Veridise. Auditing Report

Hardening Blockchain Security with Formal Methods

FOR

SOV Token

Veridise Inc.
April 12, 2023

I Prepared For:

Shib Original Vision
https://etherscan.io/token/0x2C5BC2Ba3614fD27FCc7022eA71d9172E2632c16#code

I Prepared By:

Jon Stephens

I Contact Us: contact@veridise.com

I Version History:

April 10, 2023 Initial Draft

Contents

1 Executive Summary

From April 8, 2023 to April 9, 2023, Shib Original Vision engaged Veridise to review the security of their SOV Token. The review covered all of the on-chain behaviors contained in the ShibOriginalVisionToken contract. Veridise conducted the assessment over 2 person-days, with 1 engineers reviewing the code at address `0x2C5BC2Ba3614fD27FCc7022eA71d9172E2632c16`. The auditing strategy involved a tool-assisted analysis of the source code performed by Veridise engineers as well as extensive manual auditing.

Code assessment. The Shib Original Vision developers provided the source code of the SOV Token contract for review. The contract extends OpenZeppelin's ERC20 token implementation to create a deflationary token that can be traded by users. By extending OpenZeppelin's ERC20 implementation, the SOV Token developers inherit additional security features such as the `increaseAllowance` and `decreaseAllowance` functions that avoid the potential front-running issues of `approve`. The developers add additional logic that allows users to burn their tokens and a conditional trading restriction that prevents trades from a Uniswap WETH, SOVToken Pair if it would cause the user's balance to exceed some threshold. No additional documentation or tests were provided with the source code.

Summary of issues detected. The audit uncovered 2 issues, 1 of which was judged to be of medium severity. Specifically, V-SOV-VUL-001 points out that the logic used to limit trades with a Uniswap V2 pair can be easily circumvented in some cases and can lock funds in others. In addition, the auditors note 1 low-severity issue corresponding to risks associated with a centralized owner.

Disclaimer. We hope that this report is informative but provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the system is secure in all dimensions. In no event shall Veridise or any of its employees be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the results reported here.

2 Project Dashboard

Table 2.1: Application Summary.

Name	Address	Type	Platform
SOV Token	`0x2C5BC2Ba3614fD27FCc7022eA71d9172E2632c16`	Solidity	Ethereum

Table 2.2: Engagement Summary.

Dates	Method	Consultants Engaged	Level of Effort
April 8 - April 9, 2023	Manual & Tools	1	2 person-days

Table 2.3: Vulnerability Summary.

Name	Number	Resolved
Critical-Severity Issues	0	0
High-Severity Issues	0	0
Medium-Severity Issues	1	1
Low-Severity Issues	1	1
Warning-Severity Issues	0	0
Informational-Severity Issues	0	0
TOTAL	2	2

Table 2.4: Category Breakdown.

Name	Number
Logic Error	1
Centralization	1

3 Audit Goals and Scope

3.1 Audit Goals

The engagement was scoped to provide a security assessment of Shib Original Vision's on-chain SOV Token contract. In our audit, we sought to answer the following questions:

- Can the contract be manipulated such that tokens can be stolen from users?
- Can tokens be arbitrarily created?
- Can a user manipulate tokens that are owned by another user?
- Is SOV Token's limit logic effective?
- Does the token provide a safe API to avoid `approve` front-running issues?
- Is the token supply fixed, inflationary or deflationary?
- Is it possible for user funds to become locked in an address?

3.2 Audit Methodology & Scope

Audit Methodology. To address the questions above, our audit involved a combination of human experts and automated program analysis & testing tools. In particular, we conducted our audit with the aid of the following techniques:

- *Static analysis.* To identify potential common vulnerabilities, we leveraged our custom smart contract analysis tool Vanguard, as well as the open-source tool Slither. These tools are designed to find instances of common smart contract vulnerabilities, such as reentrancy and uninitialized variables.
- *Fuzzing/Property-based Testing.* We also leverage fuzz testing to determine if the protocol may deviate from the expected behavior. To do this, we formalize the desired behavior of the protocol as [V] specifications and then use our fuzzing framework OrCa to determine if a violation of the specification can be found.

Scope. The scope of the audit is limited to the publicly available code that is associated with the SOV Token address on etherscan.

Methodology. The Veridise auditors performed a manual audit of the code assisted by both static analyzers and automated testing.

3.3 Classification of Vulnerabilities

When Veridise auditors discover a possible security vulnerability, they must estimate its severity by weighing its potential impact against the likelihood that a problem will arise. Table 3.1 shows how our auditors weigh this information to estimate the severity of a given issue.

Table 3.1: Severity Breakdown.

	Somewhat Bad	Bad	Very Bad	Protocol Breaking
Not Likely	Info	Warning	Low	Medium
Likely	Warning	Low	Medium	High
Very Likely	Low	Medium	High	Critical

In this case, we judge the likelihood of a vulnerability as follows in Table 3.2:

Table 3.2: Likelihood Breakdown

Not Likely	A small set of users must make a specific mistake
Likely	Requires a complex series of steps by almost any user(s) - OR - Requires a small set of users to perform an action
Very Likely	Can be easily performed by almost anyone

In addition, we judge the impact of a vulnerability as follows in Table 3.3:

Table 3.3: Impact Breakdown

Somewhat Bad	Inconveniences a small number of users and can be fixed by the user
Bad	Affects a large number of people and can be fixed by the user - OR - Affects a very small number of people and requires aid to fix
Very Bad	Affects a large number of people and requires aid to fix - OR - Disrupts the intended behavior of the protocol for a small group of users through no fault of their own
Protocol Breaking	Disrupts the intended behavior of the protocol for a large group of users through no fault of their own

4 Vulnerability Report

In this section, we describe the vulnerabilities found during our audit. For each issue found, we log the type of the issue, its severity, location in the code base, and its current status (i.e., acknowleged, fixed, etc.). Table 4.1 summarizes the issues discovered:

Table 4.1: Summary of Discovered Vulnerabilities.

ID	Description	Severity	Status
V-SOV-VUL-001	Ineffective Transfer Limiting Logic	Medium	Fixed
V-SOV-VUL-002	Centralized Owner	Low	Fixed

4.1 Detailed Description of Issues

4.1.1 V-SOV-VUL-001: Ineffective Transfer Limiting Logic

Severity	Medium	**Address**	`0x2C5BC2B...`
Type	Logic Error	**Status**	Fixed
File(s)	ShibOriginalVisionToken.sol		
Location(s)	_beforeTokenTransfer		

The SOV Token owner has the ability to set a flag that, if enabled, places restrictions on transfers from the `uniswapV2Pair` address. Specifically, token transfers originating from that pair will revert if it would cause a user's balance to exceed the `maxHoldingAmount` threshold. While the `_beforeTokenTransfer` function correctly implements this filter as shown below, it can be easily circumvented by users in practice as long as `maxHoldingAmount` is greater than 0. For example, a user can use a proxy account to perform a swap on their behalf. However, if the `maxHoldingAmount` is set to 0 (or a sufficiently low value), funds can effectively be locked in the `uniswapV2Pair` address.

```
function _beforeTokenTransfer(
    address from,
    address to,
    uint256 amount
) override internal virtual {
    if (limited && from == uniswapV2Pair) {
        require(super.balanceOf(to) + amount < maxHoldingAmount, "Forbid");
    }
}
```

Snippet 4.1: The definition of `_beforeTokenTransfer` that limits transfers from the `uniswapV2Pair` address

Impact As the check can easily be circumvented, the SOV Token developers should not rely on this to prevent any undesirable user behavior. Additionally, note that if `maxHoldingAmount` is set to 0, the owner can effectively lock funds in the `uniswapV2Pair` address (which we note can be an arbitrary address).

Recommendations As this contract is already deployed, developers should ensure that `maxHoldingAmount` is not set to a low value or they will risk funds being locked at the specified address.

Developer Response We included the limiting logic to prevent potential sniping when the Uniswap L2 pair was created in the constructor. It has since been disabled and we do not intend to use it in the future.

4.1.2 V-SOV-VUL-002: Centralized Owner

Severity	Low	**Address**	`0x2C5BC2B...`
Type	Centralization	**Status**	Fixed
File(s)		N/A	
Location(s)		N/A	

Similar to other tokens, the SOV Token has an owner that is given special access to some functionality in the token. We noted that the owner of this contract appears to be an individual rather than a multisig or decentralized governance. This comes with some risks as there is a single point of failure. For example, the owner's private key could be compromised giving ownership access to another malicious user.

```
function setRule(bool _limited, address _uniswapV2Pair, uint256 _maxHoldingAmount)
    external onlyOwner {
    limited = _limited;
    uniswapV2Pair = _uniswapV2Pair;
    maxHoldingAmount = _maxHoldingAmount;
}
```

Snippet 4.2: The definition of `setRule` that a malicious owner could use to lock user funds

Impact In this case, a compromised owner would have limited impact as the only owner-exclusive functionality is the `setRule` function shown above. We note, however, that as discussed in V-SOV-VUL-001, this function could be used by a malicious individual to lock funds in an arbitrary address.

Recommendations Consider transferring ownership to a multi-sig contract or decentralized governance.

Developer Response Since we only planned on using the limiting logic for launch, which has now passed, we will transfer contract ownership to the zero address so that it cannot be used in the future.