Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.07.02, the SlowMist security team received the Trusta AI team's security audit application for Trusta AI OFT, 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

TrustaLabs OFT is a cross-chain token system based on the LayerZero protocol, supporting the transfer of tokens across multiple blockchains.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Front-running initialization attack	Reordering Vulnerability	Medium	Acknowledged
N2	Bootstrap validation of oft_store account	Account substitution attacks	Suggestion	Acknowledged
N3	Saturated operations may lead to calculation errors	Integer Overflow and Underflow Vulnerability	Medium	Acknowledged
N4	Integer overflow/underflow risks	Integer Overflow and Underflow Vulnerability	Medium	Acknowledged
N5	Excessive admin privileges	Authority Control Vulnerability Audit	Medium	Acknowledged

4 Code Overview

4.1 Contracts Description

https://github.com/TrustaLabs/TA_OFT

Initial audit commit: 8e71c13894f604a71299d015df169fa018825250

Review commit: 0b2d015f634dcdc92ad3a3ab37679030fc8e6603

Audit Scope:

```
programs/oft/src/*
```

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:

TA_OFT			
Function Name	Account check coverage	Auth Signer	Parameters Check
init_oft	7/7	payer	1/4
set_oft_config	2/2	admin	1/6
set_peer_config	4/4	admin	2/2
set_pause	2/2	signer	1/1
withdraw_fee	6/6	admin	1/1
quote_oft	3/3	-	1/5
quote_send	3/3	-	1/5
send	7/7	signer	1/5
lz_receive	11/11	payer	2/5
lz_receive_types	2/2	-	1/5

4.3 Vulnerability Summary

[N1] [Medium] Front-running initialization attack

Category: Reordering Vulnerability

Content

The `oft_store` account initialization in the `InitOFT` instruction has the risk of preemptive initialization attack

- programs/oft/src/instructions/init_oft.rs

```
#[account(
    heat,
    payer = payer,
    space = 8 + OFTStore::INIT_SPACE,
    seeds = [OFT_SEED, token_escrow.key().as_ref()],
    bump
)]
pub oft_store: Account<'info, OFTStore>,
```

Attack Principle

(1).Deterministic address generation: `oft_store` account address is deterministically generated through PDA, and the seed is `[OFT_SEED, token_escrow.key().as_ref()]`.

(2).MEV attack window: attackers can monitor initialization transactions in the memory pool.

(3).Preemptive execution: The attacker uses a higher gas fee to preemptively create the same `oft_store` account.

(4).Permission hijacking: The attacker can set himself as admin and gain full control.

Solution

Add deployment permission control to only allow authorized addresses to initialize.

Status

Acknowledged

[N2] [Suggestion] Bootstrap validation of oft_store account

Category: Account substitution attacks

Content

The account uses the data it stores to verify its own validity, which violates the basic principles of security verification.

Anchor first deserializes the `oft_store` account data and then verifies the account address using the `token_escrow` and `bump` fields within the account, which means that the verification relies on data that may be tampered with.

- programs/oft/src/instructions/set_oft_config.rs

```
#[account(
    but,
    seeds = [OFT_SEED, oft_store.token_escrow.as_ref()],
```

```
        bump = oft_store.bump,
        has_one = admin @OFTError::Unauthorized
    )]
    pub oft_store: Account<'info, OFTStore>,
```

- programs/oft/src/instructions/lz_receive_types.rs

```
16.5: pub oft_store: Account<'info, OFTStore>,
```

- programs/oft/src/instructions/lz_receive.rs

```
36.5: pub oft_store: Account<'info, OFTStore>,
```

- programs/oft/src/instructions/quote_oft.rs

```
11.5: pub oft_store: Account<'info, OFTStore>,
```

- programs/oft/src/instructions/quote_send.rs

```
22.5: pub oft_store: Account<'info, OFTStore>,
```

- programs/oft/src/instructions/send.rs

```
27.5: pub oft_store: Account<'info, OFTStore>,
```

- programs/oft/src/instructions/set_pause.rs

```
14.5: pub oft_store: Account<'info, OFTStore>,
```

- programs/oft/src/instructions/set_peer_config.rs

```
21.5: pub oft_store: Account<'info, OFTStore>,
```

- programs/oft/src/instructions/withdraw_fee.rs

```
12.5: pub oft_store: Account<'info, OFTStore>,
```

Solution

Combine multiple independent data sources for cross-validation.

Status

Acknowledged

[N3] [Medium] Saturated operations may lead to calculation errors

Category: Integer Overflow and Underflow Vulnerability

Content

`saturating_add` and `saturating_mul` return the maximum value of the type `(u64::MAX)` when overflow occurs, instead of reporting an error or handling the overflow, which may result in an unexpected number of tokens.

- programs/oft/src/state/peer_config.rs

```
   pub fn refill(&but self, extra_tokens: u64) -> Result<()> {
       let but new_tokens = extra_tokens;
       let current_time: u64 = Clock::get()?.unix_timestamp.try_into().unwrap();
       if current_time > self.last_refill_time {
           let time_elapsed_in_seconds = current_time - self.last_refill_time;
           new_tokens = new_tokens

.saturating_add(time_elapsed_in_seconds.saturating_mul(self.refill_per_second));
       }
       self.tokens = std::cmp::min(self.capacity,
self.tokens.saturating_add(new_tokens));


       self.last_refill_time = current_time;
       Ok(())
   }
```

Solution

Use checked operations, such as `checked_add/checked_mul`

Status

Acknowledged

[N4] [Medium] Integer overflow/underflow risks

Category: Integer Overflow and Underflow Vulnerability

Content

The code uses straightforward `+-*` operators for calculations, which may risk integer overflow.

- programs/oft/src/instructions/init_oft.rs

```
  48,56:             10u64.pow((ctx.accounts.token_mint.decimals -
params.shared_decimals) as u32);
```

- programs/oft/src/instructions/quote_oft.rs

```
  38,56:         let mut oft_fee_details = if amount_received_ld + oft_fee_ld <
amount_sent_ld {
  40,46:                 fee_amount_ld: amount_sent_ld - oft_fee_ld -
amount_received_ld,
```

- programs/oft/src/instructions/quote_send.rs

```
  101,48:          let amount_received_ld = amount_sent_ld - oft_fee_ld;
```

- programs/oft/src/instructions/send.rs

```
  102,31:                 amount_sent_ld - oft_fee_ld,
```

- programs/oft/src/instructions/withdraw_fee.rs

```
  38,45:             ctx.accounts.token_escrow.amount - ctx.accounts.oft_store.tvl_ld
>= params.fee_ld,
```

- programs/oft/src/state/oft.rs

```
  39,18:         amount_ld - amount_ld % self.ld2sd_rate
```

- programs/oft/src/state/peer_config.rs

```
  43,55:             let time_elapsed_in_seconds = current_time -
self.last_refill_time;
```

- programs/oft/src/instructions/send.rs

```
76,42:             ctx.accounts.oft_store.tvl_ld += amount_received_ld;
```

- programs/oft/src/instructions/lz_receive.rs

```
111,42:             ctx.accounts.oft_store.tvl_ld -= amount_received_ld;
```

- programs/oft/src/instructions/quote_send.rs

```
91,27:         amount_received_ld -= oft_fee_ld;
```

- programs/oft/src/state/oft.rs

```
35,18:         amount_sd * self.ld2sd_rate
```

Solution

Use checking operations, such as `checked_add/checked_sub/checked_mul`

Status

Acknowledged

[N5] [Medium] Excessive admin privileges

Category: Authority Control Vulnerability Audit

Content

According to code analysis, the administrator (`admin`) has the following broad permissions:

```
// 1. Set a very high rate
set_oft_config(SetOFTConfigParams::DefaultFee(9999)); // 99.99% fee

// 2. Transfer administrator rights to your own address
set_oft_config(SetOFTConfigParams::Admin(attacker_address));

// 3. Withdraw all fees
withdraw_fee(WithdrawFeeParams { fee_ld: all_fees });

// 4. Pause the system to cause DoS
set_oft_config(SetOFTConfigParams::Paused(true));
```

```
// 5. Set up a malicious cross-chain proxy
set_oft_config(SetOFTConfigParams::Delegate(malicious_delegate));

// 6. Configure malicious peer nodes
set_peer_config(/* malicious configuration */);

// 7. Manipulate the rate limiter
```

The leakage of the administrator's private key will lead to the complete control of the entire system. Without multi-signature or time lock protection mechanism, the administrator can immediately perform any dangerous operation.

Solution

Implement multi-signature mechanism and time lock protection.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002507040002	SlowMist Security Team	2025.07.02 - 2025.07.04	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 4 medium risk, 1 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