BEOSIN
Web3 Security & Compliance

# unizen-solana-swap

Smart Contract Security Audit

No. 202502061212

Feb 06th, 2025

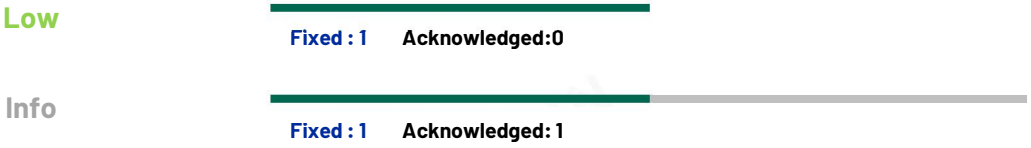
SECURING BLOCKCHAIN ECOSYSTEM

WWW.BEOSIN.COM

# Contents

# Summary of Audit Results

After auditing,1 Low and 2 Info items were identified in the unizen-solana-swap project. Specific audit details will be presented in the Findings section. Users should pay attention to the following aspects when interacting with this project:



- **Project Description:**

The project under this audit is a token exchange platform specifically designed for the Solana blockchain, aimed at providing users with convenient and secure token exchange services. Through this platform, the project facilitates different types of token conversions, mainly including the following three types of exchanges:

**SOL to SPL Token Exchange:**

Users deposit SOL into the platform's account, where the program automatically converts SOL into Wrapped SOL (WSOL), as SOL itself cannot be directly used as an SPL token on Solana.

Subsequently, WSOL is converted into the desired target SPL token via the integrated Jupiter exchange service. This conversion process involves not only the wrapping of SOL but also the complex logic of cross-token exchange.

**SPL Token to SOL Exchange:**

Users' SPL tokens are first exchanged for WSOL through Jupiter. Since WSOL is a form of SPL token, the program creates a temporary SPL token account to hold this WSOL.

This temporary account is then closed (close account), which "unwraps" WSOL back into SOL. The process of closing the account sends the SOL directly to the user's specified address. This operation requires precise account management to ensure fund security and liquidity.

**SPL Token to SPL Token Exchange:**

This type of exchange is relatively straightforward because it does not involve wrapping or unwrapping SOL. The source SPL tokens provided by the user are directly exchanged for the target SPL token through the Jupiter platform.

Here, the project acts mainly as an intermediary, ensuring a smooth exchange process and reducing the complexity for users dealing directly with trading interfaces.

**Fee Mechanism:**

All types of exchange transactions incur a fee, taken as a percentage of the source token. These fees are used for project operations, maintenance, and further development. For SOL to SPL or SPL to SOL exchanges, the fee is deducted from the respective SOL or SPL token; for SPL to SPL exchanges, the fee is subtracted from the source token before the exchange.

# 1 Overview

## 1.1 Project Overview

| | |
|---|---|
| **Project Name** | unizen-solana-swap |
| **Project Language** | Rust |
| **Platform** | Solana |
| **Github Link** | https://github.com/unizen-io/unizen-solana-swap |
| **Commit** | 653803a62fbe5c9ebb96c52d7cbeb4afec3eba10<br>a4592e0418b6a5afe26138822ae358e4c2c2b299 |

## 1.2 Audit Overview

Audit work duration: Jan 20, 2025 – Feb 06, 2025

Audit team: Beosin Security Team

## 1.3 Audit Method

The audit methods are as follows:

1. Formal Verification

Formal verification is a technique that uses property-based approaches for testing and verification. Property specifications define a set of rules using Beosin's library of security expert rules. These rules call into the contracts under analysis and make various assertions about their behavior. The rules of the specification play a crucial role in the analysis. If the rule is violated, a concrete test case is provided to demonstrate the violation.

2. Manual Review

Using manual auditing methods, the code is read line by line to identify potential security issues. This ensures that the contract's execution logic aligns with the client's specifications and intentions, thereby safeguarding the accuracy of the contract's business logic.

The manual audit is divided into three groups to cover the entire auditing process:

The Basic Testing Group is primarily responsible for interpreting the project's code and conducting comprehensive functional testing.

The Simulated Attack Group is responsible for analyzing the audited project based on the collected historical audit vulnerability database and security incident attack models. They identify potential attack vectors and collaborate with the Basic Testing Group to conduct simulated attack tests.

The Expert Analysis Group is responsible for analyzing the overall project design, interactions with third parties, and security risks in the on-chain operational environment. They also conduct a review of the entire audit findings.

3. Static Analysis

Static analysis is a method of examining code during compilation or static analysis to detect issues. Beosin-VaaS can detect more than 100 common smart contract vulnerabilities through static analysis, such as reentrancy and block parameter dependency. It allows early and efficient discovery of problems to improve code quality and security.

# 2 Findings

| Index | Risk description | Severity level | Status |
| --- | --- | --- | --- |
| unizen-solana-swap-01 | Some Accounts Lack the mut Modifier | Low | Fixed |
| unizen-solana-swap-02 | Lack of Token Verification | Info | Acknowledged |
| unizen-solana-swap-03 | Missing Event Triggers | Info | Fixed |

# Finding Details:

## [unizen-solana-swap-01] Some Accounts Lack the mut Modifier

| | |
|---|---|
| **Severity Level** | Low |
| **Lines** | src\instructions\ swap_sol_for_tokens#L70<br>src\instructions\ swap_sol_for_tokens#L33 |
| **Type** | General Vulnerability |
| **Description** | In functions like `swap_sol_for_tokens` and `take_integrator_fee`, the `integrator_wsol_ata` and `user_ata` accounts are missing the mut modifier. However, there are changes in token balances, which can cause transaction transfers to fail when executed.<br>`#[account(`<br>`mut,`<br>`associated_token::mint = sol_mint,`<br>`associated_token::authority = UNIZEN`<br>`)]`<br>`pub unizen_wsol_ata: Account<'info, TokenAccount>,`<br>`pub integrator_wsol_ata: Account<'info, TokenAccount>,` |
| **Recommendation** | It is recommended to add the mut modifier to all accounts where data changes occur. |
| **Status** | **Fixed.** The project party added the mut identifier of the corresponding account. |

## [unizen-solana-swap-02] Lack of Token Verification

| | |
|---|---|
| **Severity Level** | Info |
| **Lines** | src\instructions\swap_sol_for_tokens.rs#L34-41 |
| **Type** | Business Security |
| **Description** | In files like swap_sol_for_tokens, ATA accounts such as `integrator_src_ata` and `receiver_dst_ata` do not verify the token mint. If the token in the passed token account does not match the expected `src_token` or `dst_token`, it might lead to transaction failure or unexpected execution results. |
| **Recommendation** | It is recommended to ensure that all ATA accounts involved with tokens undergo this verification to avoid potential transaction errors. |
| **Status** | **Acknowledged.** The project party do not add token mint to keep the account list short. Transfer token would fail when destination ata has a mismatched mint address with source ata. |

```
#[account(
    mut,
    associated_token::mint = token,
    associated_token::authority = UNIZEN
)]
pub unizen_ata: Account<'info, TokenAccount>,
#[account(mut)]
pub integrator_ata: Account<'info, TokenAccount>,
```

## [unizen-solana-swap-03] Missing Event Triggers

| | |
|---|---|
| **Severity Level** | Info |
| **Type** | Coding Conventions |
| **Description** | Several critical functions in the project, such as account creation and token swapping, lack corresponding event triggers, which will increase the difficulty of project maintenance in the future. |
| **Recommendation** | It is recommended to ensure that event triggers are added for all significant operations. This not only aids in logging and debugging but also enhances the contract's transparency and traceability. |
| **Status** | **Fixed.** |

# 3 Appendix

## 3.1 Vulnerability Assessment Metrics and Status in Smart Contracts

### 3.1.1 Metrics

In order to objectively assess the severity level of vulnerabilities in blockchain systems, this report provides detailed assessment metrics for security vulnerabilities in smart contracts with reference to CVSS 3.1(Common Vulnerability Scoring System Ver 3.1).

According to the severity level of vulnerability, the vulnerabilities are classified into four levels: "critical", "high", "medium" and "low". It mainly relies on the degree of impact and likelihood of exploitation of the vulnerability, supplemented by other comprehensive factors to determine of the severity level.

| Likelihood \ Impact | Severe | High | Medium | Low |
|---|---|---|---|---|
| Probable | Critical | High | Medium | Low |
| Possible | High | Medium | Medium | Low |
| Unlikely | Medium | Medium | Low | Info |
| Rare | Low | Low | Info | Info |

### 3.1.2 Degree of impact

- **Critical**

Critical impact generally refers to the vulnerability can have a serious impact on the confidentiality, integrity, availability of smart contracts or their economic model, which can cause substantial economic losses to the contract business system, large-scale data disruption, loss of authority management, failure of key functions, loss of credibility, or indirectly affect the operation of other smart contracts associated with it and cause substantial losses, as well as other severe and mostly irreversible harm.

- **High**

High impact generally refers to the vulnerability can have a relatively serious impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a greater economic loss, local functional unavailability, loss of credibility and other impact to the contract business system.

- **Medium**

Medium impact generally refers to the vulnerability can have a relatively minor impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a small amount of economic loss to the contract business system, individual business unavailability and other impact.

- **Low**

Low impact generally refers to the vulnerability can have a minor impact on the smart contract, which can pose certain security threat to the contract business system and needs to be improved.

### 3.1.3 Likelihood of Exploitation

- **Probable**

Probable likelihood generally means that the cost required to exploit the vulnerability is low, with no special exploitation threshold, and the vulnerability can be triggered consistently.

- **Possible**

Possible likelihood generally means that exploiting such vulnerability requires a certain cost, or there are certain conditions for exploitation, and the vulnerability is not easily and consistently triggered.

- **Unlikely**

Unlikely likelihood generally means that the vulnerability requires a high cost, or the exploitation conditions are very demanding and the vulnerability is highly difficult to trigger.

- **Rare**

Rare likelihood generally means that the vulnerability requires an extremely high cost or the conditions for exploitation are extremely difficult to achieve.

### 3.1.4 Fix Results Status

| Status | Description |
| --- | --- |
| **Fixed** | The project party fully fixes a vulnerability. |
| **Partially Fixed** | The project party did not fully fix the issue, but only mitigated the issue. |
| **Acknowledged** | The project party confirms and chooses to ignore the issue. |

## 3.2 Audit Categories

| No. | Categories | Subitems |
|---|---|---|
| 1 | Coding Conventions | SPL Token Standards |
| | | Visibility Specifiers |
| | | Lamport Check |
| | | Account Check |
| | | Signer Check |
| | | Program Id Check |
| | | Deprecated Items |
| | | Redundant Code |
| 2 | General Vulnerability | Integer Overflow/Underflow |
| | | Reentrancy |
| | | Pseudo-random Number Generator (PRNG) |
| | | Transaction-Ordering Dependence |
| | | DoS (Denial of Service) |
| | | Function Call Permissions |
| | | Returned Value Security |
| | | Replay Attack |
| | | Overriding Variables |
| | | Third-party Protocol Interface Consistency |
| 3 | Business Security | Business Logics |
| | | Business Implementations |
| | | Manipulable Token Price |
| | | Centralized Asset Control |
| | | Asset Tradability |
| | | Arbitrage Attack |

Beosin classified the security issues of smart contracts into three categories: Coding Conventions, General Vulnerability, Business Security. Their specific definitions are as follows:

- **Coding Conventions**

Audit whether smart contracts follow recommended language security coding practices. For example, smart contracts developed in Solidity language should fix the compiler version and do not use deprecated keywords.

- **General Vulnerability**

General Vulnerability include some common vulnerabilities that may appear in smart contract projects. These vulnerabilities are mainly related to the characteristics of the smart contract itself, such as integer overflow/underflow and denial of service attacks.

- **Business Security**

Business security is mainly related to some issues related to the business realized by each project, and has a relatively strong pertinence. For example, whether the lock-up plan in the code match the white paper, or the flash loan attack caused by the incorrect setting of the price acquisition oracle.

*Note that the project may suffer stake losses due to the integrated third-party protocol. This is not something Beosin can control. Business security requires the participation of the project party. The project party and users need to stay vigilant at all times.

## 3.3 Disclaimer

The Audit Report issued by Beosin is related to the services agreed in the relevant service agreement. The Project Party or the Served Party (hereinafter referred to as the "Served Party") can only be used within the conditions and scope agreed in the service agreement. Other third parties shall not transmit, disclose, quote, rely on or tamper with the Audit Report issued for any purpose.

The Audit Report issued by Beosin is made solely for the code, and any description, expression or wording contained therein shall not be interpreted as affirmation or confirmation of the project, nor shall any warranty or guarantee be given as to the absolute flawlessness of the code analyzed, the code team, the business model or legal compliance.

The Audit Report issued by Beosin is only based on the code provided by the Served Party and the technology currently available to Beosin. However, due to the technical limitations of any organization, and in the event that the code provided by the Served Party is missing information, tampered with, deleted, hidden or subsequently altered, the audit report may still fail to fully enumerate all the risks.

The Audit Report issued by Beosin in no way provides investment advice on any project, nor should it be utilized as investment suggestions of any type. This report represents an extensive evaluation process designed to help our customers improve code quality while mitigating the high risks in blockchain.

## 3.4 About Beosin

Beosin is the first institution in the world specializing in the construction of blockchain security ecosystem. The core team members are all professors, postdocs, PhDs, and Internet elites from world-renowned academic institutions. Beosin has more than 20 years of research in formal verification technology, trusted computing, mobile security and kernel security, with overseas experience in studying and collaborating in project research at well-known universities. Through the security audit and defense deployment of more than 2,000 smart contracts, over 50 public blockchains and wallets, and nearly 100 exchanges worldwide, Beosin has accumulated rich experience in security attack and defense of the blockchain field, and has developed several security products specifically for blockchain.

BEOSIN
Web3 Security & Compliance

**Official Website**
https://www.beosin.com

**Telegram**
https://t.me/beosin

**X**
https://x.com/Beosin_com

**Email**
service@beosin.com

**LinkedIn**
https://www.linkedin.com/company/beosin/