Ploutoz Finance

Smart Contract Audit Report

Prepared By: Sutee Sudprasert

ThaiChain, Thailand

May 29th, 2021

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to: (i) cybersecurity vulnerabilities and issues in the smart contract source code analysed, the details of which are set out in this report, (Source Code); and (ii) the Source Code compiling, deploying and performing the intended functions. In order to get a full view of our findings and the scope of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Zenoic Proprietary Limited trading as "Iosiro" and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Iosiro) owe no duty of care towards you or any other person, nor does Iosiro make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Iosiro hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Iosiro hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Iosiro, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

Document Properties

Client	Ploutoz Finance
Title	Smart Contract Audit Report
Repository	https://github.com/PTZFinance/Ploutoz-Lending
Commit	5ab26eed1dfe94903ffb34671027eb9d0881ee53
Author	Sutee Sudprasert
Auditors	Sutee Supdrasert, Rati Montreewat, Thanarat Kuawattanaphan
Reviewed By	Thanarat Kuawattanaphan
Approved By	Dom Charoenyos
Classification	Confidential

Introduction

Thai Chain was contracted by Ploutoz Finance to conduct an audit of smart contracts. The report presents the findings of the security assessment of the smart contracts and its code review conducted between May 20th, 2021 - May 29th, 2021.

Scope

The scope of the project is smart contracts in the repository:

https://github.com/PTZFinance/Ploutoz-Lending (5ab26ee)

The files in the scope of auditing are LoanTokenLogicStandard.sol, FairLaunch.sol, and PloutozToken.sol.

Executive Summary

The Ploutoz project is a lending platform on Binance Smart Chain that was forked from BzX. However, the version that has been used is the prior version before being audited by Certik and Peckshield. So they have inherited the issues that have been found by those two lending smart contract auditing companies. The client should consider fixing those known issues too.

Our team performed an analysis of code functionality and manual audit. We found 1 critical and 1 lowest issue during the audit.

Notice: the audit scope is limited and does not include all files in the repository. Though, reviewed contracts are secure, we may not guarantee the secureness of contracts that are not in the scope.

Severity Definitions

Severity Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to asset loss or data manipulations.
High	High-level vulnerabilities have a significant impact on smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to asset loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution.
Lowest / Coding Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

Audit Overview

Critical

1. The `borrow` function in the LoanTokenLogicStandard contract does not have validations for checking the `msg.sender` is not the owner of the loan (`borrower`) when applied to the existing loan (`loanId != 0`). This issue allows an arbitrary user to impersonate a borrower for borrowing digital assets to a given receiver if the loan still has margin left.

   Exploiting Scenario

   1. Bob borrowed 1000 USDT by using 2000 BUSD as a collateral token. So Bob still has margin left 16.67% (more 333.4 USDT can be borrowed from this loan).
   2. An exploiter can find Bob's `loanId` from `event Borrow` and use the `loanId` to check that Bob still has margin left by using the public `loans` function of `pfiProtocol` contract.
   3. An exploiter called the function `borrow` with Bob's `loanId` and used an exploiter address as a `receiver`. With only a small amount of collateral token (BUSD), an attacker can get 333.4 USDT from Bob's loan.

Recommendation: Add sanity checks against the `msg.sender` and the `borrower` when `loanId` is not 0.

```
function borrow(
    bytes32 loanId,
    uint256 withdrawAmount,
    uint256 initialLoanDuration,
    uint256 collateralTokenSent,
    address collateralTokenAddress,
    address borrower,
    address receiver,
    bytes memory /*loanDataBytes*/)
    public
    payable
    nonReentrant
    returns (uint256, uint256)
{
    ...

    // ensures authorized use of existing loan
    require(loanId == 0 || msg.sender == borrower, "13");

    require(msg.value == 0 || msg.value == collateralTokenSent, "7");
    require(collateralTokenSent != 0 || loanId != 0, "8");

    ...
}
```

Status: This issue has been addressed by validating the `msg.sender` in the beginning of `borrow()` when `loanId != 0` in this commit: 3f6264c.

High

No high issues were found.

Medium

No medium issues were found.

Low

No low issues were found.

Lowest / Coding Style / Best Practice

1. The computation of `interestOwedPerDay` in the `_avgBorrowInterestRate` function of the `LoanTokenLogicStandard` contract should apply multiplication before division.

Recommendation: To improve the precision of the arithmetic operations, multiplication should be applied before division.

```
function _avgBorrowInterestRate(
        uint256 assetBorrow)
        internal
        view
        returns (uint256)
{
    if (assetBorrow != 0) {
        (uint256 interestOwedPerDay,) = _getAllInterest();
        return interestOwedPerDay
                .mul(3650**20)
                .div(assetBorrow);
    }
}
```

Conclusion

One critical severity issue has been found and fixed in the commit: 3f6264c. According to the assessment, the Customer's smart contracts are well-secured.