Silver Consulting- PremiumPool review

Review Resources:

The project's repository was provided.

Auditor:

Anh Nguyen (Usua◎ Silver)

Table of Contents

Review Summary

PremiumPool

A prize savings protocol enabling investors to win by saving. Prizes are generated on the interest earned on deposited funds.

The dev branch of the PremiumPool Repo was reviewed over 4 days. Concretely, the following file was audited:

- src/DrawController.sol
- src/CharityFactory.sol
- src/LendingController.sol
- src/PremiumPool.sol
- src/PremiumPoolStorage.sol
- src/Ticket.sol

The contracts were reviewed from October 2 to October 6. The repository was under active development during the review, but the review was limited to one specific commit.

Scope

Code Repo

Commit

The commit reviewed was 074165358a8c9cd4383fbb2837df531b5116ac87. The review covered the entire repository at this specific commit but focused on the src/ directory.

The review is a code review to identify potential vulnerabilities in the code. The reviewers did not investigate security practices or operational security and assumed that privileged accounts could be trusted. The reviewers did not evaluate the security of the code relative to a standard or specification. The review may not have identified all potential attack vectors or areas of vulnerability.

Silver Consulting make no warranties regarding the security of the code and do not warrant that the code is free from defects. Silver Consulting does not represent nor imply to third party users that the code has been audited nor that the code is free from defects. By deploying or using the code, Fundraiser Finance and users agree to use the code at their own risk.

Code Evaluation Matrix

Category	Mark	Description
Access Control	Good	Access controls are applied where needed
Mathematics	Good	Solidity 0.8.13 is used, which provides overflow and underflow protect. Unchecked code was checked in this audit to yield no anomalies. No low-level bitwise operations are performed. There was no unusually complex math.
Complexity	Good	Clear and file structure and contract organization.
Libraries	Good	Only basic Open Zeppelin contracts such as IERC20, ERC20, Ownable, and Chainlink AggregatorV2Interface, VRFConsumerBaseV2, LinkTokenInterface are imported, no other external libraries are used. Fewer and simpler external dependencies is always a plus for security.
Code stability	Average	Changes were reviewed at a specific commit and the scope was not expanded after the review was started.
Documentation	Good	Comments existed in many places, and mostly clarified what the code did.
Monitoring	Good	Events were added to all important functions that modified state variables.
Testing and verification	Good	All tests were passing and test coverage was very expansive.

Findings Explanation

Findings are broken down into sections by their respective impact:

- Critical, High, Medium, Low impact
    - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements,
- Gas savings
    - Findings that can improve the gas efficiency of the contracts
- Informational
    - Findings including recommendations and best practices

High Findings

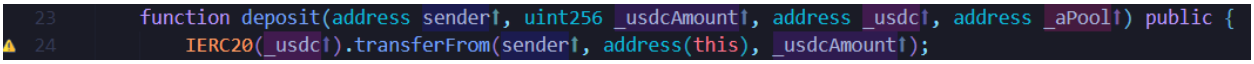
1. High - Arbitrary from in transferFrom

Proof of concept

Detect when `msg.sender` is not used as `from` in `transferFrom`.

Impact

Arbitrary `from` address in `transferFrom` in a public function allows exploitation.



Recommendation

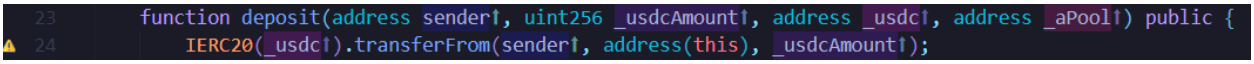
Use `msg.sender` as from in `transferFrom`.

2. High - Unchecked transfer

Proof of concept

The return value of an external transfer/transferFrom call is not checked

Impact



Recommendation

Use SafeERC20 or ensure that the transfer/transferFrom return value is checked.

Medium Findings

1. Medium - Dangerous strict equalities

Proof of concept

Use of strict equalities that can be easily manipulated by an attacker.

Impact

State variables modified after external call

```
62            if(ticket.balanceOf(msg.sender) == 0){
```

Recommendation

Don't use strict equality to determine if an account has enough Ether or tokens.

2. Medium - Unused return

Proof of concept

The return value of an external call is not stored in a local or state variable.

Impact

2a)

```
137         link.transferAndCall(address(coordinator), amount, abi.encode(subscriptionId));
```

2b)

```
25            IERC20(_usdc).approve(address(_aPool), _usdcAmount);
```

2c)

```
35            IAToken(aToken).approve(address(_aPool), _usdcAmount);
```

2d)

```
36            ILendingPool(_aPool).withdraw(address(_usdc), _usdcAmount, sender);
```

Recommendation

Ensure that all the return values of the function calls are used.

Low Findings

1. Low - Reentrancy vulnerabilities

Proof of concept

State variables modified after external call, which might act as a double call

Impact

test/PremiumPool.t.sol

2a)

```
59            vrfCoordinator = new VRFCoordinatorV2Mock(0, 0);
```

State variables written after external calls

```
57          forkId = vm.createSelectFork("https://polygon-mainnet.infura.io/v3/1fc7c7c3701c4083b769e561ae251f9a");
```

2b)

```
62          pool = new PremiumPool(usdc, aPoolAddrProv, aToken, address(vrfCoordinator), link, subscriptionId, keyhash);
63          draw = pool.draw();
```

State variables written after external calls

```
60            vrfCoordinator.createSubscription();
61            vrfCoordinator.fundSubscription(1, 7 ether);
```

2c)

```
65            ticket = pool.ticket();
```

State variables written after external calls

```
64            vrfCoordinator.addConsumer(subscriptionId, address(draw));
```

src/PremiumPool.sol

2d)

```
63                delete users[userIndex[msg.sender]];
64                userIndex[msg.sender] = 0;
```

State variables written after external calls

```
59          lending.withdraw(msg.sender, _usdcAmount, address(usdc), address(aToken), address(aPool));
60          ticket.burn(msg.sender, _usdcAmount);
```

Recommendation

Apply the [check-effects-interactions pattern](#).

2. Low - Reentrancy vulnerabilities

Proof of concept

Reentrancies leading to out-of-order events , which might lead to issues for third parties.

Impact

src/DrawController.sol

3a)

```
96              emit RandomnessRequested(requestId, currentDraw.drawId);
```

Event emitted after external calls

```
95              requestId = coordinator.requestRandomWords(keyHash, subscriptionId, 3, 50000, 1);
```

3b)

```
116                     emit WinnerElected(drawId, currentUser, prize);
```

Event emitted after external calls

```
115                     pool.mintTicket(currentUser, prize);
```

src/PremiumPool.sol

3c)

```
49              emit Deposit(msg.sender, _usdcAmount);
```

Event emitted after external calls

```
44          lending.deposit(msg.sender, _usdcAmount, address(usdc), address(aPool));
45
46          (bool success, ) = address(this).call(abi.encodeWithSignature("mintTicket(address,uint256)", msg.sender, _usdcAmount));
```

3d)

```
67            emit Withdraw(msg.sender, _usdcAmount);
```

Event emitted after external calls

```
59            lending.withdraw(msg.sender, _usdcAmount, address(usdc), address(aToken), address(aPool));
60            ticket.burn(msg.sender, _usdcAmount);
```

Recommendation

Apply the [check-effects-interactions pattern](#).

3. Low - Block timestamp

Proof of concept

Dangerous usage of `block.timestamp`.`block.timestamp` can be manipulated by miners.

Impact

```
145           if (block.timestamp>=deadline)
```

Recommendation

Avoid relying on `block.timestamp`.

Informational Findings

1. Informational - Incorrect versions of Solidity

Proof of concept

Using an old version prevents access to new Solidity security checks.

Impact

Pragma version^0.8.13 allows old versions in all files in src/ and test/

Recommendation

Use a simple pragma version such as 0.8.10. Consider using the latest version of Solidity for testing.

2. Informational - Low-level calls

Proof of concept

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact

src/PremiumPool.sol

```
46          (bool success, ) = address(this).call(abi.encodeWithSignature("mintTicket(address,uint256)", msg.sender, _usdcAmount));
```

Recommendation

Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

3. Informational - Conformance to Solidity naming conventions

Proof of concept

Solidity defines a naming convention that should be followed.

Impact

Parameters in DrawController

- _randomWords (src/DrawController.sol#102) is not in mixedCase
- _prize (src/DrawController.sol#124) is not in mixedCase
- _usdcDeposit (src/DrawController.sol#128) is not in mixedCase

Parameter LendingController

- ._usdcAmount (src/LendingController.sol#23) is not in mixedCase
- _usdc (src/LendingController.sol#23) is not in mixedCase
- aPool (src/LendingController.sol#23) is not in mixedCase
- _usdcAmount (src/LendingController.sol#33) is not in mixedCase
- _usdc (src/LendingController.sol#33) is not in mixedCase
- _aPool (src/LendingController.sol#33) is not in mixedCase

Parameter PremiumPool.deposit(uint256).

- _usdcAmount (src/PremiumPool.sol#38) is not in mixedCase
- _usdcAmount (src/PremiumPool.sol#56) is not in mixedCase
- _minter (src/PremiumPool.sol#112) is not in mixedCase
- _amount (src/PremiumPool.sol#112) is not in mixedCase

Parameter PremiumPoolTicket

- _minter (src/Ticket.sol#10) is not in mixedCase
- _amount (src/Ticket.sol#10) is not in mixedCase
- _account (src/Ticket.sol#14) is not in mixedCase
- _amount (src/Ticket.sol#14) is not in mixedCase

Parameter PremiumPoolTest

- _aliceAmount (test/PremiumPool.t.sol#98) is not in mixedCase
- _bobAmount (test/PremiumPool.t.sol#98) is not in mixedCase
- _charlieAmount (test/PremiumPool.t.sol#98) is not in mixedCase
- _usdcAmount (test/PremiumPool.t.sol#137) is not in mixedCase
- _aliceAmount (test/PremiumPool.t.sol#147) is not in mixedCase
- _bobAmount (test/PremiumPool.t.sol#147) is not in mixedCase
- _charlieAmount (test/PremiumPool.t.sol#147) is not in mixedCase
- _aliceAmount (test/PremiumPool.t.sol#216) is not in mixedCase
- _bobAmount (test/PremiumPool.t.sol#216) is not in mixedCase
- _charlieAmount (test/PremiumPool.t.sol#216) is not in mixedCase
- _randomNum (test/PremiumPool.t.sol#216) is not in mixedCase
- _usdcAmount (test/PremiumPool.t.sol#274) is not in mixedCase

Recommendation

Follow the Solidity naming convention.

4. Informational - Unused state variable

Proof of concept

Unused state variable.

Impact

DrawController.poolTicket (src/DrawController.sol#34) is never used in DrawController

Recommendation

Remove unused state variables.

Gas Findings

Gas - State variables that could be declared constant

Proof of concept

Constant state variables should be declared constant to save gas.

Impact

test > PremiumPool.t.sol

```
34      address usdc = address(0x2791Bca1f2de4661ED88A30C99A7a9449Aa84174);
35      address aPoolAddrProv = address(0xd05e3E715d945B59290df0ae8eF85c1BdB684744);
36      address aToken = address(0x1a13F4Ca1d028320A707D99520AbFefca3998b7F);
37      address link = address(0xb0897686c545045aFc77CF20eC7A532E3120E0F1);
38      bytes32 keyhash = bytes32(0x6e099d640cde6de9d40ac749b4b594126b0169747122711109c9985d47751f93);
39      /** */
40      /** POLYGON MUMBAI */
41      /* address usdc = address(0x0FA8781a83E46826621b3BC094Ea2A0212e71B23);
42      address aPoolAddrProv = address(0x178113104fEcbcD7fF8669a0150721e231F0FD4B);
43      address aToken = address(0x2271e3Fef9e15046d09E1d78a8FF038c691E9Cf9);
44      address link = address(0x326C977E6efc84E512bB9C30f76E30c160eD06FB);
45      bytes32 keyhash = bytes32(0x4b09e658ed251bcafeebbc69400383d49f344ace09b9576fe248bb02c003fe9f); */
46      /** */
47      uint64 subscriptionId = 1;
```

Recommendation

Add the `constant` attributes to state variables that never change.

Final remarks

With a focus on how contracts interact with each other and vulnerabilities with any external-call mechanics to the main PremiumPool.sol contract, I found nothing particularly worthy of note that was a critical exploit