SLOWMIST

DeFiChain Security Audit Report

1. Executive Summary

On Sep.11, 2020, the SlowMist security team received the DeFiChain team's security audit application for DeFiChain, 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 wether 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 vulnerablities	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)

2.1 Project Introduction

Project website: https://defichain.com/

Project source code: https://github.com/DeFiCh/ain

Coin symbol: DFI

Audit version: v1.0.2

2.2 Scope of Audit

The main types of security audit include:

No.	Audit Category	Audit Result

1	Code Compliance Audit	PASS
2	Random Number Generation Algorithm Audit	PASS
3	Keystore Audit	PASS
4	Cryptographic Component Call Audit	PASS
5	Encryption Strength Audit	PASS
6	Length Extension Attack Audit	PASS
7	Transaction Malleability Attack Audit	PASS
8	Replay Attack Audit	PASS
9	Top-up Program Audit	PASS
10	RPC Permission Audit	PASS

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

3. Code Overview

3.1 Infrastructure

DeFiChain developed based on the open source **Bitcoin release-v0.18.1**, using PoS consensus algorithm, The main function changes are as follows：

(1). Replace “Bitcoin” words to “DeFiChain”；

(2). The consensus protocol has been changed from PoW to PoS, masternodes produce block for the chain, which has better performance;

(3). Change chain parameters.

3.2 Code Compliance Audit

Fork open source blockchain source code will cause problems such as replay attacks and node address pool pollution. We conduct relevant security compliance assessments for this.

- ain/src/chainparams.cpp

```
/**
* The message start string is designed to be unlikely to occur in normal data.
* The characters are rarely used upper ASCII, not valid as UTF-8, and produce
* a large 32-bit integer with any alignment.
*/
pchMessageStart[0] = 0xf9;
pchMessageStart[1] = 0xbe;
pchMessageStart[2] = 0xb4;
pchMessageStart[3] = 0xd9;
nDefaultPort = 8555;
nPruneAfterHeight = 100000;
m_assumed_blockchain_size = 240;
m_assumed_chain_state_size = 3;
```

DeFiChain P2P protocol message structure’ s magic value is `F9BEB4D9`, which is the same with Bitcoin, it will cause peer pool pollution.

3.3 Random Number Generation Algorithm Audit

Generating random numbers based on /dev/urandom is security enough.

- ain/src/random.cpp

```
void GetStrongRandBytes(unsigned char* buf, int num) noexcept { ProcRand(buf, num,
RNGLevel::SLOW); }
//SlowMist//...snip...
static void ProcRand(unsigned char* out, int num, RNGLevel level)
{
// Make sure the RNG is initialized first (as all Seed* function possibly need hwrand to
be available).
RNGState& rng = GetRNGState();

assert(num <= 32);

CSHA512 hasher;
switch (level) {
case RNGLevel::FAST:
```

```
SeedFast(hasher);
break;
case RNGLevel::SLOW:
SeedSlow(hasher);
break;
case RNGLevel::SLEEP:
SeedSleep(hasher, rng);
break;
}

// Combine with and update state
if (!rng.MixExtract(out, num, std::move(hasher), false)) {
// On the first invocation, also seed with SeedStartup().
CSHA512 startup_hasher;
SeedStartup(startup_hasher, rng);
rng.MixExtract(out, num, std::move(startup_hasher), true);
}

// For anything but the 'fast' level, feed the resulting RNG output (after an additional
hashing step) back into OpenSSL.
if (level != RNGLevel::FAST) {
unsigned char buf[64];
CSHA512().Write(out, num).Finalize(buf);
RAND_add(buf, sizeof(buf), num);
memory_cleanse(buf, 64);
}
}
```

3.4 Keystore Audit

The wallet password strength has not been verified. Weak passwords such as `123456` can be used in the test, which can be easily crack.

3.5 Cryptographic Component Call Audit

No error calls were found.

3.6 Encryption Strength Audit

Weak hash functions such as md5 and sha1 are not used.

3.7 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.

3.8 Transaction Malleability Attack Audit

Bip-66/Bip-62 had solve this problem.

Vulnerability reference: https://en.bitcoinwiki.org/wiki/Transaction_Malleability

3.9 Replay Attack Audit

Based on the UTXO model, there is no replay problem on the chain, the same UTXO does not exist on different chains, and transactions cannot be replayed.

3.10 Top-up Program Audit

There is not false top-up vulnerability found on Bitcoin chain.

3.11 RPC Permission Audit

RPC has the dumpprivkey and sign function, the node keep RPC port closed by default, make sure do not open it.

Vulnerability reference: https://mp.weixin.qq.com/s/Kk2lsoQ1679Gda56Ec-zJg

4. Audit Result

4.1 High-risk vulnerabilities

- P2P protocol message structure's magic value is `F9BEB4D9`, which is the same with Bitcoin, it will cause peer pool pollution.
  (Fixed in: https://github.com/DeFiCh/ain/pull/36)

4.2 Exchange Security Summary

- Recharge entry needs to detect all relevant fields in the transaction and receipt structure, and reconcile it with the total account balance in real time. If an abnormality occurs, it needs to be manually checked before processing the entry to prevent "false top-up attacks".

4.3 Conclusion

Audit result: PASS

Audit No. : BCA002009180001

Audit date: September 18, 2020

Audit team: SlowMist security team

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

5. 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.