QuillAudits

Audit Report
November, 2024

For

>SWAYE_

Table of Content

Executive Summary

Project Name	Swaye
Project URL	https://swaye.me/
Overview	The WAYE token is a custom cryptocurrency implementation on the Sui blockchain. It's designed as a utility token with a fixed maximum supply of 1 billion tokens (with 9 decimal places). The contract implements basic token functionality including minting and burning mechanisms, with proper supply control measures.
Audit Scope	The Scope of the Audit is to analyse the Code Quality ,Security and Correctness of Swaye Smart Contract
Contracts In-Scope	https://bitbucket.org/web-ogs/waye-token
Commit Hash	daad02428e027145cb2d74172e2c1f8578d42d9f
Language	Move
Blockchain	Sui
Method	Manual Analysis, Functional Testing, Automated Testing
First Review	20th November 2024 - 22nd November 2024
Fixed In	NA

Number of Security Issues per Severity

2
Issues Found

🟥 High 🟨 Medium

🟩 Low 🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	2
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	0

Checked Vulnerabilities

- ✓ Transaction-ordering dependence
- ✓ Timestamp dependence
- ✓ Integer overflow/underflow by bit operations
- ✓ Number of rounding errors
- ✓ Denial of service / logical oversights
- ✓ Access control
- ✓ Centralization of power
- ✓ Business logic contradicting the specification
- ✓ Code clones, functionality duplication
- ✓ Gas usage
- ✓ Arbitrary token minting
- ✓ Unchecked CALL Return Values
- ✓ The flow of capability
- ✓ Witness Type

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments, match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Informational Issues

1. Unnecessary protocol implementation.

Path

https://bitbucket.org/web-ogs/waye-token

Function

burn()

Description

There is an unnecessary protocol implementation. The protocol added an extra public entry burn function to the standard code:

```
public entry fun burn(    treasury_cap: &mut TreasuryCap<WAYE>,
    coin: Coin<WAYE>
) {
    coin::burn(treasury_cap, coin);
}
```

This is essentially a wrapper around the standard library's coin::burn() function, which adds no additional functionality.

In the Sui standard library, unlike the mint(), the coin::burn() function is already an entry function that handles burning coins and decreasing the total supply.

https://github.com/MystenLabs/sui/blob/main/crates/sui-framework/packages/sui-framework/sources/coin.move#L315

Recommendation

Therefore, the protocol's implementation is unnecessary. Swaye team could directly use coin::burn() without defining a new function.

Status

Acknowledged

2. Missing Test Cases

Description

No test cases: Missing key test cases for:-
#[test]

- fun test_mint_max_supply_restriction()
- fun test_get_current_supply()
- fun test_mint()

are all missing

Recommendation

It is Recommended to Perform Unit Testing on Above Functions.

Status

Acknowledged

coding best practices - call freeze object before transferring treasuryCap

```
+   // Make the coin metadata public and immutable
+   transfer::public_freeze_object(metadata);
    // Transfer the treasury cap to the module publisher
    transfer::public_transfer(treasury_cap, tx_context::sender(ctx));
-   // Make the coin metadata public and immutable
-   transfer::public_freeze_object(metadata);
```

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of Swaye. We performed our audit according to the procedure described above.

Some issues of informational severity were found. Some suggestions, gas optimizations and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Swaye. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Swaye. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of Swaye to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

1000+
Audits Completed

$30B
Secured

1M+
Lines of Code Audited

Follow Our Journey

Audit Report
November, 2024

For

>SWAYE_

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com