Cadmos Finance Security Review

Pashov Audit Group

Conducted by: pashov

January 3rd, 2023

Contents

1. About pashov

Krum Pashov, or **pashov**, is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Check his previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **Cadmos Finance** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Cadmos Finance

The protocol has three main types of actors:

1. Investors - the users of the protocol, they deposit their capital and expect to earn some yield on it
2. Strategists - the actors who manage the Protocol Treasury with the goal of a high yield on the Treasury funds
3. Administrators - they manage Investment Pools, for example by computing the performance of the Strategists and writing it on-chain, acting as an off-chain oracle

A typical usage flow would be the following:

1. Investors deposit capital (for example `DAI` tokens) to a Settlement Pool and receive Settlement Pool Tokens (ERC20) back
2. At some point Administrator triggers a transaction to pull the deposited funds from the Settlement Pool to an Investment Pool
3. Now Administrator moves the deposited funds again, this time from Investment Pool to Treasury
4. A Strategist fine-tunes the investment approach for the deposited funds so that he can increase the Pool Net Asset Value
5. After a while, the Investor can redeem his initial capital plus the yield accrued by burning his Investment Pool Tokens.

Liquidation state of an Investment Pool

An Investment Pool's Administrator can place it in a liquidation state at any time. This applies some constraints to the Pool, most important ones are:

1. Deposits are not allowed
2. Transfers to Pool Treasury are not allowed
3. All rewards are set to 0
4. The Pool state can't be changed

When an Investment Pool is in a liquidation state, an Investor can call `InvestmentPoolCore::liquidate` to directly withdraw his deposit.

Both the Strategists and the Administrators should be 100% trusted, since they have great power in the system, mainly for moving user funds.

5. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

***review commit hash* - a3754f182851ce90f33f514e6f0bd1dd2d539cdb**

Scope

The following smart contracts were in scope of the audit:

- `InvestmentPoolCore`
- `InvestmentPoolFactory`
- `ProtocolRegistry`
- `SettlementPool`
- `SimpleAdministrator`
- `Whitelist`

7. Executive Summary

Over the course of the security review, pashov engaged with Cadmos Finance to review Cadmos Finance. In this period of time a total of **17** issues were uncovered.

Protocol Summary

Protocol Name	Cadmos Finance
Date	January 3rd, 2023

Findings Count

Severity	**Amount**
Medium	5
Low	6
QA	6
Total Findings	**17**

Summary of Findings

ID	Title	Severity	Status
[M-01]	Hardcoding gas costs should be avoided	Medium	Resolved
[M-02]	transferERC20ToTreasury won't work as intended if assetToken is a multiple-address token	Medium	Resolved
[M-03]	Front-running risk in key admin actions	Medium	Resolved
[M-04]	An important flow of admin actions is not enforced, just documented	Medium	Resolved
[M-05]	Single-step ownership transfer can be dangerous	Medium	Resolved
[L-01]	Contracts are not directly implementing their interface contracts	Low	Resolved
[L-02]	Using OpenZeppelin's ECDSA with a vulnerable library version	Low	Resolved
[L-03]	Missing input validation in InvestmentPoolCore::setWhitelistOnly	Low	Resolved
[L-04]	Missing event emission	Low	Resolved
[L-05]	Flag has too many purposes	Low	Resolved
[L-06]	Wrong NatSpec/implementation	Low	Resolved
[QA-01]	Protocol is using an older Solidity version	QA	Resolved
[QA-02]	All methods have nonReentrant modifier	QA	Resolved
[QA-03]	Check for zero balance in cancelDeposit	QA	Resolved
[QA-04]	Not used event can be removed	QA	Resolved
[QA-05]	Not used import can be removed	QA	Resolved

[QA-06]	Whitelisting modes should be handled by an enum	QA	Resolved

8. Findings

8.1. Medium Findings

[M-01] Hardcoding gas costs should be avoided

Severity

Likelihood: Medium, because changes to gas costs have happened before, but it is not certain that there will be changes that affect the protocol.

Impact: Low, because even though calculations will be wrong they can still be done off-chain

Description

The modifier `markCost` in `SimpleAdministrator` has some hard coded gas cost values like for example 21000 (the base cost of an EVM transaction). We have seen previous EVM forks changing the gas cost of some key things, for example the SSTORE opcode. This can happen again and in this case the hardcoded values in `markCost` might not be correct anymore which will lead to wrong accounting for incurred gas costs. Also if the project is deployed on a different EVM-compatible chain, the gas costs there might be different.

Recommendations

Initialize the expected gas costs in the `initialize` method and add setter functions to be able to update them in case of an EVM fork

Discussion

CADMOS

Acknowledged - corrected.

[M-02] `transferERC20ToTreasury` won't work as intended if `assetToken` is a multiple-address token

Severity

Likelihood: Low, because it requires using a multiple-address token and a malicious/compromised admin

Impact: High, because users can use 100% of their deposits

Description

Some ERC20 tokens on the blockchain are deployed behind a proxy, so they have at least 2 entry points (the proxy and the implementation) for their functionality. Example is Synthetix's `ProxyERC20` contract from where you can interact with `sUSD, sBTC etc). If such a token was used as the` assetToken`token in an InvestmentPool, then the admin will be able to rug all depositors with the`transferERC20ToTreasury` method, even though it has the following check

```
require(tokenAddress != _assetTokenAddress, "IP: Asset transfer");
```

Since the tokens have multiple addresses the admin can give another address and pass those checks.

Recommendations

Instead of checking the address of the transferred token, it is a better approach to check the balance of it before and after the transfer and to verify it is the same.

Discussion

CADMOS

Acknowledged - corrected.

[M-03] Front-running risk in key admin actions

Severity

Likelihood: Medium, because it requires the malicious user to have a script that monitors the public mempool

Impact: Medium, because key admin functionality will revert

Description

The methods `forceTransfer`, `whitelistAccount` and `freezeAccount` from `InvestmentPoolCore` and `Whitelist` can be monitored for transactions and front-ran. Imagine the following scenario:

1. Bob holds some `InvestmentPool` ERC20 tokens
2. For some reason, a holder of the `TOKEN_FREEZE_ROLE` decides Bob is malicious and his balance should be frozen, so he calls `Whitelist::freezeAccount`
3. Bob was expecting that and was already monitoring the mempool, so he front-runs the transaction with a transfer transaction to another address he controls
4. Now his address is frozen, but he can still move/redeem/swap his tokens since the new address is not frozen

The same logic applies for the `whitelistAccount` and `forceTransfer` functionalities.

Recommendations

Always execute transactions to the mentioned functions through a private mempool or redesign them so they are not front-runnable.

Discussion

CADMOS

Acknowledged.

[M-04] An important flow of admin actions is not enforced, just documented

Severity

Likelihood: Low, because it requires either a malicious/compromised admin or the admin to forget it has to do the correct flow of operations

Impact: High, because users will lose their funds

Description

The NatSpec of `InvestmentPoolCore::setInflowOutflowPool` contains the following comment:

```
/// @notice call batchSettlement
    //(id) beforehand, otherwise it will rug the old pool tokenholders
```

This can easily be forgotten or missed when executing a call to the method. This way of ensuring proper flow of operations is used is error-prone.

Recommendations

Ensure that `batchSettlement(id)` was called beforehand by using a flag or some storage variable to be certain that users won't be rugged.

Discussion

CADMOS

Acknowledged - corrected.

[M-05] Single-step ownership transfer can be dangerous

Severity

Likelihood: Low, because it requires an error on the admin side

Impact: High, because protocol will be bricked

Description

Single-step ownership transfer means that if a wrong address was passed when transferring ownership or admin rights it can mean that role is lost forever. This can be detrimental in the context of `InvestmentPoolCore`, where if `transferAdminRole` method was called with a wrong `newAdmin` address, then the `InvestmentPoolCore` contract will be bricked, since it relies heavily on admin-only methods.

Recommendations

It is a best practice to use two-step ownership transfer pattern, meaning ownership transfer gets to a "pending" state and the new owner should claim his new rights, otherwise the old owner still has control of the contract.

Discussion

CADMOS

Acknowledged - corrected:

- ProtocolRegistry now is Ownable2Step instead of Ownable.
- 2-Step transfer for InvestmentPool Admin. Role change
- 2-Step Admin Right transfer in simpleAdmin.

8.2. Low Findings

[L-01] Contracts are not directly implementing their interface contracts

There are interface contracts in `interfaces/` for all contracts in `contracts/` but they are not used directly. This means some method might actually not be overriden since the code is not making use of compiler checks. Make sure implementation contracts inherit directly from interface contracts.

Discussion

CADMOS

Acknowledged - corrected.

[L-02] Using OpenZeppelin's `ECDSA` with a vulnerable library version

The codebase uses version `4.4.0` for its OpenZeppelin's dependencies, but this version has a High severity vulnerability related to ECDSA - Reference Even though the code is not exploitable in its current state, it is best to upgrade the OpenZeppelin library dependency to the latest safe version (4.7.3)

Discussion

CADMOS

Acknowledged - bumped to 4.8.0.

[L-03] Missing input validation in InvestmentPoolCore::setWhitelistOnly

The only correct values of the `flag` argument are either 0, 1 or 2. This should be validated with a `require` statement.

Discussion

CADMOS

Acknowledged - corrected.

[L-04] Missing event emission

The `_newAdmin` method in `SimpleAdministrator` does not emit an event, but it should, because it is important that admin additions can be tracked easily off-chain. Emit a proper event in `_newAdmin`. Same thing for the `whitelistOffChain` method in `Whitelist` - it should emit `Whitelisted` event.

Discussion

CADMOS

- _newAdmin emits `AdminRightsChanged(newAdmin, 0, flag)`.
- Acknowledged for `whitelistOffChain` - corrected.

[L-05] Flag has too many purposes

The `setTreasury` method in `SimpleAdministrator` asks for the `FLAG_STRAT_CHANGE` flag, but it is better for that action to have its own flag, for example `FLAG_TREASURY_CHANGE`. Add a separate flag for this functionality.

Discussion

CADMOS [24/12/2022]

Acknowledged - corrected.

[L-06] Wrong NatSpec/implementation

The `setNewSoftHurdleRate` method in `SimpleAdministrator` says "activate/deactivate via FLAG_HURDLE_RATE_CHANGE" but it actually uses `FLAG_REWARD_CHANGE`. Update the flag validation or the NatSpec appropriately.

Discussion

CADMOS

Acknowledged - corrected.

8.3. QA Findings

[QA-01] Protocol is using an older Solidity version

The protocol is using Solidity compiler version 0.8.3, while the latest is 0.8.17 - you can get a lot of features and optimisations, for example Custom Errors by upgrading versions

Discussion

CADMOS

Acknowledged - bumped to 0.8.7

[QA-02] All methods have `nonReentrant` modifier

If a method does not have an external call then it is impossible to reenter, so you can skip this modifier in such methods

Discussion

CADMOS

Acknowledged - this must carefully be done as though the function cannot reenter another function it can itself be reentered.

[QA-03] Check for zero balance in `cancelDeposit`

The `cancelDeposit` method in `SettlementPool` is missing a check if the caller has more than zero balance.

Discussion

CADMOS

Acknowledged - corrected.

[QA-04] Not used event can be removed

The `ForcedTransfer` event in `SettlementPool` is not used and can be removed.

Discussion

CADMOS

Acknowledged - corrected.

[QA-05] Not used import can be removed

The `ReentrancyGuard` smart contract is imported in `ProtocolRegistry` but is not used and can be removed.

Discussion

CADMOS

Acknowledged - corrected.

[QA-06] Whitelisting modes should be handled by an enum

The `_BLACKLISTMODE`, `_WHITELISTPRIMARY` and `_WHITELISTALL` modes should be turned to a `WhitelistMode` enum .

Discussion

CADMOS

Acknowledged - corrected