SALUS SECURITY

OCT 2022

CODE SECURITY ASSESSMENT

SHORTER FINANCE

Overview

Project Summary

- Name: Shorter Finance
- Language: Solidity
- Repository: 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	Shorter Finance
Version	v3
Type	Solidity
Dates	Oct 27 2022
Logs	Oct 12 2022; Oct 25 2022; Oct 27 2022

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' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a 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 low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications 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

1.3 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 with 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.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Incorrect use of variable	High	Coding Practice	Resolved
2	Incorrect function call	High	Coding Practice	Resolved
3	Pragma usage	Informational	Coding Practice	Acknowledged
4	Variable declaration	Informational	Coding Practice	Acknowledged
5	Contract clarification	Informational	Coding Practice	Resolved
6	Vulnerable function call	Informational	Coding Practice	Acknowledged
7	Loop Import	Informational	Coding Practice	Resolved
8	Address lacks zero-check	Informational	Coding Practice	Resolved
9	Redundant import	Informational	Coding Practice	Resolved
10	Revert function clarification	Informational	Coding Practice	Resolved
11	Unchecked return value	Informational	Coding Practice	Acknowledged
12	Centralization of power	Informational	Coding Practice	Resolved
13	Missing event parameter	Informational	Coding Practice	Resolved
14	Optimization	Informational	Coding Practice	Partially Resolved

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Incorrect use of variable

Severity: High	Category: Coding Practice
Target: - 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");
```

2. Incorrect function call

Severity: High	Category: Coding Practice
Target: - StrPoolProviderImpl.sol	

Description

```
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

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

Recommendation

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

2.3 Informational Findings

3. Pragma usage

Severity: Informational	Category: Coding Practice
Target: - all	

Description

```
pragma solidity ^0.6.12
```

Recommendation

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

4. Variable declaration

Severity: Informational	Category: Coding Practice
Target: - TradingHubImpl.sol	

Description

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

Recommendation

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.

5. Contract clarification

Severity: Informational	Category: Coding Practice
Target: - all	

Description

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

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

6. Vulnerable function call

Severity: Informational	Category: Coding Practice
Target: - TradingHubImpl.sol - StrPoolTraderImpl.sol	

Description

```
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,bytes))",
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”.

7. Loop Import

Severity: Informational	Category: Coding Practice
Target: - IPoolGuardian.sol - IPoolRewardModel.sol	

Description

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

Redundant imports result in loop imports.

Recommendation

Recommend removing redundant imports.

8. Address lacks zero-check

Severity: Informational	Category: Coding Practice

Target:
- PoolGuardianImpl.sol
- StrPoolProviderImpl.sol
- TradingHubImpl.sol
- ShorterBone.sol
- AuctionHallImpl.sol
- VaultButlerImpl.sol
- StrPoolTraderImpl.sol
- TreasuryImpl.sol

Description

```
ShorterBone.setTetherToken(address)._TetherToken
AuctionHallImpl.initialize(address,address,address,address,address,address,address,uint2
56,uint256)._dexCenter
AuctionHallImpl.initialize(address,address,address,address,address,address,address,uint2
56,uint256)._ipistrToken
AuctionHallImpl.setDexCenter(address).newDexCenter
PoolGuardianImpl.setWrapRouter(address).newWrapRouter
StrPoolProviderImpl.initialize(address,address,address,address,address,address,uint256,u
int256,uint256)._creator
StrPoolProviderImpl.initialize(address,address,address,address,address,address,uint256,u
int256,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
```

Recommendation

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

9. Redundant import

Severity: Informational	Category: Coding Practice
Target: - VaultButlerImpl.sol - PoolRewardModelImpl.sol	

Description

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

10. Revert function clarification

Severity: Informational	Category: Coding Practice
Target: - StrPoolProviderImpl.sol	

Description

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

Redundant imports result in loop imports.

Recommendation

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

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

11. Unchecked return value

Severity: Informational	Category: Coding Practice
Target: - TradingHubImpl.sol - TreasuryImpl.sol	

Description

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

The return values of the above functions are never checked.

Recommendation

Recommend removing redundant imports.

12. Centralization of power

Severity: Informational	Category: Coding Practice
Target: - TreasuryImpl.sol - TradingHubImpl.sol - Shorterbone.sol - AuctionHallImpl.sol - PoolGuardianImpl.sol - VaultButlerImpl.sol	

Description

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.

13. Missing event parameter

Severity: Informational	Category: Coding Practice
Target: - PoolGuardianImpl.sol	

Description

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

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

14. Optimization

Severity: Informational	Category: Coding Practice
Target: - TreasuryImpl.sol	

Description

```
uint256 _threshold = threshold;
```

No need to use temporary variables.

```
shorterBone.lockedMintable
```

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

```
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)
```

```
TradingHubImpl.checkPath(bytes,address,address)
```

(contracts/v1/TradingHubImpl.sol#27-34) is never used and should be removed

Recommendation

Directly use threshold. Recommend changing the above functions to external. Recommend changing the above functions to external. Recommend using the latest “require-error” to save gas.