending |
| PRO-07 | Decrementation can be simplified | Mathematical Operations | ● Informational | ⓘ Pending |
| PRO-08 | Unnecessary re-slicing | Language Specific | ● Informational | ⓘ Pending |
| PRO-09 | Unnecessary error type conversion | Language Specific | ● Informational | ⓘ Pending |
| PRO-10 | Incrementation can be simplified | Mathematical Operations | ● Informational | ⓘ Pending |
| PRO-11 | Unnecessary error type conversion | Language Specific | ● Informational | ⓘ Pending |
| PRO-12 | Unnecessary re-slicing | Language Specific | ● Informational | ⓘ Pending |
| PRO-13 | Unnecessary double reference | Language Specific | ● Informational | ⓘ Pending |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| PRO-14 | Unnecessary re-slicing | Language Specific | ● Informational | ⓘ Pending |
| UTI-01 | Unnecessary explicit return | Language Specific | ● Informational | ⓘ Pending |

## PRO-01 | Code reuse can be streamlined through member function

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | ● Informational | src/processor.rs: 589, 497, 392, 248, 199, 131 | ⓘ Pending |

### Description

The linked lines in the `processor` module calculate the allowed amount based on the current timestamp within the span of the escrow using the same code:

```
let mut allowed_amt = (
    ((now - escrow.start_time) as f64) /
    ((escrow.end_time - escrow.start_time) as f64) *
    escrow.amount as f64
) as u64;
```

### Recommendation

Consider implementing the calculation as a member function under the `Escrow` struct in the `state` module in order to reduce any room for error and alleviate any potential future refactoring:

```
impl Escrow {
    pub fn allowed_amt(&self, timestamp: u64) -> u64 {
        (
            ((now - self.start_time) as f64) /
            ((self.end_time - self.start_time) as f64) *
            self.amount as f64
        ) as u64
    }
}
```

## PRO-02 | Decrementation can be simplified

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations | ● Informational | src/processor.rs: 167 | ⓘ Pending |

### Description

The `process_sol_withdraw_stream` function in the `processor` module performs a decrementation on the `escrow.withdraw_limit` by way of assignment and primitive subtraction on L167, which is unnecessary:

```
escrow.withdraw_limit = escrow.withdraw_limit-amount
```

### Recommendation

Consider replacing the assignment and primitive subtraction with a subtracting assignment on L167:

```
escrow.withdraw_limit -= amount;
```

## PRO-03 | Decrementation can be simplified

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations | ● Informational | src/processor.rs: 169 | ⓘ Pending |

### Description

The `process_sol_withdraw_stream` function in the `processor` module performs a decrementation on the `escrow.amount` by way of assignment and primitive subtraction on L169, which is unnecessary:

```
escrow.amount = escrow.amount-amount;
```

### Recommendation

Consider replacing the assignment and primitive subtraction with a subtracting assignment on L169:

```
escrow.amount -= amount;
```

## PRO-04 | Unnecessary re-slicing

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | src/processor.rs: 345 | ⓘ Pending |

### Description

The `process_token_stream` function in the `processor` module re-slices the entire `pda_signer_seeds` slice on L345, which is unnecessary:

```
&pda_signer_seeds[..]
```

### Recommendation

Since the `pda_signer_seeds` value is already a slice, consider passing it by value on L345 instead of re-slicing it for the entire range:

```
pda_signer_seeds
```

## PRO-05 | Unnecessary re-slicing

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | src/processor.rs: 455 | Pending |

### Description

The `process_token_withdraw_stream` function in the `processor` module re-slices the entire `pda_signer_seeds` slice on L455, which is unnecessary:

```
&pda_signer_seeds[..]
```

### Recommendation

Since the `pda_signer_seeds` value is already a slice, consider passing it by value on L455 instead of re-slicing it for the entire range:

```
pda_signer_seeds
```

## PRO-06 | Decrementation can be simplified

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations | ● Informational | src/processor.rs: 459 | Pending |

### Description

The `process_token_withdraw_stream` function in the `processor` module performs a decrementation on the `escrow.withdraw_limit` by way of assignment and primitive subtraction on L459, which is unnecessary:

```
escrow.withdraw_limit = escrow.withdraw_limit-amount
```

### Recommendation

Consider replacing the assignment and primitive subtraction with a subtracting assignment on L459:

```
escrow.withdraw_limit -= amount;
```

## PRO-07 | Decrementation can be simplified

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations | ● Informational | src/processor.rs: 461 | ⓘ Pending |

### Description

The `process_token_withdraw_stream` function in the `processor` module performs a decrementation on the `escrow.amount` by way of assignment and primitive subtraction on L461, which is unnecessary:

```
escrow.amount = escrow.amount-amount;
```

### Recommendation

Consider replacing the assignment and primitive subtraction with a subtracting assignment on L461:

```
escrow.amount -= amount;
```

## PRO-08 | Unnecessary re-slicing

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | src/processor.rs: 563 | ⓘ Pending |

### Description

The `process_token_cancel_stream` function in the `processor` module re-slices the entire `pda_signer_seeds` slice on L563, which is unnecessary:

```
&pda_signer_seeds[..]
```

### Recommendation

Since the `pda_signer_seeds` value is already a slice, consider passing it by value on L563 instead of re-slicing it for the entire range:

```
pda_signer_seeds
```

## PRO-09 | Unnecessary error type conversion

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | src/processor.rs: 731 | ⓘ Pending |

### Description

The `process_fund_sol` function in the `processor` module converts the error result `ProgramError::UninitializedAccount` on L731, which is unnecessary due to the error type for the `process_fund_sol` function already being `ProgramError`:

```
return Err(ProgramError::UninitializedAccount.into());
```

### Recommendation

Consider removing the `.into()` conversion on L731:

```
return Err(ProgramError::UninitializedAccount);
```

## PRO-10 | Incrementation can be simplified

| Category | Severity | Location | Status |
|---|---|---|---|
| Mathematical Operations | ● Informational | src/processor.rs: 741 | ⓘ Pending |

### Description

The `process_fund_sol` function in the `processor` module performs a incrementation on the `escrow.amount` by way of assignment and primitive addition on L741, which is unnecessary:

```
escrow.amount = escrow.amount+amount;
```

### Recommendation

Consider replacing the assignment and primitive addition with an incrementing assignment on L741:

```
escrow.amount += amount;
```

## PRO-11 | Unnecessary error type conversion

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | src/processor.rs: 752 | ⓘ Pending |

### Description

The `process_fund_token` function in the `processor` module converts the error result `ProgramError::UninitializedAccount` on L752, which is unnecessary due to the error type for the `process_fund_token` function already being `ProgramError`:

```
return Err(ProgramError::UninitializedAccount.into());
```

### Recommendation

Consider removing the `.into()` conversion on L752:

```
return Err(ProgramError::UninitializedAccount);
```

## PRO-12 | Unnecessary re-slicing

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | src/processor.rs: 796 | ⓘ Pending |

### Description

The `process_withdraw_sol` function in the `processor` module re-slices the entire `pda_signer_seeds` slice on L796, which is unnecessary:

```
&pda_signer_seeds[..]
```

### Recommendation

Since the `pda_signer_seeds` value is already a slice, consider passing it by value on L796 instead of re-slicing it for the entire range:

```
pda_signer_seeds
```

## PRO-13 | Unnecessary double reference

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | src/processor.rs: 820 | ⓘ Pending |

### Description

The `process_withdraw_token` function in the `processor` module passes `source_account_info.key` by reference on L820 to the `spl_associated_token_account::get_associated_token_address` function, which is unnecessary because the `source_account_info.key` is already a reference:

```
let source_associated_token = spl_associated_token_account::get_associated_token_address(
    &source_account_info.key,
    token_mint_info.key
);
```

### Recommendation

Since `source_account_info.key` is already a reference, consider removing the reference operator on L820:

```
let source_associated_token = spl_associated_token_account::get_associated_token_address(
    source_account_info.key,
    token_mint_info.key
);
```

## PRO-14 | Unnecessary re-slicing

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | src/processor.rs: 843 | ⓘ Pending |

### Description

The `process_withdraw_token` function in the `processor` module re-slices the entire `pda_signer_seeds` slice on L843, which is unnecessary:

```
&pda_signer_seeds[..]
```

### Recommendation

Since the `pda_signer_seeds` value is already a slice, consider passing it by value on L843 instead of re-slicing it for the entire range:

```
pda_signer_seeds
```

## UTI-01 | Unnecessary explicit return

| Category | Severity | Location | Status |
|---|---|---|---|
| Language Specific | ● Informational | src/utils.rs: 24 | ⓘ Pending |

### Description

The `assert_keys_equal` function in the the `utils` module performs an explicit return on L24, which is unnecessary due to the function having no further statements to execute.

### Recommendation

Consider removing the explicit return on L24:

```
pub fn assert_keys_equal(key1: Pubkey, key2: Pubkey) -> ProgramResult {
    if key1 != key2 {
        Err(TokenError::PublicKeyMismatch.into())
    } else {
        Ok(())
    }
}
```

# Appendix

## Finding Categories

### Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

### Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

### Coding Style

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

## 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.

## Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

## About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK