# Smart Contract
# Security Audit Report

# Table Of Contents

**1 Executive Summary**

**2 Audit Methodology**

**3 Project Overview**

3.1 Project Introduction

3.2 Vulnerability Information

**4 Code Overview**

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

**5 Audit Result**

**6 Statement**

# 1 Executive Summary

On 2022.11.22, the SlowMist security team received the ZKSAFE team's security audit application for ZKSAFE, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

| Test method | Description |
| --- | --- |
| Black box testing | Conduct security tests from an attacker's perspective externally. |
| Grey box testing | Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses. |
| White box testing | Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc. |

The vulnerability severity level information:

| Level | Description |
| --- | --- |
| Critical | Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities. |
| High | High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities. |
| Medium | Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities. |
| Low | Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed. |
| Weakness | There are safety risks theoretically, but it is extremely difficult to reproduce in engineering. |

| Level | Description |
| --- | --- |
| Suggestion | There are better practices for coding or architecture. |

# 2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

| Serial Number | Audit Class | Audit Subclass |
| --- | --- | --- |
| 1 | Overflow Audit | - |
| 2 | Reentrancy Attack Audit | - |
| 3 | Replay Attack Audit | - |
| 4 | Flashloan Attack Audit | - |
| 5 | Race Conditions Audit | Reordering Attack Audit |
| 6 | Permission Vulnerability Audit | Access Control Audit |
| | | Excessive Authority Audit |

| Serial Number | Audit Class | Audit Subclass |
| --- | --- | --- |
| 7 | Security Design Audit | External Module Safe Use Audit |
| | | Compiler Version Security Audit |
| | | Hard-coded Address Security Audit |
| | | Fallback Function Safe Use Audit |
| | | Show Coding Security Audit |
| | | Function Return Value Security Audit |
| | | External Call Function Security Audit |
| | | Block data Dependence Security Audit |
| | | tx.origin Authentication Security Audit |
| 8 | Denial of Service Audit | - |
| 9 | Gas Optimization Audit | - |
| 10 | Design Logic Audit | - |
| 11 | Variable Coverage Vulnerability Audit | - |
| 12 | "False Top-up" Vulnerability Audit | - |
| 13 | Scoping and Declarations Audit | - |
| 14 | Malicious Event Log Audit | - |
| 15 | Arithmetic Accuracy Deviation Audit | - |
| 16 | Uninitialized Storage Pointer Audit | - |
| 17 | Circuit Trusted Setup Risks | - |
| 18 | Overflow of Circuit Operations | - |

| Serial Number | Audit Class | Audit Subclass |
| --- | --- | --- |
| 19 | Input Signal Cracking | - |
| 20 | Input Signal Leakage | - |

# 3 Project Overview

## 3.1 Project Introduction

ZKSAFE

Module:

Circuits + ZKPass+ SafeBox

Project address:

https://etherscan.io/address/0x8528d5a340Bef2e50844CDABdFa21bC6B57c3982

## 3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

| NO | Title | Category | Level | Status |
| --- | --- | --- | --- | --- |
| N1 | Exhaustive attack risk | Input Signal Cracking | Low | Acknowledged |
| N2 | Front-running risk | Race Conditions Vulnerability | Low | Acknowledged |
| N3 | Not checking "pwdhash" is the same as before | Design Logic Audit | Suggestion | Fixed |
| N4 | Not checking "newOwner" is the same as before | Design Logic Audit | Suggestion | Fixed |

| NO | Title | Category | Level | Status |
| --- | --- | --- | --- | --- |
| N5 | Risk of centralization in the process of trusted settings | Circuit Trusted Setup Risks | Suggestion | Acknowledged |
| N6 | Gas optimization | Gas Optimization Audit | Suggestion | Fixed |

# 4 Code Overview

## 4.1 Contracts Description

The main network address of the contract is as follows:

**The code was not deployed to the mainnet.**

## 4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

| ZKPass | | | |
| --- | --- | --- | --- |
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| resetPassword | Public | Can Modify State | - |
| verify | Public | Can Modify State | - |
| verifyProof | Internal | - | - |

| Safebox | | | |
| --- | --- | --- | --- |
| Function Name | Visibility | Mutability | Modifiers |

| Safebox | | | |
|---|---|---|---|
| <Constructor> | Public | Can Modify State | - |
| <Receive Ether> | External | Payable | - |
| owner | Public | - | - |
| _checkOwner | Internal | - | - |
| _transferOwnership | Internal | Can Modify State | - |
| init | External | Can Modify State | - |
| transferOwnership | External | Can Modify State | onlyOwner |
| _doTransferOwnership | Private | Can Modify State | - |
| withdrawETH | External | Can Modify State | onlyOwner |
| withdrawERC20 | External | Can Modify State | onlyOwner |
| withdrawERC721 | External | Can Modify State | onlyOwner |
| getSocialRecover | Public | - | - |
| setSocialRecover | External | Can Modify State | onlyOwner |
| transferOwnership2 | External | Can Modify State | - |

| SafeboxFactory | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| createSafebox | Public | Can Modify State | - |
| getSafeboxAddr | Public | - | - |

| SafeboxFactory | | | |
|---|---|---|---|
| changeSafeboxOwner | External | Can Modify State | - |

## 4.3 Vulnerability Summary

**[N1] [Low] Exhaustive attack risk**

**Category: Input Signal Cracking**

**Content**

- contracts/zkPass/ZKPass.sol

ZKSAFE's witness data PWDHASH is stored in the contract, and the calculation method is public. The attacker can calculate the user's password through violent cracking. Therefore, Only if the user's private key is leaked, theoretically, the attacker can obtain the user's assets through brute force.

It takes 8S to crack a 6-digit pure digital password on the M1PRO machine.

For more estimated data, can use the calculation tool provided by the project party to estimate the brute force cracking time.

https://github.com/ZKSAFE/all-contracts/blob/dev/test/crack-estimate.js

**Solution**

**Status**

Acknowledged;

**[N2] [Low] Front-running risk**

**Category: Race Conditions Vulnerability**

**Content**

- contracts/zkSafe/Safebox.sol

`withdrawerc20`, `withdraweth` and `withdrawerc721` functions, there is a Front-Running risk.

Since the withdrawal address is `OWNER` itself, Only if the user's private key has been leaked, the attacker can wait for the victim to submit the withdrawal transaction through FRONT-Running.

**Solution**

**Status**

Acknowledged; The project party has modified it so that it can transfer to an address that is not the owner.

### [N3] [Suggestion] Not checking "pwdhash" is the same as before

**Category: Design Logic Audit**

**Content**

- contracts/zkPass/ZKPass.sol

`resetPassword` can be set to be the same as the last password.

```
    function resetPassword(
        uint[8] memory proof1,
        uint expiration1,
        uint allhash1,
        uint[8] memory proof2,
        uint pwdhash2,
        uint expiration2,
        uint allhash2
    ) public {
        uint nonce = nonceOf[msg.sender];

        if (nonce == 0) {
            //init password

            pwdhashOf[msg.sender] = pwdhash2;
            nonceOf[msg.sender] = 1;
            verify(msg.sender, proof2, 0, expiration2, allhash2);
        } else {
            //reset password

            // check old pwdhash
```

```
            verify(msg.sender, proof1, 0, expiration1, allhash1);

            // check new pwdhash
            pwdhashOf[msg.sender] = pwdhash2;
            verify(msg.sender, proof2, 0, expiration2, allhash2);
        }

        emit SetPassword(msg.sender, pwdhash2);
    }
```

**Solution**

Can check whether it is the same as the previous pwdhash.

**Status**

Fixed

### [N4] [Suggestion] Not checking "newOwner" is the same as before

**Category: Design Logic Audit**

**Content**

- contracts/zkSafe/Safebox.sol

`transferOwnership` and `transferOwnership2` do not check whether newOwner is the same as before.

**Solution**

Can check whether it is the same as the previous newOwner.

**Status**

Fixed

### [N5] [Suggestion] Risk of centralization in the process of trusted settings

**Category: Circuit Trusted Setup Risks**

**Content**

Lack of a verifiable trusted setup process, trusted setup requires multiple independent individuals to participate in the setup, and at least one of them is honest and not evil. If the parameters in the setting process (such as random

numbers) are all recorded intentionally, they can be used to falsify circuit calculation proofs.

**Solution**

Generating trusted settings using a publicly verifiable process or ceremony.

**Status**

Acknowledged; The project party generates trusted settings and publishes the video online.

https://www.youtube.com/watch?v=HIy8pQWrKEw

https://www.youtube.com/watch?v=nzmb775l0ZE

### [N6] [Suggestion] Gas optimization

**Category: Gas Optimization Audit**

**Content**

- contracts/zkSafe/SafeboxFactory.sol

Using `delete` will save more gas than directly setting address(0).

```
function changeSafeboxOwner(address fromOwner, address newOwner) external {
    address safeboxAddr = userToSafebox[fromOwner];
    require(
        safeboxAddr == _msgSender(),
        "SafeboxFactory::changeSafeboxOwner: fromOwner error"
    );
    require(
        userToSafebox[newOwner] == address(0),
        "SafeboxFactory::changeSafeboxOwner: newOwner's Safebox exist"
    );

    userToSafebox[fromOwner] = address(0);//SlowMist//
    userToSafebox[newOwner] = safeboxAddr;

    emit SafeboxOwner(fromOwner, address(0));
    emit SafeboxOwner(newOwner, safeboxAddr);
}
```

**Solution**

Can use delete instead of directly setting it to address(0).

**Status**

Fixed

# 5 Audit Result

| Audit Number | Audit Team | Audit Date | Audit Result |
| --- | --- | --- | --- |
| 0X002211290001 | SlowMist Security Team | 2022.11.22 - 2022.11.29 | Low Risk |

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 2 low risk, 4 suggestion vulnerabilities.

# 6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

**Official Website**

www.slowmist.com

**E-mail**

team@slowmist.com

**Twitter**

@SlowMist_Team

**Github**

https://github.com/slowmist