Pooltogether: Smart Contract Audit & Code Review

Prepared for: Pooltogether
Date: March 6, 2024

Table of Contents

Pooltogether: Smart Contract Audit **0**
& Code Review **0**
Table of Contents 1
Disclaimer 2
Audit Information 2
Executive Summary 3
Scope 3
Security Definitions 3
Findings 4
M001 - The owner is a single point of failure and a centralization risk: 4
Inaccurate Balance Update on TransferFromM002 - Return values of transfer()/transferFrom() not checked: 4
M003 - Return values of approve() not checked: 5
M004 - Large transfers may not work with some ERC20 tokens: 6
M005 - Large approvals may not work with some tokens: 6
M006 - Return values of transfer/transferFrom not checked: 7
L001 - Missing checks for address(0x0) when assigning values to address state variables: 8
L002 - Use Ownable2Step rather than Ownable: 8
L003 - Array does not have a pop function: 9
L004 - Setters should have initial value check: 9
L005 - No limits when setting state variable amounts: 11
L006 - Allowed fees/rates should be capped by smart contracts: 11
L007 - The call abi.encodeWithSelector is not type safe: 12
L008 - Consider implementing two-step procedure for updating protocol addresses: 13
L009 - decimals() is not a part of the ERC-20 standard: 13
L010 - Governance functions should be controlled by time locks: 14
L011 - Missing checks for address(0x0) when updating address state variables: 14
L012 - The additions/multiplications may silently overflow because they're in unchecked blocks with no preceding value checks, which may lead to

unexpected results.: 15
L013 - approve()/safeApprove() may revert if the current approval is not zero: 16
Disclaimers 17
AuditBase Disclaimer 17
Technical Disclaimer 17

Disclaimer

This document might include private information regarding Pooltogether (the "Company") intellectual property and IT systems, as well as details on potential weaknesses and ways to exploit them.

After all vulnerabilities have been patched, the Company may decide whether to use the secret information in the report internally or to make it public.

Audit Information

Name	Pooltogether
Type of Contracts	NFT Lending
Platform	Multi-chain
Language	Solidity
Testing Methods	Architecture Review, Static Analysis, Functional Analysis, Manual Review
Website	https://github.com/code-423n4/2024-03-pooltogether
Date	Mar 6, 2024

Executive Summary

AuditBase has completed an audit for Pooltogether, an NFT lending protocol on Ethereum. The scope of this audit was limited to the smart contracts powering the NFT lending platform.

Scope

Deployed Contract	N/A
Documentation	https://github.com/code-423n4/2024-03-pooltogether
Unit Tests	N/A
Contracts	2024-03-pooltogether/pt-v5-vault/src/PrizeVaultFactory.sol

Security Definitions

Risk	Description
Critical	Critical flaws can result in the loss of assets or the alteration of data and are typically simple to exploit.
High	High-level vulnerabilities are challenging to attack, but they can have a big impact on how smart contracts work, like giving the public access to essential features.
Medium	Although medium-level vulnerabilities should be fixed, they cannot result in the loss of assets or the manipulation of data.
Low	Low-level flaws are typically caused by bits of unneeded, old code that don't have a big influence on the execution.

Code Scan Findings

M001 - The owner is a single point of failure and a centralization risk:

Having a single EOA as the only owner of contracts is a large centralization risk and a single point of failure. A single private key may be taken in a hack, or the sole holder of the key may become unable to retrieve the key when necessary. Consider changing to a multi-signature setup, or having a role-based authorization model.

Severity: Medium

Java

File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
735     function setClaimer(address _claimer) external onlyOwner {

742     function setLiquidationPair(address _liquidationPair) external onlyOwner {

753     function setYieldFeePercentage(uint32 _yieldFeePercentage) external onlyOwner {

759     function setYieldFeeRecipient(address _yieldFeeRecipient) external onlyOwner {
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L759:759

Inaccurate Balance Update on TransferFromM002 - Return values of transfer()/transferFrom() not checked:

Not all IERC20 implementations revert() when there's a failure in transfer()/transferFrom(). The function signature has a boolean return value and they indicate errors that way instead. By not checking the return value, operations that should have marked as failed, may potentially go through without actually making a payment.

Severity: Medium

```
Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVaultFactory.sol


118             IERC20(_vault.asset()).transferFrom(msg.sender, address(_vault), YIELD_BUFFER);
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVaultFactory.sol#L118:118

```
Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol


939                 _asset.transfer(_receiver, _assets);
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L939:939

```
Java
File: 2024-03-pooltogether/pt-v5-vault/src/TwabERC20.sol


101             twabController.transfer(_from, _to, SafeCast.toUint96(_amount));
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/TwabERC20.sol#L101:101

M003 - Return values of approve() not checked:

Not all IERC20 implementations revert() when there's a failure in approve(). The function signature has a boolean return value and they indicate errors that way instead. By not checking the return value, operations that should have marked as failed, may potentially go through without actually approving anything.

Severity: Medium

```
Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

862           _asset.approve(address(yieldVault), _assetsWithDust);

869             _asset.approve(address(yieldVault), 0);
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L869:869

M004 - Large transfers may not work with some ERC20 tokens:

Some IERC20 implementations (e.g UNI, COMP) may fail if the valued transferred is larger than uint96. Source

Severity: Medium

```
JavaScript
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

939             _asset.transfer(_receiver, _assets);
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L939:939

M005 - Large approvals may not work with some tokens:

Not all IERC20 implementations are totally compliant, and some (e.g UNI, COMP) may fail if the valued passed is larger than uint96. Source

Severity: Medium

Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
862           _asset.approve(address(yieldVault), _assetsWithDust);
```

```
869             _asset.approve(address(yieldVault), 0);
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L869:869

M006 - Return values of transfer/transferFrom not checked:

Not all IERC20 implementations revert when there's a failure in transfer/transferFrom. The function signature has a boolean return value and they indicate errors that way instead. By not checking the return value, operations that should have marked as failed, may potentially go through without actually making a payment

Severity: Medium

Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVaultFactory.sol

```
118           IERC20(_vault.asset()).transferFrom(msg.sender, address(_vault), YIELD_BUFFER);
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVaultFactory.sol#L118:118

Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
939             _asset.transfer(_receiver, _assets);
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L939:939

L001 - Missing checks for address(0x0) when assigning values to address state variables:

This issue arises when an address state variable is assigned a value without a preceding check to ensure it isn't address(0x0). This can lead to unexpected behavior as address(0x0) often represents an uninitialized address.

Severity: Low

```
Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

959         yieldFeeRecipient = _yieldFeeRecipient;
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L959:959

L002 - Use Ownable2Step rather than Ownable:

Ownable2Step and Ownable2StepUpgradeable prevent the contract ownership from mistakenly being transferred to an address that cannot handle it (e.g. due to a typo in the address), by requiring that the recipient of the owner permissions actively accept via a contract call of its own.

Severity: Low

```
Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

65    contract PrizeVault is TwabERC20, Claimable, IERC4626, ILiquidationSource, Ownable {
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L65:65

L003 - Array does not have a pop function:

Arrays without the pop operation in Solidity can lead to inefficient memory management and increase the likelihood of out-of-gas errors.

Severity: Low

```
Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVaultFactory.sol

120           allVaults.push(_vault);
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVaultFactory.sol#L120:120

L004 - Setters should have initial value check:

Setters should have initial value check to prevent assigning wrong value to the variable. Assginment of wrong value can lead to unexpected behavior of the contract.

Severity: Low

Java

File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
735     function setClaimer(address _claimer) external onlyOwner {
736         _setClaimer(_claimer);
737     }


742     function setLiquidationPair(address _liquidationPair) external onlyOwner {
743         if (address(_liquidationPair) == address(0)) revert LPZeroAddress();
744
745         liquidationPair = _liquidationPair;
746
747         emit LiquidationPairSet(address(this), address(_liquidationPair));
748     }


753     function setYieldFeePercentage(uint32 _yieldFeePercentage) external onlyOwner {
754         _setYieldFeePercentage(_yieldFeePercentage);
755     }


759     function setYieldFeeRecipient(address _yieldFeeRecipient) external onlyOwner {
760         _setYieldFeeRecipient(_yieldFeeRecipient);
761     }


947     function _setYieldFeePercentage(uint32 _yieldFeePercentage) internal {
948         if (_yieldFeePercentage > MAX_YIELD_FEE) {
949             revert YieldFeePercentageExceedsMax(_yieldFeePercentage, MAX_YIELD_FEE);
950         }
951         yieldFeePercentage = _yieldFeePercentage;
952         emit YieldFeePercentageSet(_yieldFeePercentage);
953     }


958     function _setYieldFeeRecipient(address _yieldFeeRecipient) internal {
959         yieldFeeRecipient = _yieldFeeRecipient;
960         emit YieldFeeRecipientSet(_yieldFeeRecipient);
961     }
```

L005 - No limits when setting state variable amounts:

It is important to ensure state variables numbers are set to a reasonable value.

Severity: Low

Java

File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
309         yieldBuffer = yieldBuffer_;
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L309:309

L006 - Allowed fees/rates should be capped by smart contracts:

Fees/rates should be required to be below 100%, preferably at a much lower limit, to ensure users don't have to monitor the blockchain for changes prior to using the protocol.

Severity: Low

Java

File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
753     function setYieldFeePercentage(uint32 _yieldFeePercentage) external onlyOwner {
754         _setYieldFeePercentage(_yieldFeePercentage);
755     }
```

```
759     function setYieldFeeRecipient(address _yieldFeeRecipient) external onlyOwner {
760         _setYieldFeeRecipient(_yieldFeeRecipient);
761     }


947     function _setYieldFeePercentage(uint32 _yieldFeePercentage) internal {
948         if (_yieldFeePercentage > MAX_YIELD_FEE) {
949             revert YieldFeePercentageExceedsMax(_yieldFeePercentage, MAX_YIELD_FEE);
950         }
951         yieldFeePercentage = _yieldFeePercentage;
952         emit YieldFeePercentageSet(_yieldFeePercentage);
953     }


958     function _setYieldFeeRecipient(address _yieldFeeRecipient) internal {
959         yieldFeeRecipient = _yieldFeeRecipient;
960         emit YieldFeeRecipientSet(_yieldFeeRecipient);
961     }
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L958:961

L007 - The call abi.encodeWithSelector is not type safe:

abi.encodeCall() has compiler type safety and should be used instead.

Severity: Low

Java

File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
774             abi.encodeWithSelector(IERC20Metadata.decimals.selector)
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L774:774

L008 - Consider implementing two-step procedure for updating protocol addresses:

Lack of two-step procedure for critical operations leaves them error-prone. Consider adding two step procedure on the critical functions. See similar findings in previous Code4rena contests for reference: https://code4rena.com/reports/2022-06-illuminate/#2-critical-changes-should-use-two-step-procedure

Severity: Low

Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
742       function setLiquidationPair(address _liquidationPair) external onlyOwner {
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L742:742

L009 - decimals() is not a part of the ERC-20 standard:

The decimals() function is not a part of the ERC-20 standard, and was added later as an optional extension. As such, some valid ERC20 tokens do not support this interface, so it is unsafe to blindly cast all tokens to this interface, and then call this function.

Severity: Low

Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
774             abi.encodeWithSelector(IERC20Metadata.decimals.selector)
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L774:774

L010 - Governance functions should be controlled by time locks:

Governance functions (such as upgrading contracts, setting critical parameters) should be controlled using time locks to introduce a delay between a proposal and its execution. This gives users time to exit before a potentially dangerous or malicious operation is applied.

Severity: Low

Java

File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
759     function setYieldFeeRecipient(address _yieldFeeRecipient) external onlyOwner {

753     function setYieldFeePercentage(uint32 _yieldFeePercentage) external onlyOwner {

735     function setClaimer(address _claimer) external onlyOwner {

742     function setLiquidationPair(address _liquidationPair) external onlyOwner {
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L742:748

L011 - Missing checks for address(0x0) when updating address state variables:

Missing checks for address(0x0) when updating address state variables

Severity: Low

Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
745         liquidationPair = _liquidationPair;

959         yieldFeeRecipient = _yieldFeeRecipient;
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L959:959

L012 - The additions/multiplications may silently overflow because they're in unchecked blocks with no preceding value checks, which may lead to unexpected results.:

The additions/multiplications may silently overflow because they're in unchecked blocks with no preceding value checks, which may lead to unexpected results.

Severity: Low

Java
File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
388             _maxDeposit = _maxYieldVaultDeposit - _latentBalance;

800         return type(uint96).max - _totalSupply;

813             return _totalAssets - totalDebt_;

828             return totalYieldBalance_ - _yieldBuffer;
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L828:828

L013 - approve()/safeApprove() may revert if the current approval is not zero:

Calling approve() without first calling approve(0) if the current approval is non-zero will revert with some tokens, such as Tether (USDT). While Tether is known to do this, it applies to other tokens as well, which are trying to protect against this attack vector. safeApprove() itself also implements this protection.Always reset the approval to zero before changing it to a new value (SafeERC20.forceApprove() does this for you), or use safeIncreaseAllowance()/safeDecreaseAllowance()

Severity: Low

Java

File: 2024-03-pooltogether/pt-v5-vault/src/PrizeVault.sol

```
862         _asset.approve(address(yieldVault), _assetsWithDust);
```

https://github.com/code-423n4/2024-03-pooltogether/tree/main/pt-v5-vault/src/PrizeVault.sol#L862:862

Disclaimers

AuditBase Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

Regarding the code's security, the audit offers no claims or guarantees. It also cannot be taken into account as an adequate evaluation of the code's usefulness and safety, its bug-free status, or any other contract clauses. Despite the fact that we did our best in conducting the analysis and putting together this report, it is crucial to note that you shouldn't rely solely on it. Instead, we advise moving forward with a number of independent audits and a public bug bounty program to ensure the security of smart contracts.

This audit does not constitute a formal partnership or relationship with the Company and should not be used to assume a business relationship.

Technical Disclaimer

On a blockchain network, smart contracts are set up and carried out. Hacking vulnerabilities may exist in the platform, the programming language used, and other applications used in conjunction with the smart contract. The explicit security of the audited smart contracts cannot therefore be guaranteed.