Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.10.08, the SlowMist security team received the team's security audit application for Iloop-contract, 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.
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
- 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
- Account substitution attack Audit
- Malicious Event Log Audit

3 Project Overview

3.1 Project Introduction

ILoop Protocol is a decentralized lending platform on the Solana blockchain, designed for secure and efficient leverage and capital optimization. ILoop stands out as an Automated DeFi protocol that integrates Lending, and Looping into a unified and secure DeFi product suite.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing event record	Malicious Event Log Audit	Suggestion	Fixed
N2	Redundant subtraction operation	Others	Suggestion	Fixed
N3	Mint address is missing the token_program constraint.	Others	Suggestion	Fixed
N4	Risks of excessive privilege	Authority Control Vulnerability Audit	Medium	Acknowledged
N5	Preemptive Initialization	Race Conditions Vulnerability	Low	Fixed
N6	Price manipulation risk in the invoke_inf_to_sol function	Design Logic Audit	High	Fixed

4 Code Overview

4.1 Contracts Description

https://github.com/ILoopfinance06/Iloop-contract

Commit: a3b02c80e8db5413a08a4c315445a4df8e2628d4

Review commit: d51b3c92c69c7eda2ab3799fff7dbe62d0fc9702

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

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

iloop-sc			
Function Name	Account check coverage	Auth Signer	Params Check
close_reserve	12/12	owner	0/0
deposit	12/12	lender	1/1
flash_borrow	10/10	borrower	1/1
flash_repay	10/10	borrower	2/2
init_lending_market	3/3	owner	0/0
init_obligation	6/6	owner	1/1
init_reserve	9/9	owner	1/1
init_user_metadata	3/3	owner	1/1
liquidate	17/17	liquidator	1/1
redeem	12/12	lender	1/1
refresh_reserve_price	7/7	owner	0/0
update_lending_market	3/3	owner	2/2
update_reserve_collateral	10/10	owner	0/0
update_reserve_config	8/8	owner	12/12
withdraw	13/13	lender	1/1
borrow	10/10	borrower	1/1
close_lending_market	3/3	owner	0/0

iloop-sc			
repay	10/10	borrower	1/1
supply	12/12	lender	1/1

4.3 Vulnerability Summary

[N1] [Suggestion] Missing event record

Category: Malicious Event Log Audit

Content

The changes to the following key parameters have not been logged with corresponding events.

- programs/iloop-sc/src/instructions/update_lending_market.rs

```
update_lending_market
```

- programs/iloop-sc/src/instructions/update_reserve_collateral.rs

```
update_reserve_collateral
```

- programs/iloop-sc/src/instructions/update_reserve_config.rs

```
update_reserve_config
```

Solution

Record the corresponding event.

Status

Fixed

[N2] [Suggestion] Redundant subtraction operation

Category: Others

Content

In the `calculate_borrow_rate function`, `u0` is always 0, so `u1.saturating_sub(u0)` is a redundant subtraction. Subtracting 0 does not change the value of u1.

- programs/iloop-sc/src/state/reserve.rs

```
pub fn calculate_borrow_rate(&self, utilization_rate: u64, min_lst_apy: u64) -> u64 {
        let u0 = 0;
        let u1 = self.optimal_utilization_rate_1;
        let u2 = self.optimal_utilization_rate_2;
        if utilization_rate <= u1 {
            let slope = (min_lst_apy.saturating_sub(self.margin_rate) as u128)
                .checked_mul(PRECISION as u128).unwrap()
                .checked_div(u1.saturating_sub(u0) as u128).unwrap(); *//@SlowMist *
            slope.checked_mul(utilization_rate.saturating_sub(u0) as u128).unwrap()
                .checked_div(PRECISION as u128).unwrap() as u64
        } else if utilization_rate <= u2 {
        ....
 }
            )
        }
    }
```

Solution

Verify whether the value of u0 aligns with the design intent. If it is confirmed to be a meaningless operation, the subtraction can be removed.

Status

Fixed

[N3] [Suggestion] Mint address is missing the token_program constraint.

Category: Others

Content

The constraint `token::token_program = collateral_token_program` is missing.

- programs/iloop-sc/src/instructions/withdraw.rs line:225

```
#[account(
      mut,
      seeds = [b"collateral_mint", reserve.key().as_ref()],
      bump
  )]
  pub collateral_mint: Box<Account<'info, Mint>>,
```

- programs/iloop-sc/src/instructions/close_reserve.rs line:109

```
#[account(
      mut,
      seeds = [b"collateral_mint", reserve.key().as_ref()],
      bump
  )]
  pub collateral_mint: Box<Account<'info, Mint>>,
```

- programs/iloop-sc/src/instructions/deposit.rs line:150

```
#[account(
      mut,
      seeds = [b"collateral_mint", reserve.key().as_ref()],
      bump
  )]
```

```
  pub collateral_mint: Box<Account<'info, Mint>>,
```

- programs/iloop-sc/src/instructions/redeem.rs line:160

```
#[account(
      mut,
      seeds = [b"collateral_mint", reserve.key().as_ref()],
      bump
  )]
  pub collateral_mint: Box<Account<'info, Mint>>,
```

- programs/iloop-sc/src/instructions/supply.rs line:128

```
#[account(
      mut,
      seeds = [b"collateral_mint", reserve.key().as_ref()],
      bump
  )]
  pub collateral_mint: Box<Account<'info, Mint>>,
```

Solution

Add the constraint token::token_program = collateral_token_program.

Status

Fixed

[N4] [Medium] Risks of excessive privilege

Category: Authority Control Vulnerability Audit

Content

In the protocol, the `lending_market owner` plays a crucial role, with the ability to update prices, interest rates,

and other key settings. If the owner's private key is compromised, it could have severe consequences for the entire protocol.

```
lending_market owner can update_lending_market

lending_market owner can update_reserve_collateral

lending_market owner can update_reserve_config
```

Solution

In the short term, transferring owner ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. The authority involving user funds should be managed by the community, and the authority involving emergency contract suspension can be managed by the EOA address. This ensures both a quick response to threats and the safety of user funds.

Status

Acknowledged

[N5] [Low] Preemptive Initialization

Category: Race Conditions Vulnerability

Content

The `init_lending_market` function can be called by anyone, which creates the risk of malicious calls setting parameters beyond expectations.

- programs/iloop-sc/src/instructions/init_lending_market.rs

```
pub fn init_lending_market(
    ctx: Context<InitLendingMarket>,
) -> Result<()> {
    let lending_market = &mut ctx.accounts.lending_market.load_init()?;


    lending_market.owner = ctx.accounts.owner.key();
    lending_market.min_lst_apy = 0;
    lending_market.max_age_price_seconds = MAX_AGE_PRICE_SECONDS;
    lending_market.reserves_count = 0;
```

```
    Ok(())
}
```

Solution

Restrict the caller to ensure that the creation is executed by the intended address.

Status

Fixed

[N6] [High] Price manipulation risk in the invoke_inf_to_sol function

Category: Design Logic Audit

Content

In the `invoke_inf_to_sol` price calculation function, the key calculation variables are tied to real-time data from the pool. This poses a potential issue, as there is a risk of malicious manipulation during the price update. For instance, before the price update transaction is executed, someone could manipulate the token balance in the pool to set a malicious price.

- programs/iloop-sc/src/util/inf_sol_price_calculator.rs

```
pub fn invoke_inf_to_sol<'c: 'info, 'info>(
    inf_mint: &AccountInfo<'info>,
    inf_amt: u64,
    remaining_accounts: &'c [AccountInfo<'info>],
) -> Result<U64ValueRange> {
    let pool_state = &remaining_accounts[1];
    require!(
        pool_state.key() == INF_POOL_STATE,
        InfSolValueCalculatorError::InvalidPoolState,
    );
    let inf_mint = Mint::try_deserialize(&mut inf_mint.data.borrow().as_ref())?;
    let start_total_sol_value = pool_state.total_sol_value().unwrap();
    let sol_value = (inf_amt as u128).checked_mul(start_total_sol_value as
u128).unwrap()
        .checked_div(inf_mint.supply as u128).unwrap() as u64; // sol_value = inf_amt
* total_sol_value / inf_supply

    Ok(U64ValueRange::single(sol_value))
}
```

Solution

It is recommended to use Pyth or other well-known oracles to fetch the price.

Status

Fixed; This part of the logic has been removed.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002410140001	SlowMist Security Team	2024.10.08 - 2024.10.14	Medium Risk

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 1 high risk, 1 medium risk, 1 low risk, 3 suggestion vulnerabilities.

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

Github

https://github.com/slowmist