



Smart Contract Security Audit Report

[2021]



Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2021.08.02, the SlowMist security team received the Larix team's security audit application for Larix Obsolete, 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 party 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.

Level	Description
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:

Reentrancy Vulnerability

Replay Vulnerability

Reordering Vulnerability

Short Address Vulnerability

Denial of Service Vulnerability

Transaction Ordering Dependence Vulnerability

Race Conditions Vulnerability

Authority Control Vulnerability

Integer Overflow and Underflow Vulnerability

TimeStamp Dependence Vulnerability

Unsafe External Call Audit

Design Logic Audit

Scoping and Declarations Audit

Forged account attack Audit

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Arbitrary permission initialization of lend/oracle contract	Authority Control Vulnerability	Medium	Fixed
N2	Flash loan repayment detection bypass	Reentrancy Vulnerability	Critical	Fixed
N3	Process_flash_loan forged account risk	Forged account attack	Critical	Fixed
N4	Process_reserve forged account risk	Forged account attack	Critical	Fixed
N5	process_borrow_obligation_liquidity host_fee transfer target is not verified	Forged account attack	Critical	Fixed

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

Audit version:

SHA256(larix-audit-souce.rar)= d59d09ecf5efe3ce3062708b8052a83dd86d87320a76a0c56ca8ce64d82ec730

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

lending accounts check				
Function Name	Lamport check	Owner/Key check	Signer check	ProgramID check
process_init_lending_market	o	4/8	x	o
process_init_reserve	o	11/15	o	o
process_set_lending_market_owner	-	-	-	-
process_refresh_reserve	x	4/4	x	o
process_deposit_reserve_liquidity	x	9/10	x	o
process_redeem_reserve_collateral	x	9/10	x	o
process_init_obligation	o	5/6	o	o
process_refresh_obligation	x	4/4	x	o
process_deposit_obligation_collateral	x	9/10	o	o
process_withdraw_obligation_collateral	x	9/9	o	o
process_borrow_obligation_liquidity	x	10/11	o	o

lending accounts check				
process_repay_obligation_liquidity	x	7/8	x	o
process_liquidate_obligation	x	11/12	x	o
process_flash_loan	x	6/9	x	o
process_init_mining	o	4/4	o	o
process_deposit_mining	x	8/10	o	o
process_withdraw_mining	x	8/9	o	o
process_claim_mining_mine	x	7/9	o	o
process_claim_obligation_mine	x	6/8	o	o
process_claim_owner_fee	x	6/7	o	o
process_config_instruction->process_market	x	2/3	o	o
process_config_instruction->process_reserve	x	5/6	o	o

oracle accounts check				
Function Name	Lamport check	Owner/Key check	Signer check	ProgramID check
process_init_oracle	x	2/2	x	o
process_init_price_account	o	4/5	o	o
process_submit	x	5/5	o	o
process_set_submit_authority	x	2/2	o	o

o: Yes, x: No, -:Ignored

4.3 Vulnerability Summary

[N1] [Medium] Arbitrary permission initialization of lend/oracle contract

Category: Authority Control Vulnerability

Content

Anyone can initialize the lend/oracle contract, which may lead to the illegal use of the contract, and malicious users may use the officially deployed Program to conduct fraudulent activities.

Solution

Status

Fixed; Only allow official initialization

[N2] [Critical] Flash loan repayment detection bypass

Category: Reentrancy Vulnerability

Content

After the attacker calls `process_flash_loan` to borrow, he uses the borrowed funds to recharge to the contract. In this way, the flash loan will detect that the funds have been returned during the repayment check, which leads to the success of the flash loan, but the funds are not actually returned. Instead, the attacker get a deposit position is established, and the attacker can withdraw this fund at any time, thereby stealing all the funds in the fund pool.

Solution

Status

Fixed; `reentry_lock` has been added to prevent hacker reentry attacks

[N3] [Critical] Process_flash_loan forged account risk

Category: Forged account attack**Content**

Code location: larix-lending/src/processor.rs

```
fn process_flash_loan(  
    program_id: &Pubkey,  
    liquidity_amount: u64,  
    accounts: &[AccountInfo],  
)
```

If the owner or key of the `reserve_info` account is not verified, the attacker may attack the contract by maliciously constructing the data stored in the account.

Solution**Status**

Fixed; Verified the owner of the `reserve_info` account

[N4] [Critical] Process_reserve forged account risk**Category: Forged account attack****Content**

Code location: larix-lending/src/config/config_process.rs

```
fn process_reserve(program_id: & Pubkey, accounts: &  
[AccountInfo], reserve_type: ConfigReserveType)
```

If the owner or key of the `reserve_info` account is not verified, the attacker may attack the contract by maliciously constructing the data stored in the account.

Solution

Verify the owner or key of the `reserve_info` account

Status

Fixed; Verified the owner of the `reserve_info` account

[N5] [Critical] process_borrow_obligation_liquidity host_fee transfer target is not verified**Category: Forged account attack****Content**

Code location: `larix-lending/src/processor.rs`

```
if let Ok(host_fee_receiver_info) = next_account_info(account_info_iter) {
    if host_fee > 0 {
        owner_fee = owner_fee
            .checked_sub(host_fee)
            .ok_or(LendingError::MathOverflow)?;

        spl_token_transfer(TokenTransferParams {
            source: source_liquidity_info.clone(),
            destination: host_fee_receiver_info.clone(),
            amount: host_fee,
            authority: lending_market_authority_info.clone(),
            authority_signer_seeds,
            token_program: token_program_id.clone(),
        })?;
    }
}
```

The owner or key of the `host_fee_receiver_info` account is not verified, and the user can steal `host_fee` by specifying `host_fee_receiver_info`.

Solution

Verify the owner or key of the `host_fee_receiver_info` account

Status

Fixed; Verified the key of the `host_fee_receiver_info` account

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0x002108200001	SlowMist Security Team	2021.08.02 - 2021.08.20	Passed

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 4 critical risk, 1 medium risk. All the 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.



Official Website
www.slowmist.com



E-mail
team@slowmist.com



Twitter
[@SlowMist_Team](https://twitter.com/SlowMist_Team)



Github
<https://github.com/slowmist>