Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.06.14, the SlowMist security team received the OKX team's security audit application for OKX Account on StarkNet, 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.
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	Denial of Service Audit	-
5	Race Conditions Audit	-
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	Block Data Dependence Security Audit
		Explicit Visibility of Functions Audit
		Hard-coded Address Security Audit
		Function Return Value Security Audit
		External Module Safe Use Audit
		Unsafe External Call Audit

Serial Number	Audit Class	Audit Subclass
8	Gas Optimization Audit	-
9	Design Logic Audit	-
10	Arithmetic Accuracy Deviation Audit	-
11	Malicious Event Log Audit	-
12	Scoping and Declarations Audit	-
13	"False top-up" Vulnerability Audit	-
14	Variable Coverage Vulnerability Audit	-
15	Cross-chain Security Audit	-
16	Unsafe Assert Method Audit	-

3 Project Overview

3.1 Project Introduction

The account is a 2-of-2 custom multisig where the signer key is typically stored on the user's phone and the guardian key is managed by an off-chain service to enable fraud monitoring (e.g. trusted contacts, daily limits, etc) and recovery. More specifically, the guardian acts both as a co-validator for typical operations of the wallet, and as the trusted actor that can recover the wallet in case the signer key is lost or compromised.

Normal operations of the wallet (execute, changeSigner, changeGuardian, changeGuardianBackup, validateGuardianSignature, cancelEscape) require the approval of both parties to be executed.

Each party alone can trigger the escape mode (a.k.a. recovery) on the wallet if the other party is not cooperating or lost. An escape takes 7 days before being active, after which the non-cooperating party can be replaced. The wallet is asymmetric in favor of the signer who can override an escape triggered by the guardian. And a triggered escape can always be cancelled with the approval of both parties.

Currently, the wallet will set the guardian to zero address by default, so there is no need to use the social recovery feature.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing address check when setting up roles	Design Logic Audit	Suggestion	Ignored
N2	Redundant code	Other	Suggestion	Fixed
N3	Loss of permissions may make account recovery impossible	Excessive Authority Audit	Suggestion	Ignored

4 Code Overview

4.1 Contracts Description

https://github.com/okx/Web3-Contracts-Starknet

Audit Version:

https://github.com/okx/Web3-Contracts-Starknet

commit: 9003985feaf62ee49e1e657703da4795b2b0bf81

Fixed Version:

https://github.com/okx/Web3-Contracts-Starknet

commit: 05a215e50463fce30ab8ecfafccf99ab7095a237

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:

Proxy		
Function Name	Mutability	Modifiers
constructor	Can Modify State	constructor
__default__	Can Modify State	external raw_input raw_output
__l1_default__	Can Modify State	l1_handle raw_input
get_implementation	-	view

OKXAccount		
Function Name	Mutability	Modifiers
__validate__	Can Modify State	external
__execute__	Can Modify State	external raw_output
__validate_declare__	Can Modify State	external
__validate_deploy__	Can Modify State	external raw_input
isValidSignature	-	view
supportsInterface	-	view
initialize	Can Modify State	external
upgrade	Can Modify State	external
execute_after_upgrade	Can Modify State	external
changeSigner	Can Modify State	external
changeGuardian	Can Modify State	external
changeGuardianBackup	Can Modify State	external
triggerEscapeGuardian	Can Modify State	external
triggerEscapeSigner	Can Modify State	external

OKXAccount		
cancelEscape	Can Modify State	external
escapeGuardian	Can Modify State	external
escapeSigner	Can Modify State	external
getSigner	-	external
getGuardian	-	external
getGuardianBackup	-	external
getEscape	-	external
getVersion	-	external
getName	-	external
is_valid_signature	-	external

4.3 Vulnerability Summary

[N1] [Suggestion] Missing address check when setting up roles

Category: Design Logic Audit

Content

For multi-signature accounts, different roles should be set to different addresses so that other roles can be used to recover the wallet account in the event of a loss or compromise of the private key. If all the multi-signature roles are set to the same address, the wallet will never be recovered if the private key is lost.

Code Location:

contracts/account/library.cairo

```
    func initialize{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, range_check_ptr}(
        signer: felt, guardian: felt
    ) {
        // check that we are not already initialized
        let (current_signer) = _signer.read();
        with_attr error_message("OKX: already initialized") {
            assert current_signer = 0;
        }
        // check that the target signer is not zero
        with_attr error_message("OKX: signer cannot be null") {
            assert_not_zero(signer);
        }
        // initialize the contract
        _signer.write(signer);
        _guardian.write(guardian);
        return ();
    }

    ...

    func change_signer{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*,
range_check_ptr}(
        new_signer: felt
    ) {
        // only called via execute
        assert_only_self();

        // change signer
        with_attr error_message("OKX: signer cannot be null") {
            assert_not_zero(new_signer);
        }
        _signer.write(new_signer);
        signer_changed.emit(new_signer=new_signer);
        return ();
    }

    func change_guardian{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*,
range_check_ptr}(
        new_guardian: felt
    ) {
        alloc_locals;

        // only called via execute
        assert_only_self();

        // make sure guardian_backup = 0 when new_guardian = 0
        let (guardian_backup) = _guardian_backup.read();
        if (new_guardian == 0) {
            with_attr error_message("OKX: new guardian invalid") {
                assert guardian_backup = 0;
            }
        }

        // change guardian
        _guardian.write(new_guardian);
        guardian_changed.emit(new_guardian=new_guardian);
        return ();
    }

    func change_guardian_backup{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*,
range_check_ptr}(
        new_guardian: felt
    ) {
        // only called via execute
        assert_only_self();

        // no backup when there is no guardian set
        assert_guardian_set();

        // change guardian
        _guardian_backup.write(new_guardian);
        guardian_backup_changed.emit(new_guardian=new_guardian);
        return ();
    }

    func escape_guardian{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*,
range_check_ptr}(
        new_guardian: felt
    ) {
        alloc_locals;

        ...

        // change guardian
        assert_not_zero(new_guardian);
        _guardian.write(new_guardian);
        guardian_escaped.emit(new_guardian=new_guardian);

        return ();
    }

    func escape_signer{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*,
range_check_ptr}(
        new_signer: felt
    ) {
        alloc_locals;

        ...

        // change signer
        assert_not_zero(new_signer);
        _signer.write(new_signer);
        signer_escaped.emit(new_signer=new_signer);
        return ();
    }
```

Solution

It is recommended to check that the signer role and the guardian role (or guardian_backup role) should not be equal when setting up the role.

Status

Ignored; The response from the project team: The feature is social recovery related and the wallet will currently set the guardian to a zero address by default without the need to use the social recovery feature.

[N2] [Suggestion] Redundant code

Category: Other

Content

1.In the OKXAccount contract, the code logic of the isValidSignature and is_valid_signature functions is the same: Both of these verify the legitimacy of the signature by calling OKXModel's is_valid_signature function.

Code Location:

contracts/account/OKXAccount.cairo

```
func isValidSignature{
    syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, ecdsa_ptr: SignatureBuiltin*,
range_check_ptr
}(hash: felt, sig_len: felt, sig: felt*) -> (isValid: felt) {
    let (isValid) = OKXModel.is_valid_signature(hash, sig_len, sig);
    return (isValid=isValid);
}

...

func is_valid_signature{
    syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, ecdsa_ptr: SignatureBuiltin*,
range_check_ptr
}(hash: felt, sig_len: felt, sig: felt*) -> (is_valid: felt) {
    let (is_valid) = OKXModel.a(hash, sig_len, sig);
    return (is_valid=is_valid);
}
```

2.In the library contract, the imported get_tx_info function is not used in the contract.

Code Location:

contracts/account/library.cairo#L11

```
from starkware.starknet.common.syscalls import (
    library_call,
    call_contract,
    get_tx_info,
    get_contract_address,
    get_caller_address,
    get_block_timestamp,
)
```

3.In the OKXAccount contract, the imported memcpy function is not used in the contract.

Code Location:

contracts/account/OKXAccount.cairo#L5

```
...

from starkware.cairo.common.memcpy import memcpy

...
```

Solution

It is recommended to merge these two functions into the same function and remove the unused codes.

Status

Fixed

[N3] [Suggestion] Loss of permissions may make account recovery impossible

Category: Excessive Authority Audit

Content

When the signer role's key is lost, the guardian role can change the signer via the triggerEscapeSigner function and escapeSigner function to restore use of the account.

However, there is a situation when the signer role's key is obtained by a hacker, who can then overwrite the escaping signer in progress by calling the triggerEscapeGuardian function. This will cause the expected account recovery to fail. Then the hacker can replace the guardian with escapeGuardian and the operation could not be overwritten, allowing the hacker to take over the entire account for any malicious operation.

Solution

It is recommended that a social recovery function be added so that when a signer is lost, the signer can be changed directly after it has been verified by guardian and another third party role (e.g. guardian backup or other) to recover the account.

Status

Ignored; The response from the project team: The feature is social recovery related and the wallet will currently set the guardian to a zero address by default without the need to use the social recovery feature.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002306260001	SlowMist Security Team	2023.06.14 - 2023.06.26	Passed

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 3 suggestion vulnerabilities. And 2 suggestion vulnerabilities were ignored; All other findings were fixed. The code was not deployed to the mainnet.

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.

SLOWMIST

Official Website

www.slowmist.com

✉

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist