



**CONTRACT
WOLF**

Blockchain Security - Smart Contract Audits

Security Assessment

December 6, 2022



| | |
|--|-----------|
| Disclaimer | 3 |
| Scope of Work & Engagement | 3 |
| Project Description | 4 |
| Risk Level Classification | 5 |
| Methodology | 6 |
| Used Code from other Frameworks / Smart Contracts (Imports) | 7 |
| Token Description | 8 |
| Inheritance Graph | 9 |
| Overall Checkup | 10 |
| Verify Claim | 11 |
| Write Functions of Contract | 12 |
| Call Graph | 13 |
| SWC Attacks | 14 |
| Audit Result | 16 |
| Findings | 17 |
| Audit Comments | 19 |

Disclaimer

ContractWolf.io audits and reports should not be considered as a form of project's "advertisement" and does not cover any interaction and assessment from "project's contract" to "external contracts" such as Pancakeswap or similar.

ContractWolf does not provide any warranty on its released reports.

ContractWolf should not be used as a decision to invest into an audited project and is not affiliated nor partners to its audited contract projects.

ContractWolf provides transparent report to all its "clients" and to its "clients participants" and will not claim any guarantee of bug-free code within its **SMART CONTRACT**.

ContractWolf presence is to analyze, audit and assess the client's smart contract's code.

Each company or projects should be liable to its security flaws and functionalities.

Scope of Work

SHIWA team agreed and provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract.

The goal of this engagement was to identify if there is a possibility of security flaws in the implementation of the contract or system.

ContractWolf will be focusing on contract issues and functionalities along with the projects claims from smart contract to their website, whitepaper and repository which has been provided by **SHIWA**.

Description

SHIWA is a solution of the future now from the hand of the SHIWA mascot that combines the hype of a meme with real utility in a blockchain, DAO options and NFTs marketing platform so that users can enjoy the advantages of decentralized governance and end organizational hierarchy.



Risk Level Classification

Risk Level represents the classification or the probability that a certain function or threat that can exploit vulnerability and have an impact within the system or contract.

Risk Level is computed based on CVSS Version 3.0

| Level | Value | Vulnerability |
|---------------|-----------|---|
| Critical | 9 - 10 | An Exposure that can affect the contract functions in several events that can risk and disrupt the contract |
| High | 7 - 8.9 | An Exposure that can affect the outcome when using the contract that can serve as an opening in manipulating the contract in an unwanted manner |
| Medium | 4 - 6.9 | An opening that could affect the outcome in executing the contract in a specific situation |
| Low | 0.1 - 3.9 | An opening but doesn't have an impact on the functionality of the contract |
| Informational | 0 | An opening that consists of information's but will not risk or affect the contract |

Auditing Approach

Every line of code along with its functionalities will undergo manual review to check its security issues, quality, and contract scope of inheritance. The manual review will be done by our team that will document any issues that there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:

- Review of the specifications, sources, and instructions provided to ContractWolf to make sure we understand the size, scope, and functionality of the smart contract.
- Manual review of code, our team will have a process of reading the code line-by-line with the intention of identifying potential vulnerabilities and security flaws.

2. Testing and automated analysis that includes:

- Testing the smart contract functions with common test cases and scenarios, to ensure that it returns the expected results.

3. Best practices review, the team will review the contract with the aim to improve efficiency, effectiveness, clarifications, maintainability, security, and control within the smart contract.

4. Recommendations to help the project take steps to secure the smart contract.

Used Code from other Frameworks/Smart Contracts (Direct Imports)

Imported Packages

- IERC20Permit
- Address
- SafeERC20
- IERC165
- IERC721
- IERC20
- EnumerableSet
- EnumerableMap
- ShiwaDao

Description

Optimization enabled: No

Capabilities

Components

| Version | Contracts | Libraries | Interfaces | Abstract |
|---------|-----------|-----------|------------|----------|
| 1.0 | 1 | 4 | 4 | 0 |

Exposed Functions

| Version | Public | Private | External | Internal |
|---------|--------|---------|----------|----------|
| 1.0 | 2 | 7 | 40 | 75 |

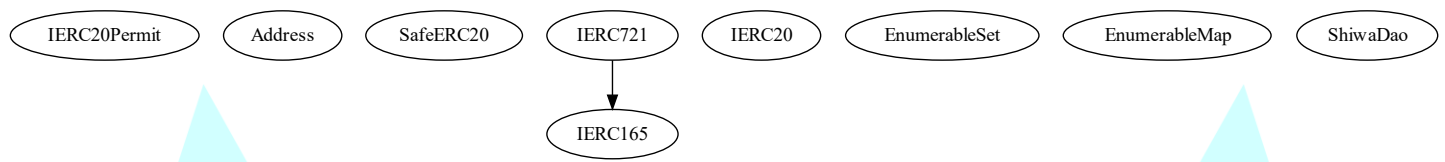
State Variables

| Version | Total | Public |
|---------|-------|--------|
| 1.0 | 21 | 10 |

Capabilities

| Version | Solidity Versions Observed | Experimental Features | Can Receive Funds | Uses Assembly | Has Destroyable Contracts |
|---------|----------------------------|-----------------------|-------------------|---------------|---------------------------|
| 1.0 | v0.8.0 | | No | Yes | No |

Inheritance Graph



Correct implementation of Token Standard

| Tested | Verified |
|--------|----------|
| ✓ | ✓ |

Overall Checkup (Smart Contract Security)

| Tested | Verified |
|--------|----------|
| ✓ | ✓ |

| Function | Description | Exist | Tested | Verified |
|--------------|--|-------|--------|----------|
| TotalSupply | Information about the total coin or token supply | ✓ | ✓ | ✓ |
| BalanceOf | Details on the account balance from a specified address | ✓ | ✓ | ✓ |
| Transfer | An action that transfers a specified amount of coin or token to a specified address | ✓ | ✓ | ✓ |
| TransferFrom | An action that transfers a specified amount of coin or token from a specified address | ✓ | ✓ | ✓ |
| Approve | Provides permission to withdraw specified number of coin or token from a specified address | ✓ | ✓ | ✓ |

Verify Claims

| Statement | Exist | Tested | Deployer |
|--------------------|-------|--------|----------|
| Renounce Ownership | — | — | — |
| Mint | — | — | — |
| Burn | — | — | — |
| Block | — | — | — |
| Pause | — | — | — |

Legend

| Attribute | Symbol |
|--------------------------|--------|
| Verified / Can | ✓ |
| Verified / Cannot | ✗ |
| Unverified / Not checked | 🚩 |
| Not Available | — |

Write Functions of Contract

1. createProposal (0xc32946f1)

2. createProposalSafe (0x8c77afef)

3. recoverERC20 (0x8980f11f)

4. updateFee (0x2740c197)

5. updateManager (0x3a35ae2d)

6. updateMinDuration (0x3a95225f)

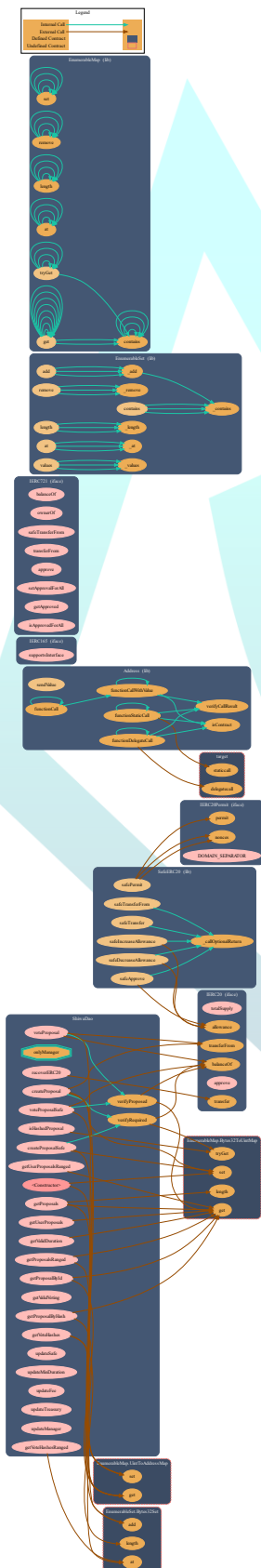
7. updateSafe (0xf6d3f169)

8. updateTreasury (0x7f51bb1f)

9. voteProposal (0x5384cfd5)

10. voteProposalSafe (0xd1256b73)

Call Graph



SWC Attacks

| ID | Title | Status |
|-------------------------|---|------------|
| SWC-136 | Unencrypted Private Data On-Chain | PASSED |
| SWC-135 | Code With No Effects | PASSED |
| SWC-134 | Message call with hardcoded gas amount | PASSED |
| SWC-133 | Hash Collisions with Multiple Variable Length Arguments | PASSED |
| SWC-132 | Unexpected Ether balance | PASSED |
| SWC-131 | Presence of unused variables | PASSED |
| SWC-130 | Right-To Left Override control character (U+202E) | PASSED |
| SWC-129 | Typographical Error | PASSED |
| SWC-128 | DoS With Block Gas Limit | PASSED |
| SWC-127 | Arbitrary Jump with Function Type Variable | NOT PASSED |
| SWC-126 | Insufficient Gas Griefing | PASSED |
| SWC-125 | Incorrect Inheritance Order | PASSED |
| SWC-124 | Write to Arbitrary Storage Location | PASSED |
| SWC-123 | Requirement Violation | PASSED |
| SWC-122 | Lack of Proper Signature Verification | PASSED |
| SWC-121 | Missing Protection against Signature Replay Attacks | PASSED |
| SWC-120 | Weak Sources of Randomness from Chain Attributes | PASSED |
| SWC-119 | Shadowing State Variables | PASSED |
| SWC-118 | Incorrect Constructor Name | PASSED |
| SWC-117 | Signature Malleability | PASSED |
| SWC-116 | Block values as a proxy for time | PASSED |
| SWC-115 | Authorization through tx.origin | PASSED |
| SWC-114 | Transaction Order Dependence | PASSED |
| SWC-113 | DoS with Failed Call | PASSED |
| SWC-112 | Delegate call to Untrusted Callee | PASSED |
| SWC-111 | Use of Deprecated Solidity Functions | PASSED |

| | | |
|--------------------------------|--------------------------------------|------------------|
| <u>SWC-110</u> | Assert Violation | LOW ISSUE |
| <u>SWC-109</u> | Uninitialized Storage Pointer | PASSED |
| <u>SWC-108</u> | State Variable Default Visibility | LOW ISSUE |
| <u>SWC-107</u> | Reentrancy | PASSED |
| <u>SWC-106</u> | Unprotected SELFDESTRUCT Instruction | PASSED |
| <u>SWC-105</u> | Unprotected Ether Withdrawal | PASSED |
| <u>SWC-104</u> | Unchecked Call Return Value | PASSED |
| <u>SWC-103</u> | Floating Pragma | LOW ISSUE |
| <u>SWC-102</u> | Outdated Compiler Version | PASSED |
| <u>SWC-101</u> | Integer Overflow and Underflow | PASSED |
| <u>SWC-100</u> | Function Default Visibility | PASSED |

AUDIT PASSED

High Issues

| | |
|---|---------|
| The caller can redirect execution to arbitrary bytecode locations (SWC-127) | L: 1662 |
|---|---------|

Low Issues

| | |
|--|--|
| A floating pragma is set (SWC-103) | L: 27, L: 29, L: 87, L: 344, L: 489, L: 512, L: 669, L: 759, L: 1137 |
| State variable visibility is not set (SWC-108) | L: 1673 |
| A user-provided assertion failed (SWC-110) | L: 1881, L: 1795, L: 1832 |

Findings

Description:

A floating pragma is set (SWC-103)

Suggestion:

Specific version to ensure that the bytecode does not vary between builds.

Description:

State variable visibility is not set (SWC-108)

Suggestion:

Specify variables as **public, internal, or private**.

Description:

An assertion violation was triggered (SWC-110)

Suggestion:

Consider whether the condition checked in the `assert()` is actually an invariant. If not, replace the `assert()` statement with a `require()` statement.

If the exception is indeed caused by unexpected behaviour of the code, fix the underlying bug(s) that allow the assertion to be violated.

Description:

The caller can redirect execution to arbitrary bytecode locations (SWC-127)

Suggestion:

The use of assembly should be minimal. A developer should not allow a user to assign arbitrary values to function type variables.



Audit Comments

- Manager can set minimum voting tokens and proposal tokens
- Manager can update safe status
- Manager can update minimum duration
- Manager can update treasury address
- Manager can set/unset addresses as manager
- Manager can collect tokens from contract
- Owner cannot burn tokens
- Owner cannot block user
- Owner cannot pause contract
- Owner cannot mint after initial deployment



CONTRACTWOLF

Blockchain Security - Smart Contract Audits