QuillAudits

Audit Report
July, 2024

For

Slash Vision Labs

Table of Content

Executive Summary

Project Name

SlashVisionLabs Token

Overview

SlashVisionLabsToken is an ERC20 token implementation with the following key features:

1. **Token Details:**
   - Name: Slash Vision Labs
   - Symbol: SVL
   - Decimals: 18
   - Total Supply: 10 billion tokens
2. **Ownership:**
   - Implements an ownership system with transfer capability
   - Only the owner can mint tokens and transfer ownership
3. **Minting:**
   - Owner can mint tokens up to the total supply limit
   - Tracks total minted tokens
4. **Burning:**
   - Users can burn their own tokens
5. **Transfer Restrictions:**
   - Prevents transfers to the contract address
6. **Trusted Forwarder:**
   - Implements a trusted forwarder system, likely for meta-transactions
   - Custom _msgSender() and _msgData() functions to support the forwarder
7. **Inheritance:**
   - Inherits from OpenZeppelin's ERC20 contract

SlashVisionLabs is token Contract with added features for ownership control, minting limitations, and meta-transaction support.

Timeline

20th August 2024 - 22nd August 2024

Executive Summary

Method	Manual Review, Functional Testing, Automated Testing, etc. All the raised flags were manually reviewed and re-tested to identify any false positives.
Audit Scope	This audit aimed to analyze the SlashVisionLabs Token Codebase for quality, security, and correctness.
Blockchain	Mantle
Source Code	https://explorer.mantle.xyz/token/0xabBeED1d173541e0546B38b1C0394975be200000?tab=contract
Fixed In	NA

Number of Security Issues per Severity

High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	3	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	0

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

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.
- Implementation of ERC-20 token standards.
- 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.

Tools and Platforms used for Audit

Hardhat, Foundry.

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.

Low Severity Issues

1. Ownership transfer should be a 2 step process

Path

SlashVisionLabsToken.sol

Function

transferOwnership(address newOwner)

Description

The current process of transferring ownership involves owner calling `transferOwnership()`. This will change the newOnwer to the address provided by the previous owner.

This process might cause unintended consequence.

A single-step ownership transfer means that if an incorrect address is provided during the transfer of ownership or admin rights, the role could be permanently lost.

Recommendation

It is a best practice to use two-step ownership transfer pattern, meaning ownership transfer gets to a “pending” state and the new owner should claim his new rights, otherwise the old owner still has control of the contract. Consider using OpenZeppelin’s Ownable2Step contract.

Status

Acknowledged

2. Lack of address(0) checks while transferring ownership might cause unintended consequences

Path

SlashVisionLabsToken.sol

Function

transferOwnership(address newOwner)

Description

The function transferOwnership()does not check is newOwner is address(0). This might be problematic if ownership is transferred to a null address and is lost forever.

Recommendation

Create a safety check that reverts when newOwner == address(0)

Status

Acknowledged

SlashVisionLabs Team's Comment

we are assuming that it will be transferred ownership to zero address after minting, so we are doing this intentionally.

3. Centralization risk leads to central point of failure

Path

SlashVisionLabsToken.sol

Description

The most important address in the contract is owner's address.
Any compromise on the owner's address might create a point of failure since the owner has escalated privileges through which they can call tranferOwnership() to transfer ownership to an arbitrary address.

Recommendation

Ensure that the owner is a trusted role and is protected via multisig wallet.

Status

Acknowledged

Functional Tests Cases

Some of the tests performed are mentioned below:

- ✓ Should get the name of the token
- ✓ Should get the symbol of the token
- ✓ Should approve another account to spend token
- ✓ Should transfer tokens to other address
- ✓ Should approve another account to spend token

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 the SlashVisionLabs Token contract. We performed our audit according to the procedure described above.

Three issues of Low severity were found. In The End, the SlashVision Team Acknowledged all Issues.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in SlashVisionLabs Token smart contracts. 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 SlashVisionLabs Token smart contracts. 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 the SlashVisionLabs Token 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
July, 2024

For

Slash Vision Labs

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com