# ZKSwap Security Audit Report

Contents

1. Executive Summary 1

2. Scope of Audit 3

3 Coverage 4

3.1 Target Code and Revision 4

3.2 Areas of Concern 4

4 Risk Analysis 6

4.1 Consistency security audit 6

4.1.1 Concurrent security audit 6

4.1.2 Non-deterministic library audit 6

4.2 Encryption security audit 6

4.2.1 Random number generation algorithm audit 6

4.2.2 Private key storage audit 7

4.2.3 Hash intensity audit 7

4.2.4 Length extension attack audit 8

4.2.5 Transaction malleability attack audit 8

4.3 Account and transaction model security 8

4.3.1 Transaction verification audit 8

4.3.2 Transaction replay audit 8

4.3.3 "False Top-up" audit 9

4.4 Zero-knowledge proof circuit audit(Black Box Test) 9

4.4.1 Transaction signature script 9

4.4.2 Test coverage 10

5 Findings 12

5.1 Multiple crate versions are too low and there are security vulnerabilities[low-risk] 13

5.2 Transaction replay attack vulnerability[critical-risk] 14

5.3 Withdrawal fee attack vulnerability[high-risk] 15

5.4 Lack of a risk control mechanism for spam transactions[low-risk] 15

5.5 There is a risk of "False Top-up" attacks[weakness] 15

5.6 The configuration file saves the private key information in plain text[enhancement] 15

6 Fix Log 15

7 Conclusion 16

8. Statement 17

# 1. Executive Summary

On Dec.01, 2020, the SlowMist security team received the L2labs team's security audit application for ZKSwap, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

ZKSwap is a token Swap protocol based on Automated Market Maker (AMM). Through ZK-Rollup technology, the full set of uniswap functions are realized in Layer-2, while providing unlimited scalability and privacy. ZKSwap provides liquidity providers and traders with ultra-high throughput Swap infrastructure, and transactions do not require any Gas fees.

SlowMist blockchain system test method:

| | |
|---|---|
| Black box testing | Conduct security tests from an attacker's perspective externally. |
| Grey box testing | Conduct security testing on code module 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 suck as nodes, SDK, etc. |

In black box testing and gray box testing, we use methods such as fuzz testing and script testing to test the robustness of the interface or the stability of the components by feeding random data or constructing data with a specific structure, and to mine some boundaries Abnormal performance of the system under conditions such as bugs or abnormal performance. In white box testing, we use methods such as code review, combined with the relevant experience accumulated by the security team on known blockchain security vulnerabilities, to analyze the object definition and logic implementation of the code to ensure that the code has the key components of the key logic. Realize no known vulnerabilities; at the same time, enter the vulnerability mining mode for new scenarios and new technologies, and find possible 0day errors.

SlowMist blockchain risk level:

| | |
|---|---|
| Critical vulnerabilities | Critical vulnerabilities will have a significant impact on the security of the blockchain, and it is strongly recommended to fix the critical vulnerabilities. |
| High-risk vulnerabilities | High-risk vulnerabilities will affect the normal operation of blockchain. It is strongly recommended to fix high-risk vulnerabilities. |
| Medium-risk vulnerabilities | Medium vulnerability will affect the operation of blockchain. It is recommended to fix medium-risk vulnerabilities. |
| Low-risk vulnerabilities | Low-risk vulnerabilities may affect the operation of blockchain in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed. |
| Weaknesses | There are safety risks theoretically, but it is extremely difficult to reproduce in engineering. |
| Enhancement Suggestions | There are better practices for coding or architecture. |

## 2. Scope of Audit

The main types of security audit include:

| No. | Audit category | Subclass | Audit result |
|---|---|---|---|
| 1 | Code static check | RUSTSEC audit | Pass |
| | | Numerical overflow audit | Pass |
| 2 | Consistency security audit | Concurrent security audit | Pass |
| | | Non-deterministic library audit | Pass |
| | | Parameter validation audit | Pass |
| 3 | Encrypted signature security | Random number generation algorithm audit | Pass |
| | | Private key storage audit | Pass |
| | | Cryptographic component call audit | Pass |
| | | Hash intensity audit | Pass |
| | | Transaction malleability attack audit | Pass |
| 4 | Account and transaction model security | Authority verification audit | Pass |
| | | Transaction replay audit | Pass |

| | | | |
|---|---|---|---|
| | | “False Top-up “ audit | Pass |
| 5 | Zero-knowledge proof circuit audit | - | Pass |

# 3 Coverage

## 3.1 Target Code and Revision

| | |
|---|---|
| Source | https://github.com/l2labs/zkswap/tree/master/core |
| Version | V0.9 |
| Type | Ethereum Layer2 App |
| Platform | Rust |

## 3.2 Areas of Concern

(1). circuit

Module description: Zero-knowledge proof circuit

Audit content: run unit tests and conduct security tests on related methods

(2). plasma

Module description: main business logic layer, such as transaction pair management, transfer, exchange, etc.

Audit content: Audit the value overflow, transaction slippage protection, business logic, etc.

(3). storage

Module description: database storage layer

Audit content: Perform security scans on dependent libraries, run unit tests, and perform security tests on related methods

(4). key_generator

Module description: ZKSwap key generator. Generate ZKSwap main circuit (for various block sizes) and generate outflow circuit verification key. Generate a verifier contract based on the verification key.

Audit content: Perform security scans on dependent libraries, run unit tests, and perform security tests on related methods

(5). prover

Module description: Zero-knowledge proof calculation

Audit content: Perform security scans on dependent libraries, run unit tests, and perform security tests on related methods

(6). data_restore

Module description: Restore ZKSwap state from smart contract.

Audit content: Perform security scans on dependent libraries, run unit tests, and perform security tests on related methods

(7). server

Module description: zkswap server main program

Audit content: Perform security scans on dependent libraries, run unit tests, and perform security tests on related methods

(8). eth_client

Module description: Ethereum transaction signature library

Audit content: Perform security scans on dependent libraries, run unit tests, and perform security tests on related methods

(9). models

Module description: model definition

Audit content: Perform security scans on dependent libraries, run unit tests, and perform security tests on related methods

(10). exit_cli / tok_cli

Module description: local command line tool

Audit content: Perform security scans on dependent libraries, run unit tests, and perform security tests on related methods

(11). spec_test / testkit / loadtest

Module description: Test module.

Audit content: Perform security scans on dependent libraries, run unit tests, and perform security tests on related methods

# 4 Risk Analysis

## 4.1 Consistency security audit

### 4.1.1 Concurrent security audit

Concurrency will lead to uncertain transaction status and should be avoided when executing transaction. The audit did not find that the relevant business logic uses std::thread::spawn for concurrent processing.

### 4.1.2 Non-deterministic library audit

When quoting operating system time and data outside the blockchain to process transactions, it will lead to inconsistent transaction status and should be avoided. The audit did not find the use of non-deterministic libraries in related business logic.

## 4.2 Encryption security audit

### 4.2.1 Random number generation algorithm audit

- core/plasma/benches/criterion/ops.rs

```
/// Creates a random ZKSync account.
fn generate_account() -> (H256, PrivateKey, Account) {
let default_balance = 1_000_000u32.into();

let rng = &mut thread_rng();
let sk = priv_key_from_fs(rng.gen());

let eth_sk = H256::random();
let address = PackedEthSignature::address_from_private_key(&eth_sk)
.expect("Can't get address from the ETH secret key");

let mut account = Account::default();
account.pub_key_hash = PubKeyHash::from_privkey(&sk);
account.address = address;
account.set_balance(ETH_TOKEN_ID, default_balance);

(eth_sk, sk, account)
}
```

Use crypto_exports::rand::{thread_rng, Rng} to generate random numbers to meet random number security.

### 4.2.2 Private key storage audit

- etc/env/dev.env.example

The OPERATOR_PRIVATE_KEY private key information is stored in plain text, and you need to pay attention to controlling the access rights of the private key.

### 4.2.3 Hash intensity audit

Weak hash functions such as md5 and sha1 were not found for encryption.

### 4.2.4 Length extension attack audit

In cryptography and computer security, a length extension attack is a type of attack where an attacker can use Hash(message1) and the length of message1 to calculate Hash(message1 ‖ message2) for an attacker-controlled message2, without needing to know the content of message1. Algorithms like MD5, SHA-1, and SHA-2 that are based on the Merkle–Damgård construction are susceptible to this kind of attack. The SHA-3 algorithm is not susceptible.
No error calls were found.

### 4.2.5 Transaction malleability attack audit

The ECDSA algorithm generates two large integers r and s combined as a signature, which can be used to verify transactions. And r and BN-s can also be used as signatures to verify transactions. In this way, the attacker gets a transaction, extracts the r and s of inputSig, uses r, BN-s to generate a new inputSig, and then forms a new transaction with the same input and output, but different TXID. Attacker Can successfully generate legal transactions at almost no cost without having the private key.
ZKSwap uses the Secp256k1 algorithm to sign, but the transaction structure does not use script input, and the signature is cleared before calculating the transactionId to ensure that the change of the signature will not affect the transactionId.

## 4.3 Account and transaction model security

### 4.3.1 Transaction verification audit

### 4.3.2 Transaction replay audit

There is no chainid in the transaction signature, and there is a risk of replay attacks between the main network and the test network.

### 4.3.3 "False Top-up" audit

You can initiate a malicious transfer by forging the amount and other parameters, and the payee needs to wait for Layer1 to confirm before indicating that the transfer is successful.

## 4.4 Zero-knowledge proof circuit audit(Black Box Test)

### 4.4.1 Transaction signature script

A token withdrawal example:

- zk_withdraw.js

```
var Web3 = require('web3');
const ethers_1 = require("ethers");
const utils_1 = require("./utils");
const crypto_1 = require("./crypto");
const signer_1 = require("./signer");
if (typeof web3 !== 'undefined') {
web3 = new Web3(web3.currentProvider);
} else {
web3 = new Web3(new Web3.providers.HttpProvider("http:/x.x.x.x:8545"));
}
const privKey = "";
const txMessageEthSignature="Withdraw 0.01 ETH\nTo: 0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
\nNonce: 12\nFee: 0.0 ETH\nAccount Id: 61";
const txSigned = web3.eth.accounts.sign(txMessageEthSignature, privKey);
console.log(txSigned);


var withdraw = {
// "type": "Withdraw",
"accountId": 61,
"from": "0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
"ethAddress": "0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
"tokenId": 0,
"amount": "10000000000000000",
"fee": "0",
"nonce": 12,
};


const message = "Access ZKSwap account.\n\nOnly sign this message for a trusted client!";
const signedBytes = utils_1.getSignedBytesFromMessage(message, false);
const signed = web3.eth.accounts.sign(ethers_1.utils.hexlify(signedBytes),privKey);
const signature = signed.signature;
const seed = ethers_1.ethers.utils.arrayify(signature);
const signer = signer_1.Signer.fromSeed(seed);

signedWithdrawTransaction = signer.signSyncWithdraw(withdraw);
console.log(signedWithdrawTransaction);
```

Using scripts, we can calculate the signature and EthereumSignature values in the transaction to construct a complete transaction.

```
curl 'https://alpha.zks.app/api/zksync/3' \
  -H 'Content-Type: application/json;charset=UTF-8' \
  --data-binary
'{"id":1,"jsonrpc":"2.0","method":"tx_submit","params":[{"type":"Withdraw","accountId":61,"from":"0xaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaa","to":"0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa","token":0,"amount":"10000000000000
000000","fee":"0","nonce":12,"signature":{"pubKey":"1b5831610cbf3912a893ed5747d9ba23912f19d6e6bc1fa6c0b28c67
2e81df0a","signature":"b14562008b7a709ea708741fe3fb652104cd6e7fb6d05768da134daa366d8f19bf23b9d714ba3f2d
5a540113b49cc910bc28792958562012511 9e8a3979f4402"}},{"type":"EthereumSignature","signature":"0xe42b41ca9e5b
83473702be386c83bd499572ea669ad06601a69e83b7618e67774091ab0155f9ca49c73cd44fa65dfaff9e784036c774c2
0ceb35bb30f630c65a1b"},null]}'
```

### 4.4.2 Test coverage

(1). Deposit:

Signature verification: correct

Is the accuracy accurate: accurate

Arrival time: 10 block confirmation

Token fake recharge: The recharge logic is implemented in the contract, and the contract code analysis reveals that the status cannot be changed by fake recharge.

Impact of transaction rollback:

Fund custody: custody by contract.

(2). Withdraw

Signature verification: correct

Is the accuracy accurate: accurate

Arrival time: commitBlock--verifyBlock--completeWithdrawals

Replay vulnerability: failed

Over withdrawal: failed

The ultra-small amount of 0.000000000000000001 ETH is successfully withdrawn. Each withdrawal of the Layer 1 contract will consume a separate fee. A large amount of dust transactions can be used to attack the system, which consumes a large amount of fee and causes economic losses.

https://ropsten.etherscan.io/tx/0xa3585bff87da688ae25095a7182c91546036d599a06fc5d60b68f4593f44e7ea

Unauthorized withdrawal: failed

(3). Transfer

Signature verification: correct

Is the accuracy accurate: accurate

Replay vulnerability: failed

Excess transfer: failed

Unauthorized transfer: failed

There is no handling fee for transfers, and a large amount of dust transactions can be used to block the system, causing a serious decline in system performance.

(4). Exchange

Signature verification: correct

Is the accuracy accurate: accurate

Slippage protection: Yes

(5). Add liquidity

Signature verification: correct

Is the accuracy accurate: accurate

Slippage protection: Yes

(6). Remove liquidity

Signature verification: correct

Is the accuracy accurate: accurate

Slippage protection: Yes

# 5 Findings

Vulnerability distribution:

| | | |
|---|---|---|
| Critical vulnerabilities | 1 | 🟩 |
| High-risk vulnerabilities | 1 | 🟦 |
| Medium-risk vulnerabilities | 0 | |
| Low-risk vulnerabilities | 2 | 🟦🟦 |
| Weaknesses | 1 | 🟩 |
| Enhancement Suggestions | 1 | 🟨 |
| Total | 6 | |

🟥Consistency security 🟨Encryption security 🟩Account and transaction model

🟩Zero-knowledge proof circuit 🟦Code static check 🟦other

## 5.1 Multiple crate versions are too low and there are security vulnerabilities[low-risk]

```
ID:        RUSTSEC-2020-0049
Crate:     actix-codec
Version:   0.1.2
Date:      2020-01-30
URL:       https://rustsec.org/advisories/RUSTSEC-2020-0049
Title:     Use-after-free in Framed due to lack of pinning
Solution:  upgrade to >= 0.3.0-beta.1

ID:        RUSTSEC-2020-0048
Crate:     actix-http
Version:   0.2.11
Date:      2020-01-24
URL:       https://rustsec.org/advisories/RUSTSEC-2020-0048
Title:     Use-after-free in BodyStream due to lack of pinning
Solution:  upgrade to >= 2.0.0-alpha.1

ID:        RUSTSEC-2020-0091
Crate:     arc-swap
```

```
Version:   0.4.7
Date:       2020-12-10
URL:         https://rustsec.org/advisories/RUSTSEC-2020-0091
Title:      Dangling reference in `access::Map` with Constant
Solution:   upgrade to >= 1.1.0 OR >= 0.4.8

ID:          RUSTSEC-2020-0052
Crate:      crossbeam-channel
Version:   0.4.3
Date:       2020-06-26
URL:         https://rustsec.org/advisories/RUSTSEC-2020-0052
Title:      Undefined Behavior in bounded channel
Solution:   upgrade to >= 0.4.4

ID:          RUSTSEC-2020-0060
Crate:      futures-task
Version:   0.3.5
Date:       2020-09-04
URL:         https://rustsec.org/advisories/RUSTSEC-2020-0060
Title:      futures_task::waker may cause a use-after-free if used on a type that isn't 'static
Solution:   upgrade to >= 0.3.6

ID:          RUSTSEC-2020-0059
Crate:      futures-util
Version:   0.3.5
Date:       2020-10-22
URL:         https://rustsec.org/advisories/RUSTSEC-2020-0059
Title:      MutexGuard::map can cause a data race in safe code
Solution:   upgrade to >= 0.3.7

ID:          RUSTSEC-2020-0043
Crate:      ws
Version:   0.9.1
Date:       2020-09-25
URL:         https://rustsec.org/advisories/RUSTSEC-2020-0043
Title:      Insufficient size checks in outgoing buffer in ws allows remote attacker to run the process out of memory
Solution:   No safe upgrade is available!
```

## 5.2 Transaction replay attack vulnerability[critical-risk]

There is no chainid in the transaction signature, and there is a risk of replay attacks between the main network and the test network.

## 5.3 Withdrawal fee attack vulnerability$_{\text{[high-risk]}}$

Each withdrawal will consume a separate fee, and a large amount of dust transactions can be used to attack the system, which causes the completeWithdrawals operation to consume a lot of fee and cause economic losses.

## 5.4 Lack of a risk control mechanism for spam transactions$_{\text{[low-risk]}}$

There is no fee mechanism for transfers, adding liquidity, and deleting liquidity. Each transaction will increase the fee for commitBlock operations, causing economic losses, and it is necessary to prevent spam transactions in the risk control mechanism.

## 5.5 There is a risk of "False Top-up" attacks$_{\text{[weakness]}}$

You can initiate a malicious transfer by forging the amount and other parameters, and the payee needs to wait for Layer1 to confirm before indicating that the transfer is successful.

## 5.6 The configuration file saves the private key information in plain text$_{\text{[enhancement]}}$

The OPERATOR_PRIVATE_KEY private key information is stored in plain text, and you need to pay attention to controlling the access rights of the private key.

# 6 Fix Log

Fix commit: 68ce0ff6f384366312b321056a0e42f0a6516b23

Fix description:

Fix 5.2: increase chainId and check in the circuit;

Fix 5.3: add server risk control module, and add corresponding fee design in the circuit;

Ignore 5.1, 5.4, 5.5, 5.6 low-risk risks, weaknesses and enhancements.

# 7 Conclusion

Audit result: PASS

Audit No. : BCA002012140001

Audit date: December 22, 2020

Review date: January 26, 2021

Audit team: SlowMist security team

Summary conclusion: After correction, all problems found have been fixed and the above risks have been eliminated by ZKSwap. Comprehensive assessed, ZKSwap has no risks above already.

# 8. 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 base 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 this 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.