Parallel Security Audit Report

Contents

1. Executive Summary

On June 01, 2021, the SlowMist security team start security audit for Parallel, 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 "black, grey box lead, white box assists" to conduct a complete security test on the project in the way closest to the real attack.

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.

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. Project Background (Context)

The main types of security audit include:

No.	Audit category	Subclass	Audit result
1	Code static check	-	-
2	Encrypted signature security	Random number generation algorithm audit	PASSED
		Private key storage audit	PASSED
		Cryptographic component call audit	PASSED
		Hash intensity audit	PASSED
		Transaction malleability attack audit	PASSED
3	Account and transaction	Transaction verification audit	PASSED

	model security	Transaction replay audit	PASSED
		"False Top-up" audit	PASSED
4	DeFi logical security	-	PASSED

(other unknown security vulnerabilities are not included in the scope of responsibility of this audit)

3. Coverage

3.1 Target Code and Revision

Source	https://github.com/parallel-finance/parallel
Version	a223cd7910af3540048b58f958fea5b784876468
Type	Blockchain base on Substrate v0.9.3.
Platform	Rust

4. Risk Analysis

4.1 Encrypted signature security

4.1.1 Random number generation algorithm audit

Checked at the substrate layer.

- substrate/client/cli/src/commands/generate.rs

```
let mnemonic = Mnemonic::new(words, Language::English);
```

This is a Rust implementation of the bip39 standard for Bitcoin HD wallet mnemonic phrases.

4.1.2 Private key storage audit

There is no wallet module.

4.1.3 Cryptographic component call audit

Hash function:

*sha2

*keccak

*blake2

*xxhash

Signature function:

*ed25519(Schnorr)

*sr255519(Schnorrkel)

*secp256k1

Address format:

Adopt SS58 address format

No error calls were found.

4.1.4 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.

4.1.5 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.
Parallel uses the ed25519 algorithm to sign, in the algorithm design of ED25519, by using a cryptographic hash function to replace the pseudo-random number generator, it avoids the security problems that the users of the signature algorithm use because the random number generator used is not random enough. In addition to the generation of the private key, the implementation of ED25519 has completely departed from the dependence on the random number generator, avoiding the leakage and security problems of the key due to the randomization problem.
No error calls were found.
Reference：https://en.bitcoinwiki.org/wiki/Transaction_Malleability

4.2 Account and transaction model security

4.2.1 Transaction verification audit

Checked at the substrate layer.

4.2.2 Transaction replay audit

Checked at the substrate layer.

- parallel/runtime/parallel/src/lib.rs

```
/// The SignedExtension to the basic transaction logic.
pub type SignedExtra = (
frame_system::CheckSpecVersion<Runtime>,
frame_system::CheckTxVersion<Runtime>,
frame_system::CheckGenesis<Runtime>,
frame_system::CheckEra<Runtime>,
frame_system::CheckNonce<Runtime>,
frame_system::CheckWeight<Runtime>,
pallet_transaction_payment::ChargeTransactionPayment<Runtime>,
);
```

Transaction eras are used to protect against transaction replay attacks in the event that an account is reaped and its (replay-protecting) nonce is reset to zero.

https://substrate.dev/docs/en/knowledgebase/getting-started/glossary#transaction-era

4.2.3 "False Top-up" audit

There is no similar EOS Hard Failed mechanism.

There is no similar EVM return fail mechanism.

There is no mechanism similar to USDT OP_RETURN.

There is no similar Ripple deliver_amount field.

All relevant fields of event need to be verified when recharging and entering the account.

Vulnerability reference:

https://mp.weixin.qq.com/s/fKINfZLW65LYaD4qO-21nA

https://mp.weixin.qq.com/s/3cMbE6p_4qCdVLa4FNA5-A

4.3 DeFi security

Check the audit findings for details.

4.4 Code static check

Modules	Rustsec	State consistency	Fail rollback	Unit test	Numeric overflow
pallets/liquid-staking	0 issue	0 issue	0 issue	mostly	0 issue
pallets/liquidation	0 issue	0 issue	0 issue	lack	0 issue
pallets/loans	0 issue	0 issue	0 issue	mostly	1 issue
pallets/prices	0 issue	0 issue	0 issue	mostly	0 issue
runtime/heiko	0 issue	0 issue	0 issue	lack	0 issue
runtime/parallel	0 issue	0 issue	0 issue	lack	0 issue
runtime/vanilla	0 issue	0 issue	0 issue	lack	0 issue
other	1 issue	0 issue	0 issue	-	0 issue

5. Findings

Vulnerability distribution:

Critical vulnerabilities	0	
High-risk vulnerabilities	0	
Medium-risk vulnerablities	0	
Low-risk vulnerabilities	1	■

Weaknesses	2	■■
Enhancement Suggestions	1	■
Total	4	
■Code static check ■DeFi Security ■Account and transaction model security ■Encrypted signature security ■Other		

5.1 Need to upgrade the module [low-risk]

```
ID:          RUSTSEC-2021-0067
Crate:       cranelift-codegen
Version: 0.71.0
Date:        2021-05-21
URL:         https://rustsec.org/advisories/RUSTSEC-2021-0067
Title:       Memory access due to code generation flaw in Cranelift module
Solution: upgrade to >= 0.73.1 OR >= 0.74
Dependency tree:
cranelift-codegen 0.71.0
```

Fixed in: https://github.com/parallel-finance/parallel/pull/210

5.2 Numeric overflow[enhancement]

- parallel/pallets/loans/src/lib.rs

```
let total_reserves_new = total_reserves - reduce_amount;
//...snip code...
let total_reserves_new = total_reserves + add_amount;
```

It is recommended to use `checked_add/checked_sub` to prevent numerical overflow.

Fixed in: https://github.com/parallel-finance/parallel/pull/241

5.3 Lack of bounds checking[weakness]

`mint_amount` has no boundary limit, it is recommended to enhance.

Fixed in: https://github.com/parallel-finance/parallel/pull/258

5.4 Oracle price feed risk[weakness]

Due to the lack of time parameter, if the price is not feed in time, the price may be inaccurate.

- parallel/pallets/prices/src/lib.rs

```
pub enum Event<T: Config> {
/// Set emergency price. \[currency_id, price_detail\]
SetPrice(CurrencyId, PriceWithDecimal),
/// Reset emergency price. \[currency_id\]
ResetPrice(CurrencyId),
}
```

Feedback: At present, the source of price feeding is controlled by authority and credible. The range of trustworthiness includes the accuracy and real-time of the price, that is, outdated prices will not be sent to the chain, but the price feeding transaction sent is through operational on the chain. Transaction level guarantees will be packaged in real time, that is, the current block.

6. Conclusion

Audit result: PASSED

Audit No. : BCA002106180001

Audit date: June 18, 2021

Audit team: SlowMist security team

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

7. 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.

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

Github
https://github.com/slowmist