EdgeSwap Security Audit Report

Contents

1. Executive Summary

On September 1, 2021 the SlowMist security team received the EdgeSwap team's security audit application for EdgeSwap, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

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

1	Code static check	RUSTSEC audit
		Numerical overflow audit
2	Consistency security audit	Concurrent security audit
		Non-deterministic library audit
		Parameter validation audit
3	Encrypted signature security	Random number generation algorithm audit
		Private key storage audit
		Cryptographic component call audit
		Hash intensity audit
		Transaction malleability attack audit
4	Account and transaction model security	Authority verification audit
		Transaction replay audit
		“False Top-up “ audit
5	Zero-knowledge proof circuit audit	-

3 Coverage

3.1 Target Code and Revision

Source	https://github.com/EdgeSwap/Circuit
Version	e72187eedfc21e4bbb3edcd6c765091a63cf9181
Type	Ethereum Layer2 App
Platform	Rust

3.2 Areas of Concern

lib/circuit/src/circuit.rs

lib/circuit/src/account.rs

lib/circuit/src/allocated_structures.rs

lib/circuit/src/primitives.rs

lib/circuit/src/exit_circuit.rs

lib/circuit/src/exit_lp_circuit.rs

lib/circuit/src/witness/add_liquidity.rs

lib/circuit/src/witness/claim_bonus.rs

lib/circuit/src/witness/create_pair.rs

lib/circuit/src/witness/deposit.rs

lib/circuit/src/witness/full_exit.rs

lib/circuit/src/witness/mining_maintenance.rs

lib/circuit/src/witness/remove_liquidity.rs

lib/circuit/src/witness/swap.rs

lib/circuit/src/witness/withdraw.rs

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

There is no random number related module in the audit scope.

4.2.2 Private key storage audit

There is no key storage related module in the audit scope.

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.
EdgeSwap 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

The server will return the transaction ID in the following cases, but the transaction will not be executed:

Withdraw:

It is not checked whether `from` and `signature` are consistent;

It is not checked whether `to` is zero address;

It is not checked whether the `Nonce` is too large;

Transfer:

It is not checked whether `to` is zero address;

It is not checked whether the `amount` is sufficient;

Swap:

It is not checked whether the `token_out_amount` is too large;

4.3.2 Transaction replay audit

The transaction signature contains `chainid`, and there is no risk of replay attacks between the main network and the test network.

Failed transactions can be replayed under certain circumstances.

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

Use test-submit-transaction.py provided by the project party to sign the transaction and construct a complete transaction.

```
Take the withdrawal transaction as an example, submit_withdraw()
Link: http://39.105.129.22:8478/tx
Parameters: {
    "signature": {},
    "tx": {
        "accountId": 14,
        "amount": "1",
        "fee": 1,
        "from": "0x116A9B16d40D69c8D5B4BFE375C166a7D5b718D5",
        "nonce": 5,
        "pair": 14,
        "signature": {
            "pubKey": "a03772dd696bbd431046047e56001ee50989a689f6e6e4ac8500b2eddca821a3",
            "signature":
"8ab21d549d560b282219ce609924b3f8b9a8216ff1bcc493496605a8626a1fa61412269bdc39247c69197ed598eeb256
5a11794e27da8ace8c5e6dbe88c56804"
        },
        "to": "0x116a9b16d40d69c8d5b4bfe375c166a7d5b718d5",
        "token": 3,
        "type": "Withdraw"
    }
}
```

4.4.2 Test coverage

(1). Deposit:

Signature verification: correct

Is the accuracy accurate: accurate

False recharge of tokens: The recharge logic is implemented in the contract, and no risk of false recharge has been found.

Fund custody: custody by contract.

(2). Withdraw

```
Small amount test:
{"success": false, "code": 7026, "message": "invalid fee: too more or too little"}
The amount is negative or zero
{"success": false, "code": 7212, "message": "amount is not suitable"}

Unauthorized transfer:
 {
      "success": true,
      "code": 0,
      "data": "82d00e21a7037c3aa94b46d3d2b6c8575d48445350cc1985d5318702900751db"
}
Enhancement: unchecked whether from and signature are consistent

Test interface Create withdrawal task:
 {
      "success": false,
      "code": 7005,
      "message": "invalid currency"
}

Withdraw to address 0
{
      "success": true,
      "code": 0,
      "data": "5f2314f3182740e007729cb960421de49df2d62c53da256da0d971f796bb7e2a"
}
Enhancement: the address of to is not 0 is not detected

Nonce is too small
{
      "success": false,
      "code": 7013,
      "message": "invalid nonce"
}
```

```
Nonce is too big
{
      "success": true,
      "code": 0,
      "data": "c388428dbc797b7cd2a8aea843b821552cd95fdcc5ef7d13198667c5b2126138"
}
Enhancement: Return the transaction ID, but the transaction eventually fails. It is recommended that the interface check
whether the Nonce is too large

Replay transaction
{
      "success": false,
      "code": 7013,
      "message": "invalid nonce"
}

Fake accountId
{
      "success": false,
      "code": 7018,
      "message": "User Account mismatch"
}

Forged signature content
{
      "success": false,
      "code": 7014,
      "message": "verify l2 signature fail"
}
```

(3). Transfer

Signature verification: correct

Is the accuracy accurate: accurate

Replay vulnerability: failed

Excess transfer: failed

Unauthorized transfer: failed

Free of charge within 50 transfers.

(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	0	
High-risk vulnerabilities	0	
Medium-risk vulnerabilities	2	🟩🟩
Low-risk vulnerabilities	1	🟦
Weaknesses	2	🟩🟩

Enhancement Suggestions	0	
Total	5	

🟥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]

```
Crate:          crossbeam-deque
Version:        0.7.3
Title:          Data race in crossbeam-deque
Date:           2021-07-30
ID:             RUSTSEC-2021-0093
URL:            https://rustsec.org/advisories/RUSTSEC-2021-0093
Solution:       Upgrade to >=0.7.4, <0.8.0 OR >=0.8.1
Dependency tree:
crossbeam-deque 0.7.3

Crate:          crossbeam-deque
Version:        0.8.0
Title:          Data race in crossbeam-deque
Date:           2021-07-30
ID:             RUSTSEC-2021-0093
URL:            https://rustsec.org/advisories/RUSTSEC-2021-0093
Solution:       Upgrade to >=0.7.4, <0.8.0 OR >=0.8.1
Dependency tree:
crossbeam-deque 0.8.0

Crate:          hyper
Version:        0.13.10
Title:          Integer overflow in `hyper`'s parsing of the `Transfer-Encoding` header leads to data loss
Date:           2021-07-07
ID:             RUSTSEC-2021-0079
URL:            https://rustsec.org/advisories/RUSTSEC-2021-0079
Solution:       Upgrade to >=0.14.10
Dependency tree:
```

```
hyper 0.13.10

Crate:          hyper
Version:        0.13.10
Title:          Lenient `hyper` header parsing of `Content-Length` could allow request smuggling
Date:           2021-07-07
ID:             RUSTSEC-2021-0078
URL:            https://rustsec.org/advisories/RUSTSEC-2021-0078
Solution:       Upgrade to >=0.14.10

Crate:          aes-ctr
Version:        0.3.0
Warning:        unmaintained
Title:          `aes-ctr` has been merged into the `aes` crate
Date:           2021-04-29
ID:             RUSTSEC-2021-0061
URL:            https://rustsec.org/advisories/RUSTSEC-2021-0061
Dependency tree:
aes-ctr 0.3.0

Crate:          aesni
Version:        0.6.0
Warning:        unmaintained
Title:          `aesni` has been merged into the `aes` crate
Date:           2021-04-29
ID:             RUSTSEC-2021-0059
URL:            https://rustsec.org/advisories/RUSTSEC-2021-0059
Dependency tree:
aesni 0.6.0

Crate:          block-cipher-trait
Version:        0.6.2
Warning:        unmaintained
Title:          crate has been renamed to `block-cipher`
Date:           2020-05-26
ID:             RUSTSEC-2020-0018
URL:            https://rustsec.org/advisories/RUSTSEC-2020-0018
Dependency tree:
block-cipher-trait 0.6.2

Crate:          net2
Version:        0.2.37
```

```
Warning:          unmaintained
Title:          `net2` crate has been deprecated; use `socket2` instead
Date:            2020-05-01
ID:              RUSTSEC-2020-0016
URL:              https://rustsec.org/advisories/RUSTSEC-2020-0016
Dependency tree:
net2 0.2.37

Crate:            rust-crypto
Version:          0.2.36
Warning:          unmaintained
Title:          rust-crypto is unmaintained; switch to a modern alternative
Date:            2016-09-06
ID:              RUSTSEC-2016-0005
URL:              https://rustsec.org/advisories/RUSTSEC-2016-0005
Dependency tree:
rust-crypto 0.2.36

Crate:            stream-cipher
Version:          0.3.2
Warning:          unmaintained
Title:          crate has been renamed to `cipher`
Date:            2020-10-15
ID:              RUSTSEC-2020-0058
URL:              https://rustsec.org/advisories/RUSTSEC-2020-0058
Dependency tree:
stream-cipher 0.3.2

Crate:            aes
Version:          0.3.2
Warning:          yanked
Dependency tree:
aes 0.3.2

Crate:            aes-soft
Version:          0.3.3
Warning:          yanked

Crate:            aesni
Version:          0.6.0
Warning:          yanked
```

```
Crate:          block-cipher-trait
Version:        0.6.2
Warning:         yanked

Crate:          crossbeam-deque
Version:        0.7.3
Warning:         yanked

Crate:          crossbeam-deque
Version:        0.8.0
Warning:         yanked

Crate:          stream-cipher
Version:        0.3.2
Warning:         yanked
```

5.2 The pricing of withdrawal fees is lack of flexibility [weaknesses]

The withdrawal fee is fixed at 5 USDT or equivalent tokens. In the case of network congestion, the fee may be lower than the actual gas cost, which will cause completeWithdrawals to consume additional fees and cause economic losses to the project party.

5.3 Lack of a risk control mechanism for spam transactions [medium-risk]

The first 50 transfers are free of handling fees per day, and 2U equivalent tokens will be charged for those exceeding 50 transfers, but malicious users can bypass the 50 free transfer limit by continuously deriving accounts. Since each transaction will add additional commission fees for commitBlock operations, causing economic losses to the project, it is necessary to add a mechanism to prevent spam transactions in risk control.

5.4 Risk of fake recharge attack [weakness]

The beneficiary needs to wait for Layer1 to confirm the transfer is successful. If the transaction is not finally confirmed, it may fail and roll back.

5.5 Transaction Replay Risk[medium-risk]

When a transaction fails, its status on the second-layer network is PACKAGED and has been packaged into the block, but the corresponding account nonce has not increased, and there is a certain risk. We usually assume that a signed transaction has been made public. The current execution of this signed transaction has failed, but it may be successful in the future. This transaction can be replayed without the trader’s knowledge.

6 Fix Log

7 Conclusion

Audit result: Fixing

Audit No. : BCA002109140001

Audit date: September 14, 2021

Audit team: SlowMist security team

Summary conclusion: All problems have been reported and are waiting for the project team to fix.

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.