Smart Contract Security Audit Report

For

PointDex

Date Issued: Jun 16, 2023

Version: v1.0

Confidentiality Level: Public

Contents

1 Abstract

This report was prepared for PointDex smart contract to identify issues and vulnerabilities in its smart contract source code. A thorough examination of PointDex smart contracts was conducted through timely communication with PointDex, static analysis using multiple audit tools and manual auditing of their smart contract source code.

The audit process paid particular attention to the following considerations.

- A thorough review of the smart contract logic flow
- Assessment of the code base to ensure compliance with current best practice and industry standards
- Ensured the contract logic met the client's specifications and intent
- Internal vulnerability scanning tools tested for common risks and writing errors
- Testing smart contracts for common attack vectors
- Test smart contracts for known vulnerability risks
- Conduct a thorough line-by-line manual review of the entire code base

As a result of the security assessment, issues ranging from critical to informational were identified. We recommend that these issues are addressed to ensure a high level of security standards and industry practice. The recommendations we made could have better served the project from a security perspective.

- Enhance general coding practices to improve the structure of the source code.
- Provide more comments for each function to improve readability.
- Provide more transparency of privileged activities once the agreement is in place.

2 Overview

2.1 Project Summary

Project Name	Shield Security
Start date	Apr 25, 2023
End date	Jun 16, 2023
Contract type	DeFi
Language	Solidity
File	ExchangeController.sol, ExchangePair.sol, ExchangeRouter.sol, Migrations.sol

2.2 Audit Scope

File	SHA256
ExchangeController.sol	0FEFC2D5D93AE0603D9D5665135D5CD4DD2AEA92D9747000E9AD303A6D46FC24
ExchangePair.sol	AC9AB499FC00DCE1D1E3E44835894DEA6A70A99149FA66D9ED2062BB339255F8
ExchangeRouter.sol	066E0AD6E6958C7B32259760E52A9892675EBDBA5FC8DD2708075DD457F628CC
Migrations.sol	D3D14AF80B22C4C08F72C0F4B2A0B6B7457592C40DC7E4F9375E37978C80056D

3 Project contract details

3.1 Contract Overview

ExchangeController Contract

The ExchangeController contract is mainly invoked by project privileged roles and belongs to the management contract. It mainly sets transaction pair information and creates transaction pairs. In addition, the contract provides the function of obtaining transaction pair information.

ExchangePair Contract

The main function of the ExchangePair contract is to match user orders. The main methods include canceling orders, user order information, and pending orders.

ExchangeRouter Contract

The ExchangeRouter contract is mainly used for the entrance of Token mutual exchange, including user order data, order cancellation, and handling fees can also be obtained through this contract.

Migrations Contract

The Migrations contract specifies the last_completed_migration variable. The contract has only one setCompleted method, which is mainly used to modify the last_completed_migration variable, and this method can only be called by the privileged role owner.

OrderLinkedList Contract

The main user of OrderLinkedList implements the function logic of the ExchangePair contract, and the main logic implementation is to add order addOrder and delete order removeOrder.

POTToken Contract

The POTToken and USDTToken contracts use the ERC20 standard method, the main function is to issue tokens, and release the token mint to the address of the deployer.

WrappedCoin Contract

The WrappedCoin contract uses the ERC20 standard method, and its main functions are mortgage and withdrawal. Users use native tokens to mortgage to obtain project tokens, and vice versa.

3.2 Code Overview

ExchangeController Contract

Function Name	Visibility	Modifiers
initialize	External	initializer
setPairConfig	External	onlyRole(MANAGER_ROLE)
createPair	External	onlyRole(MANAGER_ROLE)
getPair	External	-
pairInfo	External	-

ExchangePair Contract

Function Name	Visibility	Modifiers
initialize	External	initializer
_update	Internal	-
sync	Public	lock
putaway	External	lock
_matchmaking	Internal	-
priceAmount	External	-
userOnOrderInfo	External	-
cancelOrder	External	lock
safeTransferToken	Internal	-

ExchangeRouter Contract

Function Name	Visibility	Modifiers
ExchangeRouter	External	-
swapTokensForTokens	External	ensure(deadline)
priceAmount	External	-
userOnOrderInfo	External	-
cancelOrder	External	-

Migrations Contract

Function Name	Visibility	Modifiers
setCompleted	Public	restricted

4 Audit results

4.1 Key messages

ID	Title	Severity	Status
01	redundant code	Informational	confirm
02	code used multiple times	Informational	confirm
03	methods that may not work	Low	confirm
04	call logic is not executed	Low	confirm
05	Possible Malicious Trading Pairs	Low	confirm

4.2 Audit details

4.2.1 redundant code

ID	Severity	Location	Status
01	Informational	Migrations.sol: 1, 16	confirm

Description

The contract has an owner administrator privilege role, and the owner administrator privilege role can set the last_completed_migration variable, but through the entire project, the specific function of the contract has not been found; it is recommended to determine the main function of this function, if it is meaningless, it is recommended to delete this code.

Code location:

```
2   pragma solidity >=0.4.22 <0.9.0;
3
4   contract Migrations {
5     address public owner = msg.sender;
6     uint public last_completed_migration;
7
8     modifier restricted() {
9       require(msg.sender == owner,"This function is restricted to the contract's owner");
10      _;
11    }
12
13    function setCompleted(uint completed) public restricted {
14      last_completed_migration = completed;
15    }
16  }
```

Recommendation

It is recommended to remove unused contract code.

Status

confirm.

4.2.2 code used multiple times

ID	Severity	Location	Status
02	Informational	ExchangeRouter.sol: 129, 130	confirm

Description

All methods in the ExchangeRouter contract judge the incoming token path and pair. Here, it is recommended to write these two judgment conditions into the judgment method separately, and then call them through the judgment method to avoid repeated code usage.

Code location:

```
83      function priceAmount(
84          address[2] memory path,
85          uint8 length
86      ) external view returns (uint256[] memory, uint256[] memory) {
87          address pair = controller.getPair(path[0], path[1]);
88          require(pair != address(0), "ExchangeRouter: No Pair");
89          return
90              path[0] == ITokenPair(pair).gold()
91                  ? ITokenPair(pair).priceAmount(length, true)
92                  : ITokenPair(pair).priceAmount(length, false);
93      }
94
95      function userOnOrderInfo(
96          address[2] memory path,
97          uint256[] memory orderIndexs
98      )
99          external
100         view
101         returns (
102             uint256[] memory,
103             uint256[] memory,
104             uint256[] memory,
105             uint256[] memory,
106             uint256[] memory
107         )
108     {
109         address pair = controller.getPair(path[0], path[1]);
110         require(pair != address(0), "ExchangeRouter: No Pair");
111         return
112             path[0] == ITokenPair(pair).gold()
113                 ? ITokenPair(pair).userOnOrderInfo(true, orderIndexs)
114                 : ITokenPair(pair).userOnOrderInfo(false, orderIndexs);
115     }
116
117
118     function cancelOrder(address[2] memory path, uint256 orderIndex) external {
119         address pair = controller.getPair(path[0], path[1]);
120         require(pair != address(0), "ExchangeRouter: No Pair");
121         if (path[0] == ITokenPair(pair).gold()) {
122             ITokenPair(pair).cancelOrder(true, orderIndex);
123         } else {
124             ITokenPair(pair).cancelOrder(false, orderIndex);
125         }
126     }
```

Recommendation

It is recommended to write these two judgment conditions into the judgment method separately, and then call them through the judgment method to avoid repeated use of repeated codes.

Status

confirm.

4.2.3 methods that may not work

ID	Severity	Location	Status
03	Low	ExchangePair.sol: 107, 114	confirm

Description

In the putaway method, the reserveGold value is updated through the sync and _update methods. When the sync method is called before the putaway method is called, and the funds carried here are not reserveGold and reserveToken, the calculation results of the above userInitBalance are all zero, so the following condition judgment will never will be established, and the putaway method call will not succeed.

Code location:

```
112     function _update() internal {
113         reserveGold = IERC20Upgradeable(gold).balanceOf(address(this));
114         reserveToken = IERC20Upgradeable(token).balanceOf(address(this));
115     }
116
117     function sync() public lock {
118         _update();
119     }
120
121
122     function putaway(
123         address[2] memory path,
124         uint256 price,
125         uint256 index,
126         address account
127     ) external payable lock returns (uint256, uint256, uint256) {
128
129         MatchmakingParams memory matchmakingPrams;
130         matchmakingPrams.user = account;
131         matchmakingPrams.userPrice = price;
132
133
134         (
135             matchmakingPrams.isBuy,
136             matchmakingPrams.reserveFee,
137             matchmakingPrams.chargeRate,
138             matchmakingPrams.chargeReceiver
139         ) = ITokenController(controller).pairInfo(address(this));
140
141         require(matchmakingPrams.isBuy, "closed");
142         require(msg.value >= matchmakingPrams.reserveFee, "Invalid Reserve");
143         matchmakingPrams.isBuy = (path[0] == gold ? true : false);
144         matchmakingPrams.userInitBalance = matchmakingPrams.isBuy
145             ? IERC20Upgradeable(gold).balanceOf(address(this)) - reserveGold
146             : IERC20Upgradeable(token).balanceOf(address(this)) - reserveToken;
147         require(matchmakingPrams.userInitBalance > 0, "Invalid amount");
148         matchmakingPrams.userBalance = matchmakingPrams.userInitBalance;
149
150         _update();
```

Recommendation

It is recommended to optimize the calculation of the value of the userInitBalance variable in the putaway method to avoid the value being controlled and causing security risks.

Status

confirm.

4.2.4 call logic is not executed

ID	Severity	Location	Status
04	Low	ExchangePair.sol: 190, 205	confirm

Description

When the _matchmaking method is executed, the initial sellOrderLst.list[0].next and buyOrderLst.list[1].prev are not greater than 1. So (data.iterator != 0 && data.iterator != 1) the judgment condition is not satisfied, so the content in the while statement is not executed, and then because the condition of (data.adverseAcquired > 0) is not satisfied, the statement if (data.adverseAcquired > 0) will not be executed, and the subsequent two conditional transfer fee conditions will not be established, and finally this method is meaningless.

Code location:

```
190     function _matchmaking(MatchmakingParams memory data) internal {
191
192         data.iterator = data.isBuy
193             ? sellOrderLst.list[0].next
194             : buyOrderLst.list[1].prev;
195         data.reserveGold = reserveGold;
196         data.reserveToken = reserveToken;
197
198         while (data.iterator != 0 && data.iterator != 1) {
199             OrderLinkedList.Order storage adverseOrder = data.isBuy
200                 ? sellOrderLst.list[data.iterator]
201                 : buyOrderLst.list[data.iterator];
202
203             if (
204                 data.adverseAcquired > 0 && data.adverse != adverseOrder.trader
205             ) {
206                 safeTransferToken(
207                     data.isBuy ? gold : token,
208                     data.adverse,
209                     data.adverseAcquired -
210                         (data.adverseAcquired * data.chargeRate) /
211                         1e12
212                 );
213                 if (data.isBuy) {
214                     data.reserveGold -= data.adverseAcquired;
215                 } else {
216                     data.reserveToken -= data.adverseAcquired;
217                 }
218                 data.adverseAcquired = 0;
219             }
```

Recommendation

It is recommended to strictly judge whether the conditions are feasible when setting variables and judging variable values.

Status

confirm.

4.2.5 Possible Malicious Trading Pairs

ID	Severity	Location	Status
05	Low	ExchangeController.sol: 50, 92	confirm

Description

The project transaction pair is set by the privileged role MANAGER_ROLE, but if the address is an EOA address, there is a greater security risk. When the privileged role is maliciously controlled, malicious transaction pairs may be created, and all the transaction pair parameters that have been set may be modified, causing the project to fail. normal operation. It is recommended that privileged roles use multisig and timelock contracts.

Code location:

```
50      function setPairConfig(
51          address pair,
52          bool isOpen,
53          uint256 chargeRate,
54          address chargeReceiver
55      ) external onlyRole(MANAGER_ROLE) {
56          pairConfigs[pair].isOpen = isOpen;
57          pairConfigs[pair].chargeRate = chargeRate;
58          pairConfigs[pair].chargeReceiver = chargeReceiver;
59      }
60
61      function createPair(
62          address gold,
63          address token,
64          uint256 chargeRate,
65          address chargeReceiver,
66          uint256 reserveFee
67      ) external onlyRole(MANAGER_ROLE) {
68          (address token0, address token1) = gold < token
69              ? (gold, token)
70              : (token, gold);
71          bytes32 pairHash = keccak256(abi.encodePacked(token0, token1));
72          require(pairs[pairHash] == address(0), "PAIR_EXISTED");
73
74          pairs[pairHash] = address(
75              new BeaconProxy(
76                  pairTemplate,
77                  abi.encodeWithSelector(
78                      ITokenPair.initialize.selector,
79                      gold,
80                      token,
81                      address(this),
82                      weth
83                  )
84              )
85          );
86          pairConfigs[pairs[pairHash]].isOpen = true;
87          pairConfigs[pairs[pairHash]].chargeRate = chargeRate;
88          pairConfigs[pairs[pairHash]].chargeReceiver = chargeReceiver;
89          pairConfigs[pairs[pairHash]].reserveFee = reserveFee;
90
91          emit CreatePair(pairs[pairHash], token0, token1);
92      }
```

Recommendation

It is recommended that privileged roles use multisig and timelock contracts.

Status

confirm.

5 Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

This report is issued in response to facts that occurred or existed prior to the issuance of this report, and liability is assumed only on that basis.
Shield Security cannot determine the security status of this program and assumes no responsibility for facts occurring or existing after the date of this report. The security audit analysis and other content in this report is based on documents and materials provided to Shield Security by the information provider through the date of the insurance report. in Shield Security's opinion. The information provided is not missing, falsified, deleted or concealed. If the information provided is missing, altered, deleted, concealed or not in accordance with the actual circumstances, Shield Security shall not be liable for any loss or adverse effect resulting therefrom. shield Security will only carry out the agreed security audit of the security status of the project and issue this report. shield Security is not responsible for the background and other circumstances of the project. Shield Security is not responsible for the background and other circumstances of the project.