HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Civic
**Date**: 28 Jun 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io



## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for Civic |
| **Approved By** | Yevheniy Bezuhlyi \| SC Audits Head at Hacken OÜ |
| **Type** | Digital Identity Platform |
| **Platform** | Solana |
| **Language** | Rust |
| **Methodology** | Link |
| **Website** | civic.com |
| **Changelog** | 19.04.2023 - Initial Review<br>19.05.2023 - Second Review<br>28.06.2023 - Third Review |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Table of contents

**Introduction** **4**
**Scope** **4**
**Severity Definitions** **7**
**Executive Summary** **8**
**Risks** **9**
**System Overview** **10**
**Checked Items** **12**
**Findings** **14**
Critical 14
High 14
H01. Denial Of Service State 14
Medium 15
M01. Improper Account Funding 15
Low 15
L01. Unfinalized Code 15
L02. Confusing Code 15
L03. Redundant Code 15
L04. Outdated Dependencies 17
L05. Floating Language Version 18
L06. Best Practices Violation 18
L07. Best Practices Violation 19
L08. Best Practices Violation 19
L09. Unsafe Rust Code 20
L10. Missing Documentation 20
L11. Unfinalized Code 21
L12. Confusing Code 21
**Disclaimers** **22**

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Introduction

Hacken OÜ (Consultant) was contracted by Civic (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

## Scope

The scope of the project includes review and security analysis of the following smart contracts from the provided repository:

### Initial review scope

| | |
|---|---|
| **Repository** | github.com/identity-com/on-chain-identity-gateway |
| **Commit** | c939b6feb8aa92d596306a1aeb2dc497c2f7f693 |
| **Whitepaper** | Link |
| **Functional Requirements** | General Overview<br>Integration Example<br>Program Functions Description |
| **Technical Requirements** | Are not provided |
| **Contracts** | File: ./solana/integration-lib/src/borsh.rs<br>SHA3: 6cad789905fe29079c66311e9dcc18a8332f354b6d53440cbbea16ff8810d824<br><br>File: ./solana/integration-lib/src/error.rs<br>SHA3: 29bc298d1924311ad8e959023b529e814b0d71438505525bad2d07fec18fbd49<br><br>File: ./solana/integration-lib/src/instruction.rs<br>SHA3: 3a621d0042e8d0a69be8f6c1ccf36f8ae165e5492358e0e821cc426b0746f0b9<br><br>File: ./solana/integration-lib/src/lib.rs<br>SHA3: c6f27eea7d07010ed647d32a29d7b1f8b49134ec4116e0f3aa87604777fa25ab<br><br>File: ./solana/integration-lib/src/networks.rs<br>SHA3: c0420db1bed69bd3b94dfef28f1a72d4528cf9556e20025c478f812abd942336<br><br>File: ./solana/integration-lib/src/state.rs<br>SHA3: e951c41932455f0f82f610fe9da9ac5a5693e2cbdad9d6a8ae29a0222717cffe<br><br>File: ./solana/program/src/entrypoint.rs<br>SHA3: be82823540d5afb969684a314d2251f6638941f951e434c73656e9c885ee8fe4<br><br>File: ./solana/program/src/error.rs<br>SHA3: 5ae952c07ba22395bb9b6e4ec89cb4e535bea369b5a4d977c72787d3a5bec8f5<br><br>File: ./solana/program/src/lib.rs<br>SHA3: 78472a8f0b94caabbe97333845dfa72836790d1e55a3697d071eef9c65d58c0d<br><br>File: ./solana/program/src/processor.rs<br>SHA3: b9f4fd75c36c6d12117facd4dc767983c5862fad94422c6a6e4424d372215156<br><br>File: ./solana/program/src/state.rs<br>SHA3: 79dc5176667d50af1527d80d71379b4edb13769ac642f7208606941c07c3c325 |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Second review scope

| | |
|---|---|
| **Repository** | github.com/identity-com/on-chain-identity-gateway |
| **Commit** | c181e2db70c1f7b41d88f67f60dbed1fac8c5143 |
| **Whitepaper** | Link |
| **Functional Requirements** | General Overview |
| **Technical Requirements** | Development Docs |
| **Contracts** | File: ./solana/program/src/borsh.rs<br>SHA3: f11511c83fe0fcd14564277a9e9b31b0c66688b3b7f50913faa1e112562bf506<br><br>File: ./solana/program/src/entrypoint.rs<br>SHA3: f6dcb0b1eca536cead7f6da545f5b4aefaa9a8aeb5c047dbed8e12ecbdfd4b5c<br><br>File: ./solana/program/src/error.rs<br>SHA3: 213fc54e55175b4dc113ecfa82be7820b22270b6287af76a869485a374a525ff<br><br>File: ./solana/program/src/instruction.rs<br>SHA3: 75aec7f5644d3dc656c920d2f39817707a52ff5b76efe6321306faa6e1cde653<br><br>File: ./solana/program/src/lib.rs<br>SHA3: c071ec3261c921f8d96e7cda025e8140706dc7a24026267a6716b7d0c403ce36<br><br>File: ./solana/program/src/networks.rs<br>SHA3: 5918fd006bb54a581502df2d863cda6575747a7450869fc0b5ab8f5df7124479<br><br>File: ./solana/program/src/processor.rs<br>SHA3: 424f77be568217d186558c8663f7bd86f48b2014596ddf383586ccacc3beedc1<br><br>File: ./solana/program/src/state.rs<br>SHA3: 812162bdfe16149523411d04f17b7ce3b4a1bac2acb79d6ae39109c295ce758b |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Third review scope

| | |
|---|---|
| **Repository** | github.com/identity-com/on-chain-identity-gateway |
| **Commit** | d94bfee1a35b533583efc1b2151a9224b1a4b305 |
| **Whitepaper** | Link |
| **Functional Requirements** | General Overview |
| **Technical Requirements** | Development Docs |
| **Contracts** | File: ./solana/program/src/borsh.rs<br>SHA3: f11511c83fe0fcd14564277a9e9b31b0c66688b3b7f50913faa1e112562bf506<br><br>File: ./solana/program/src/entrypoint.rs<br>SHA3: f6dcb0b1eca536cead7f6da545f5b4aefaa9a8aeb5c047dbed8e12ecbdfd4b5c<br><br>File: ./solana/program/src/error.rs<br>SHA3: 213fc54e55175b4dc113ecfa82be7820b22270b6287af76a869485a374a525ff<br><br>File: ./solana/program/src/instruction.rs<br>SHA3: 75aec7f5644d3dc656c920d2f39817707a52ff5b76efe6321306faa6e1cde653<br><br>File: ./solana/program/src/lib.rs<br>SHA3: c071ec3261c921f8d96e7cda025e8140706dc7a24026267a6716b7d0c403ce36<br><br>File: ./solana/program/src/networks.rs<br>SHA3: 5918fd006bb54a581502df2d863cda6575747a7450869fc0b5ab8f5df7124479<br><br>File: ./solana/program/src/processor.rs<br>SHA3: b26c3abb47a62531ae2ae2547757cfcc4b3e3cb0b3f098953c91a650970e3717<br><br>File: ./solana/program/src/state.rs<br>SHA3: 812162bdfe16149523411d04f17b7ce3b4a1bac2acb79d6ae39109c295ce758b |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors. |
| High | High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors. |
| Medium | Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category. |
| Low | Low vulnerabilities are related to outdated and unused code or minor Gas optimization. These issues won't have a significant impact on code execution but affect code quality. |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

## Documentation quality

The total Documentation Quality score is **9** out of **10**.

- README.md in the *program* crate as well as doc comments in *program::instruction* state the need to pass the rent sysvar account to some instructions, but actually the instructions do not expect it.

## Code quality

The total Code Quality score is **9** out of **10**.

- There are minor cases of unfinalized or confusing code.
- There are hardcoded generated values whose derivation is not validated properly.

See the Findings section for detailed issue descriptions.

## Test coverage

Code coverage of the project is **91%**.

- There is both positive and negative cases coverage.
- All kinds of actors are tested.
- *program::processor::remove_feature_from_network* is not tested.

## Security score

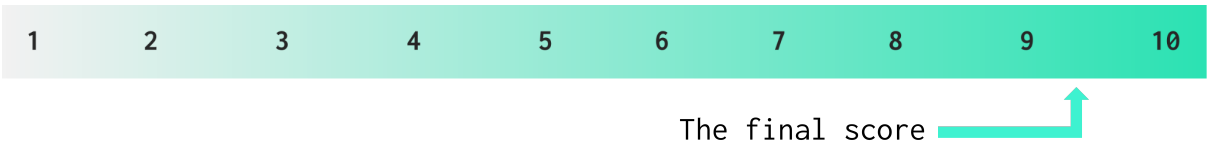
As a result of the audit, the code contains **3** low severity issues. The security score is **10** out of **10**.

All found issues are displayed in the Findings section of the report.

## Summary

According to the assessment, the Customer's smart contract has the following score: **9.38**.

The system users should acknowledge all the risks summed up in the Risks section of the report.



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

*Table. The distribution of issues during the audit*

| Review date | Low | Medium | High | Critical |
|---|---|---|---|---|
| 19 April 2023 | 10 | 1 | 1 | 0 |
| 19 May 2023 | 3 | 0 | 0 | 0 |
| 28 June 2023 | 3 | 0 | 0 | 0 |

## Risks

- Generally, in Solana, a program may be deployed as mutable, which could be used to change the implementation in an unexpected way; additionally, insufficient funding of the program-containing account may lead to the program going down.
- The gatekeeper that issued a token is able to freeze it at any moment.
- Any gatekeeper of the network that issued a token is able to revoke it, remove it, or render it expired at any moment.
- A gatekeeper network may remove a gatekeeper at any moment.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## System Overview

*On-chain Identity Gateway* is a platform that implements auth token creation and management. The main purpose of the system is to allow other smart contracts to validate the user's identity (for example, KYC verification, reCAPTCHA, etc.).

The domain model has the following key entities: "gatekeeper network", "gatekeeper", and "gateway token". A gatekeeper network can add/remove gatekeepers to itself. Gatekeepers can create gateway tokens within their network for arbitrary parties. A gateway token represents a credential that is meant to be used by client systems to authenticate their users. A party may be granted many gateway tokens at the same time, including many tokens from the same network. A gateway token may have an expiration time, which can be increased or decreased arbitrarily by any gatekeeper in the network. A gateway token may be paused/unpaused (only by the issuing gatekeeper), revoked or removed by any gatekeeper in the network. A network may add/remove features to itself. Currently, the only feature is self-expiration, which allows a grantee of a gateway token to make the token expire immediately.

The platform supports several blockchains. The platform implementation designed for the Solana blockchain is in the audit scope.

In-scope files:

- ./solana/program/ (also referred to as the *program* crate) – the folder contains a Rust crate that defines the operations that gatekeepers can perform on the Solana blockchain, and the client-side code for interacting with the program.
  - ./solana/program/src/entrypoint.rs – the file contains the program entrypoint and performs a redirect to the processor.
  - ./solana/program/src/lib.rs – the file contains module declarations, the program ID declaration, and reading/validation utilities for gateway tokens.
  - ./solana/program/src/processor.rs – the file contains the implementation of fundamental operations over the domain entities.
  - ./solana/program/src/state.rs – the file contains the program state data structures definitions and helper functions to work with the state.
  - ./solana/program/src/borsh.rs – the file contains Borsh helpers to work with data slices.
  - ./solana/program/src/error.rs – the file contains the protocol error declarations.

- ./solana/program/src/instruction.rs – the file contains the program instruction signatures and the functions constructing calls into the respective program APIs.
- ./solana/program/src/networks.rs – the file contains official gateway network addresses.

## Privileged roles

- The owner of the account that contains the program - as allowed in Solana - can modify the account, including the program code, if it was not deployed as immutable.
- Within the program, there are no universal high-privileged roles. Each gateway network is a root of an isolated graph of entities. For each graph, the ultimate-privilege entity is the gateway network, which can spawn many gatekeepers that have second-class privileges. The details of the abilities of the privileged entities are described in the main body of the System Overview.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

| Item | Description | Status |
|---|---|---|
| **Integer Overflow and Underflow** | If unchecked math is used, all math operations should be safe from overflows and underflows. | Passed |
| **Unchecked Call Return Value** | The return value of a message call should be checked. | Passed |
| **Access Control & Authorization** | Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users. | Passed |
| **Assert Violation** | Properly functioning code should never reach a failing assert statement. | Passed |
| **Deprecated Rust Functions** | Deprecated built-in functions should never be used. | Passed |
| **DoS (Denial of Service)** | Execution of the code should never be blocked by a specific contract state unless required. | Passed |
| **Block values as a proxy for time** | Block numbers should not be used for time calculations. | Not Relevant |
| **Signature Unique Id** | Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. | Not Relevant |
| **Weak Sources of Randomness** | Random values should never be generated from Chain Attributes or be predictable. | Not Relevant |
| **Race Conditions** | Race Conditions and Transactions Order Dependency should not be possible. | Passed |
| **Calls Only to Trusted Addresses** | All external calls should be performed only to trusted addresses. | Passed |
| **Presence of Unused Variables** | The code should not contain unused variables if this is not justified by design. | Passed |
| **Assets Integrity** | Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract. | Passed |
| **User Balances Manipulation** | Contract owners or any other third party should not be able to access funds belonging to users. | Passed |

| | | |
|---|---|---|
| **Data Consistency** | Smart contract data should be consistent all over the data flow. | Passed |
| **Flashloan Attack** | When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used. | Not Relevant |
| **Token Supply Manipulation** | Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the Customer. | Passed |
| **Gas and Loops** | Transaction execution costs should not depend dramatically on the amount of data stored on the contract. | Passed |
| **Compiler Warnings** | The code should not force the compiler to throw warnings. | Passed |
| **Requirements Compliance** | The code should be compliant with the requirements provided by the Customer. | Passed |
| **Environment Consistency** | The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code. | Passed |
| **Secure Oracles Usage** | The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles. | Not Relevant |
| **Tests Coverage Above 90%** | The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. The usage of contracts by multiple users should be tested. | Passed |
| **Stable Imports** | The code should not reference draft contracts, that may be changed in the future. | Passed |
| **Unsafe Rust code** | The Rust type system does not check the memory safety of unsafe Rust code. Thus, if a smart contract contains any unsafe Rust code, it may still suffer from memory corruptions such as buffer overflows, use after frees, uninitialized memory, etc. | Passed |
| **Improper account funding** | All Solana accounts holding an Account, Mint, or Multisig must contain enough SOL to be considered rent exempt. Otherwise, the accounts may fail to load. | Passed |
| **Missing freeze authority checks** | When freezing is enabled but the program does not verify that the freezing account call has been signed by the appropriate freeze_authority. | Not Relevant |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Findings

## Critical

No critical severity issues were found.

## High

### H01. Denial Of Service State

Note: this could be a false positive - however, this was not possible to confirm from the functional requirements.

*program::processor::issue_vanilla* allocates the size for a new token account as equal to the size (of Borsh encoding) of the newly created *GatewayToken* instance.

Since *GatewayToken* contains several *Optional*-typed fields (*parent_gateway_token*, *owner_identity*, *expire_time*), the encoding size of an instance may vary depending on whether some of the optionals are non-*None*. This is because *None* is encoded compactly in Borsh i.e. a field encoding is small if the value is *None*, and may be larger if the value is different than *None*.

In particular, *program::processor::issue_vanilla* does not set *expire_time* if it was not given to the method. The current space allocation approach leads to the impossibility of setting the field later.

If *expire_time* is not given to the method, the *GatewayToken* instance initial size will be 101. If later for that token *program::processor::update_expiry* is called, the token size will become 109, and the execution will fail to write it to the account. In other words, if a token is issued without expiration time, the expiration time cannot be set later. Note that this also blocks *program::processor::expire_token* for such a token, because the function works by setting an expiration time in the past.

**Path:** program::processor::issue_vanilla

**Recommendation**: Take the variable encoding size of *Optional*-typed fields into consideration or explicitly document the relevant effects in the current code.

**Found in:** c939b6f

**Status**: Fixed (Revised commit: c181e2d)

## Medium

### M01. Improper Account Funding

The usage of *Rent::default()* may cause insufficient or excessive funding of a newly created account depending on the underlying blockchain configuration/state. In particular, insufficient funding may lead to an unexpected purging of the account.

**Path:** program::processor::add_feature_to_network

**Recommendation**: Read *Rent* as a sysvar instead of default-constructing.

**Found in:** c939b6f

**Status**: Fixed (Revised commit: c181e2d)

## Low

### L01. Unfinalized Code

- *integration_lib::state*:
  - TODO at line 152
  - Commented-out code at lines 123-124
  - Commented-out code at lines 546-550

**Path:** In the description

**Recommendation**: Eliminate the mentioned signs of unfinalized code.

**Found in:** c939b6f

**Status**: Fixed (Revised commit: c181e2d)

### L02. Confusing Code

- *integration_lib::state::verify_gatekeeper*:
  - The parameter name *gatekeeper* could mean “gatekeeper authority” or “gatekeeper account”
- *integration_lib::state::get_gateway_token_address_with_seed*:
  - The parameter name *authority* is misleading, since it actually represents a grantee, whereas “authority” is something that grants.

**Path:** In the description

**Recommendation**: Eliminate the mentioned confusion.

**Found in:** c939b6f

**Status**: Fixed (Revised commit: c181e2d)

### L03. Redundant Code

- *program::processor::GATEKEEPER_ACCOUNT_LENGTH*:

    - The constant represents 0, and it is used only in cases that have no direct relation to the gatekeeper account concept. Therefore, it is better to eliminate it by replacing it with a literal 0, which would also eliminate the confusion due to its name.
- *program::processor::add_gatekeeper*:
    - The check at line 99 is redundant, because *data_len* is always 0 regardless of whether the account is initialized, and because the subsequent *create_account* call ensures that an account has not been created.
    - The *funder_info.is_signer* check is redundant, since it will be done in the system program.
- *program::processor::issue_vanilla*:
    - The check at line 198 is redundant because *size* cannot be 0
    - The *funder_info.is_signer* check is redundant, since it will be done in the system program.
- *program::processor::set_state*:
    - The check at line 238 is redundant because the other validations cover it.
- *program::processor::update_expiry*:
    - The check at lines 291-296 is redundant because the other validations cover it.
- *program::processor::expire_token*:
    - The check at line 366 is redundant because the other validations cover it.
- *program::processor::add_feature_to_network*:
    - The *funder_account.is_signer* check is redundant since it will be done in the system program.
- *program::processor::verify_token_length*:
    - The function is redundant because all its usages are redundant (listed above).
- *program::error*:
    - The module is not used.
- *program::state*:
    - Redundant abstraction via the *Transitionable* trait, since there is only one implementation.
- *program* (lib.rs):
    - The program id string at line 14 is duplicated: it is also defined at *integration_lib::Gateway::program_id.*
- *integration_lib::error::GatewayError*
    - *InvalidGatekeeperAccount* is unused.
- *integration_lib::state::Feature*:
    - Only *Expirable* is actually used in the contract.
- *integration_lib::state::GatewayToken*:
    - *parent_gateway_token* is effectively unused because it is always *None* in the contract.
    - *owner_identity* is unused in the contract.

    - The *features* bitfield is excessive, since the only feature is *Expirable*, and its presence is equivalent to *expire_time.is_some()*.
- *integration_lib::state*:
    - *CompatibleTransactionDetails* is unused.
    - *SimpleTransactionDetails* is unused.
- *integration_lib::borsh::try_from_slice_incomplete*:
    - The local variable *data_mut* is redundant because the *data* parameter could be declared as *mut* initially.
- *integration_lib::state::GatewayTokenFunctions::hasFeature*:
    - The *if* check is redundant since there is a compile-time guarantee that the check will always succeed. Moreover, if the check fails, then the function should panic instead of returning *false* - which is a valid result of this function, thus the fact of a critical error is hidden and ignored; if the function panics in such a case, then everything depending on it will always panic, making a significant part of the system functionality always unavailable. A good way to do such checks - is a compile-time assertion; there are libraries for that. However, any test that touches this function directly or indirectly would reveal the programmer's mistake (of adding too many variants to the enum), if the function panicked - this would be an adequate substitute for a compile-time assertion.

**Path:** In the description

**Recommendation**: Eliminate the mentioned redundancies.

**Found in:** c939b6f

**Status**: Fixed (Revised commit: c181e2d)

### L04. Outdated Dependencies

Some dependencies are significantly outdated, which may cause missing important improvements or fixes.

- solana-*: current is 1.9.29, latest is 1.15.2
- borsh: current is 0.9.1, latest is 0.10.3
- bitflags: current is 1.3.2, latest is 2.1.0
- sol-did: current is 0.2.0, latest is 3.3.0

**Path:**

- program::Cargo.toml
- integration_lib::Cargo.toml

**Recommendation**: Update the dependencies as long as it does not cause a major conflict with the current implementation.

**Found in:** c939b6f

**Status**: Fixed (Revised commit: c181e2d)

### L05. Floating Language Version

It is preferable for a production project, especially a smart contract, to have the programming language version pinned explicitly. This results in a stable build output, and guards against unexpected toolchain differences or bugs present in older versions, which could be used to build the project.

The language version could be pinned in automation/CI scripts, as well as proclaimed in README or other kinds of developer documentation. However, in the Rust ecosystem, it can be achieved more ergonomically via a *rust-toolchain.toml* descriptor (see https://rust-lang.github.io/rustup/overrides.html#the-toolchain-file)

**Paths:** ./solana/rust-toolchain.toml

**Recommendation:** Pin the language version at the project level.

**Found in:** c939b6f

**Status:** Fixed (Revised commit: c181e2d)

### L06. Best Practices Violation

*integration_lib::state::InPlaceGatewayToken* significantly increases the obscurity, complexity, and rigidity of the codebase in exchange for a limited performance gain.

The goal of this type is to provide read/write access to the data fields without wholly decoding/encoding it into a conventional data struct *integration_lib::state::GatewayToken* by utilizing the random-access-ability of Borsh encoding.

This does reduce the run time of a decode-modify-encode workload by 60-75%. However, the only usage of *InPlaceGatewayToken* in the contract code (at *program::processor::expire_token*) leads to a reduction by ~19% of compute units.

The price of performance improvement is:

- *InPlaceGatewayToken* is a nontrivial piece of code that takes an effort to verify. Understanding its purpose is less problematic, although it may appear challenging because of how the code looks.
- It causes a significant increase in the number of lines of code. In particular, there is duplication of access patterns for every field.
- It increases efforts when adding/removing/reordering *GatewayToken* fields or changing their types, especially for developers other than the author of the code.
- It uses *unsafe* (see **L09**).
- The test code for it inherits the complexity and appearance issues.

It is worth noting that the rest of the code is clear and simple - in sharp contrast to *InPlaceGatewayToken*.

Full transparency is one of the main traits that distinguish smart contracts from normal programs: their code, state, and the way they are executed are meant to be public. This yields a development/usage culture in which it is important that the code is as comprehensible as possible.

**Path:** integration_lib::state::InPlaceGatewayToken

**Recommendation**:

Use *GatewayToken* instead of *InPlaceGatewayToken*.

If the performance in this case is crucial, there is an alternative way to achieve a faster and cleaner solution, given that an upgrade of data schema is possible (i.e. if versioning is implemented). It is possible to use a raw Rust struct representation (see Rust layout explainer, also keep in mind the code compilation target) as the binary format for the Solana account data. In *GatewayToken*, fields that are *Optional<T>* could be replaced with separate *is_present: bool* and *value: T* fields to achieve a fixed layout of *GatewayToken*. The idea is to use *transmute* to cast between raw bytes and *&GatewayToken* or *&mut GatewayToken*.

**Found in:** c939b6f

**Status**: Fixed (Revised commit: c181e2d)

### L07. Best Practices Violation

The *GATEWAY_NETWORKS* array contains several networks whose addresses are hardcoded.

It is considered best practice to avoid hardcoding generated values or provide corresponding tests to validate the data.

**Path:** integration_lib::networks::(IGNITE, TIBER, TEST_TIBER)

**Recommendation**: Make the networks' addresses and bump seeds run-time calculated or provide corresponding tests to check that the values are derived correctly.

**Found in:** c939b6f

**Status**: Reported

### L08. Best Practices Violation

The low-level crate *program* depends on a higher-level crate *integration_lib*. This makes the code dependency structure not aligned with the architecture.

*integration_lib* is at a higher level of abstraction than *program*, because the purpose of *integration_lib* (as the name and much of its code suggest) is to provide client-side means for working with the program.

**Path:**

- integration_lib::state
- program::state

**Recommendation**: Move the parts of *integration_lib* needed by *program* to a separate interface-crate or move them to *program*.

**Found in:** c939b6f

**Status**: Fixed (Revised commit: c181e2d)

### L09. Unsafe Rust Code

*unsafe* usage in situations where it can be avoided is widely recognized as an antipattern. It may be justified in a dedicated library code, given it is thoroughly tested and there is no equivalent solution in the standard library.

In smart contracts especially, it may harm the credibility of the codebase even when it is done flawlessly.

**Path:**

- integration_lib::state::pubkey_ref_from_array
- integration_lib::state::pubkey_mut_ref_from_array

**Recommendation**: Avoid *unsafe* in the contract code.

**Found in:** c939b6f

**Status**: Fixed (Revised commit: c181e2d)

### L10. Missing Documentation

*program::processor::issue_vanilla* allows specifying the *expire_time* token parameter. However, it is not checked that the gatekeeper network supports the expiration feature (symbolized by *NetworkFeature::UserTokenExpiry*).

Currently, the effect of *integration_lib::instruction::NetworkFeature::UserTokenExpiry* is that a grantee of the token may make the token expired via *program::processor::expire_token*. However, the name of the feature may suggest that this is a feature flag for the availability of *integration_lib::state::GatewayToken::expire_time* as a whole.

This may lead to wrong assumptions on the token life cycle.

**Path:** ./solana/

**Recommendation**: Clarify the flow of features enabling/disabling, code boundary cases by the documentation.

**Found in:** c939b6f

**Status**: Fixed (Revised commit: c181e2d)

### L11. Unfinalized Code

- *program::state*:
  - TODO at line 15

**Path:** In the description

**Recommendation**: Eliminate the mentioned signs of unfinalized code.

**Found in:** c181e2d

**Status**: New

### L12. Confusing Code

The parameter name *gatekeeper* could mean “gatekeeper authority” or “gatekeeper account”.

**Path:** program::state::verify_gatekeeper_address_and_account

**Recommendation**: Rename the parameter to *gatekeeper_authority*.

**Found in:** c181e2d

**Status**: New

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Disclaimers

## Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

## Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.