# Code Security Assessment

For ShorterFinance

**27**

**October**

**2022**

# Overview

## Project Summary

- Name: ShorterFinance
- Version: 3.0
- Language: solidity
- Codebase: https://github.com/IPILabs/shorter-v1
- Audit Range: contracts/ShorterBone.sol; contracts/v1/PoolGuardianImpl.sol; contracts/v1/TradingHubImpl.sol; contracts/v1/AuctionHallImpl.sol; contracts/v1/VaultButlerImpl.sol; contracts/v1/StrPoolProviderImpl.sol; contracts/v1/StrPoolTraderImpl.sol; contracts/v1/TreasuryImpl.sol

# Project Dashboard

## Application Summary

| | |
|---|---|
| **Name** | ShorterFinance |
| **Version** | v3.0 |
| **Type** | Solidity |
| **Date** | 10/27 |
| **Logs** | 10/12; 10/25; 10/27 |

## Vulnerability Summary

| | | |
|---|---|---|
| **Total High-Severity issues** | 2 | |
| **Total Medium-Severity issues** | 0 | |
| **Total Low-Severity issues** | 0 | |
| **Total informational issues** | 12 | |
| **Total** | 14 | |

## Contact


- E-mail: support@salusec.io

## Risk Level Description

| | |
|---|---|
| **High Risk** | The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputation or serious financial implications for client and users. |
| **Medium Risk** | The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact. |
| **Low Risk** | The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is lowmpact in view of the client's business circumstances. |
| **Informational** | The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth. |

# Content

# Introduction

## 1.1 About SALUS

Salus Security is an all-rounded blockchain security company. With rich experiences in traditional and blockchain security, we are born to solve some of the most complex security issues in the industry and make security services accessible for all. Our smart contract auditing service is equipped with an automated tool and expert services. Every project needs an invincible shield to achieve long-term success; with complete coverage from traditional to blockchain, Salus Security is what you need. We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

## 1.2 Audit Breakdown

Objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

# Findings

## 2.1 Key Findings

| Severity | Title | Category | Status |
|---|---|---|---|
| High | Incorrect Use of Variable | Coding Practice | Resolved |
| High | Incorrect function call | Business Logic | Resolved |
| Informational | Pragma usage | Coding Practice | Unresolved |
| Informational | Variable declaration | Coding Practice | Unresolved |
| Informational | Contract clarification | Coding Practice | Resolved |
| Informational | Vulnerable function call | Coding Practice | Unresolved |
| Informational | Loop import | Coding Practice | Resolved |
| Informational | Address lacks zero-check | Coding Practice | Resolved |
| Informational | Redundant Import | Coding Practice | Resolved |
| Informational | Revert function clarification | Coding Practice | Resolved |
| Informational | Unchecked return value | Coding Practice | Unresolved |
| Informational | Centralization of Power | Coding Practice | Resolved |
| Informational | Missing Event Parameter | Coding Practice | Resolved |
| Informational | Optimization | Coding Practice | Partially-Resolved |

## 2.2 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues of the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit-based assessment cannot be considered

# Detailed Result

## Incorrect use of variable

- **Target Contract: StrPoolProviderImpl.sol**
- **Severity: High**
- **Category: Coding Practice**
- **File(s) affected: StrPoolProviderImpl.sol**

## Description

Incorrect use of variable, The variable “amount <= 100” should be “percent <= 100”

```
StrPoolProviderImpl.getWithdrawAmount(uint256,uint256).require(percent > 0 &&
amount <= 100, "StrPool: Invalid withdraw percentage");
```

## Incorrect function call

- **Target Contract: StrPoolProviderImpl.sol**
- **Severity: High**
- **Category: Coding Practice**
- **File(s) affected: StrPoolProviderImpl.sol**

## Description

The functions “burnAmount” has redundant “mul(userShare)” and causes the wrong calculation result.

```
StrPoolProviderImpl.getWithdrawAmount(uint256,uint256)


burnAmount =
userStakedTokenAmount[msg.sender].mul(userShare).mul(percent).div(1e20);
(contracts/v1/StrPoolProviderImpl.sol#140)
burnAmount =
userWrappedTokenAmount[msg.sender].mul(userShare).mul(percent).div(1e20);
(contracts/v1/StrPoolProviderImpl.sol#144)
```

## Recommendation

The function will be correct once the “mul(userShare)” part is removed.

# Information

## Pragma usage

- **Target Contract: All files**
- **Category: Coding Practice**
- **File(s) affected: All files**

```
pragma solidity 0.6.12;
```

Recommend using the most up-to-date version of compiler.

## Variable declaration

- **Target Contract: TradingHubImpl.sol**
- **Category: Coding Practice**
- **File(s) affected: TradingHubImpl.sol**

```
require(estimatePrice.mul(amount).mul(9) <
amountOutMin.mul(10**(uint256(19).add(pool.stakedTokenDecimals).sub(pool.stableTok
enDecimals))), "TradingHub: Slippage too large");
```

Recommend using variable declaration for 9 and 19 in the above function, which will be beneficial for future code maintenance. Without the variable declaration, it can be confusing for readers.

## Contract clarification

- **Target Contract: Pausable.sol**
- **Category: Coding Practice**
- **File(s) affected: any files that shares inheritance**

```
@openzeppelin/contracts/utils/Pausable.sol
/util/Pausable.sol
```

The above two contracts “Pausable.sol” co-exist. Be aware when calling in case of confusion

## Vulnerable function call

- **Target Contract: TradingHubImpl.sol; StrPoolTraderImpl.borrow**
- **Category: Coding Practice**
- **File(s) affected: TradingHubImpl.sol; StrPoolTraderImpl.borrow**

```
TradingHubImpl.sellShort (#62)
IStrPool(pool.strToken).borrow(dexCenter.isSwapRouterV3(swapRouter),
address(dexCenter), swapRouter, position, msg.sender, amount, amountOutMin,
path);

StrPoolTraderImpl.borrow (#25)
bytes memory data = delegateTo(dexCenter,
abi.encodeWithSignature("sellShort((bool,uint256,uint256,address,address,byt
es))", IDexCenter.SellShortParams({isSwapRouterV3: isSwapRouterV3, amountIn:
amountIn, amountOutMin: amountOutMin, swapRouter: swapRouter, to:
address(this), path: path})));
```

The above two functions are sophisticated and can potentially result in the risk of modifying StrPool status of “dexCenter.sellShort” .

## Loop Import

- **Target Contract: IPoolGuardian.sol; IPoolRewardModel.sol**
- **Category: Coding Practice**
- **File(s) affected: IPoolGuardian.sol; IPoolRewardModel.sol**

```
import “../IShorterBone.sol";
import “../../libraries/AllyLibrary.sol";
import “../../IShorterBone.sol";
import “./IRewardModel.sol";
import "../../../libraries/AllyLibrary.sol";
```

Redundant import result in loop import. Recommend removing redundant imports.

## Address lacks zero-check

- **Target Contract: PoolGuardianImpl.sol; StrPoolProviderImpl.sol; TradingHubImpl.sol; ShorterBone.sol; AuctionHallImpl.sol; VaultButlerImpl.sol; StrPoolTraderImpl.sol; TreasuryImpl.sol**
- **Category: Coding Practice**
- **File(s) affected: PoolGuardianImpl.sol; StrPoolProviderImpl.sol; TradingHubImpl.sol; ShorterBone.sol; AuctionHallImpl.sol; VaultButlerImpl.sol; StrPoolTraderImpl.sol; TreasuryImpl.sol**

```
ShorterBone.setTetherToken(address)._TetherToken

AuctionHallImpl.initialize(address,address,address,address,address,address,address,uint256,uint256)._dexCenter

AuctionHallImpl.initialize(address,address,address,address,address,address,address,uint256,uint256)._ipistrToken

AuctionHallImpl.setDexCenter(address).newDexCenter

PoolGuardianImpl.setWrapRouter(address).newWrapRouter

StrPoolProviderImpl.initialize(address,address,address,address,address,address,uint256,uint256,uint256)._creator

StrPoolProviderImpl.initialize(address,address,address,address,address,address,uint256,uint256,uint256)._tradingHub

TradingHubImpl.setDexCenter(address).IDexCenter(newDexCenter)

TradingHubImpl.setPriceOracle(address).IPriceOracle(newPriceOracle)

ShorterBone.constructor(address)._SAVIOR

PoolGuardianImpl.constructor(address)._SAVIOR
```

```
TradingHubImpl.constructor(address)._SAVIOR

AuctionHallImpl.constructor(address)._SAVIOR

VaultButlerImpl.constructor(address)._SAVIOR

StrPoolProviderImpl.constructor(address)._SAVIOR

StrPoolTraderImpl.constructor(address)._SAVIOR

TreasuryImpl.constructor(address)._SAVIOR
```

Recommend adding a zero check. For example: require(_SAVIOR != address(0),"receiver can't be zero address");

## Redundant import

- **Target Contract: VaultButlerImpl.sol; PoolRewardModelImpl.sol**
- **Category: Coding Practice**
- **File(s) affected: VaultButlerImpl.sol; PoolRewardModelImpl.sol**

```
import “@openzeppelin/contracts/token/ERC20/SafeERC20.sol”;
import “@openzeppelin/contracts/utils/EnumerableSet.sol”;
import “../libraries/OracleLibrary.sol”;

import “@openzeppelin/contracts/utils/EnumerableSet.sol”;
```

## Revert function clarification

- **Target Contract: StrPoolProviderImpl.sol**
- **Category: Coding Practice**
- **File(s) affected: StrPoolProviderImpl.sol**

```
StrPoolProviderImpl.getWithdrawAmount
StrPoolProviderImpl._deposit
StrPoolProviderImpl._withdraw
StrPoolProviderImpl._transferWithHarvest

Recommended:
revert("StrPool getWithdrawAmount: Insufficient balance");
revert("StrPool _deposit: Insufficient balance");
revert("StrPool _withdraw: Insufficient balance");
revert("StrPool _transferWithHarvest: Insufficient balance");
```

Recommend changing the revert functions in the above functions into the below format for future bug tracing

## Unchecked return value

- **Target Contract: TreasuryImpl.sol; TradingHubImpl.sol**
- **Category: Coding Practice**
- **File(s) affected: TreasuryImpl.sol; TradingHubImpl.sol**

```
TreasuryImpl.removeOwner(address)
TreasuryImpl._setOwner(address)
TradingHubImpl.sellShort(uint256,uint256,uint256,address,bytes)
```

The return values of the above functions are never checked.

## Centralization of power

- **Target Contract: TreasuryImpl.sol; TradingHubImpl.sol; Shorterbone.sol; AuctionHallImpl.sol; PoolGuardianImpl.sol; VaultButlerImpl.sol**
- **Category: Coding Practice**
- **File(s) affected: TreasuryImpl.sol; TradingHubImpl.sol; Shorterbone.sol; AuctionHallImpl.sol; PoolGuardianImpl.sol; VaultButlerImpl.sol**

Keeper manager has too much power. Recommend using MultiSigWallet and / DAO to replace Keeper manager.
The client has changed keeper manager to DAO, However, the realization of DAO is beyond the auditing scope.

## Missing event parameter

- **Target Contract: PoolGuardianImpl.sol**
- **Category: Coding Practice**
- **File(s) affected: PoolGuardianImpl.sol**

```
PoolGuardianImpl.setStateFlag(uint256,PoolStatus)
```

PoolStatus is missing the event from “Liquidating; recover; genesis”

## Optimization (Resolved)

- **Target Contract: TreasuryImpl.sol**
- **Category: Coding Practice**
- **File(s) affected: TreasuryImpl.sol**

```
uint256 _threshold = threshold;
```

No need to use temporary variable. Directly use thereshold.

## Optimization (Resolved)

- Target Contract: ShorterBone.sol
- Category: Coding Practice
- File(s) affected: ShorterBone.sol

```
shorterBone.lockedMintable
```

No actual usage of the above function, hence can be deleted.

## Optimization (Resolved)

- Target Contract: PoolGuardianImpl.sol; StrPoolProviderImpl.sol; TradingHubImpl.sol; ShorterBone.sol; AuctionHallImpl.sol; VaultButlerImpl.sol; StrPoolTraderImpl.sol; TreasuryImpl.sol
- Category: Coding Practice
- File(s) affected: PoolGuardianImpl.sol; StrPoolProviderImpl.sol; TradingHubImpl.sol; ShorterBone.sol; AuctionHallImpl.sol; VaultButlerImpl.sol; StrPoolTraderImpl.sol; TreasuryImpl.sol

```
ShorterBone.addTokenWhiteList(address,address,uint256)

AuctionHallImpl.setDexCenter(address)

PoolGuardianImpl.setStrPoolImplementations(bytes4[],address)

PoolGuardianImpl.queryPools(address,IPoolGuardian.PoolStatus)

StrPoolTraderImpl.withdrawRemnantAsset(address)

TradingHubImpl.getPositionsByAccount(address,ITradingHub.PositionState)

TradingHubImpl.getPositionsByPoolId(uint256,ITradingHub.PositionState)

TradingHubImpl.getPositionsByState(ITradingHub.PositionState)

TradingHubImpl.setDexCenter(address)

TradingHubImpl.setPriceOracle(address)
```

```
TreasuryImpl.initialize(address[],uint256)

VaultButlerImpl.priceOfLegacy(address)

VaultButlerImpl.initialize(address,address,address,address)
```

Recommend changing the above functions to external.

## Optimization

- **Target Contract: All files**
- **Category: Coding Practice**
- **File(s) affected: All files**

Recommend using the latest “require-error” to save gas.

## Optimization (Resolved)

- **Target Contract: TradingHubImpl.sol**
- **Category: Coding Practice**
- **File(s) affected: TradingHubImpl.sol**

```
TradingHubImpl.checkPath(bytes,address,address) (contracts/v1/
TradingHubImpl.sol#27-34) is never used and should be removed
```

Above function is never used, please remove.

SALUS