Smart Contract Audit
CATS Farm

coinspect

CATS Farm

Smart Contract Audit

Prepared for CryptoArena • August 2021

v210806b

1. Executive Summary

In **August 2021**, CryptoArena engaged Coinspect to perform a source code review of the CATS Farm smart contract. The objective of the project was to evaluate the security of the smart contract.

The following issues were identified during the assessment:

High Risk	Medium Risk	Low Risk
0	1	1

The medium risk issue is about not taking into account that some "ERC20" contracts might discount transfer fees, and has the consequence that **some users might be unable to unstake their tokens, which end up locked** (see CATS-002). The low risk issue is about a problem in the function `emergencyUnstake()` that breaks an invariant and as a result **some rewards are never distributed to users and instead end up locked** (see CATS-001).

Both issues and other suggestions included in this document have been promptly fixed following Coinspect's recommendation and no further problems have been found.

2. Assessment

The audit started on **August 3** and was conducted on the `main` branch of the private git repository at https://github.com/CryptoArenaSDEX/CATS-Farm-v1 as of commit `d8565dc7c30cd69070512a612c770e73fff6e5c3` of **July 23, 2021:**

```
commit d8565dc7c30cd69070512a612c770e73fff6e5c3 (HEAD -> main, origin/main, origin/HEAD)
Author: ppoliani <ppo...com>
Date:   Fri Jul 23 11:22:43 2021 +0100

    feat(*): remove dev infura key
```

The scope of the audit was limited to the following Solidity source files, shown here with their sha256sum hash:

```
27098cbc51e88b66b0d27843124cc18e7203c5ae5ba3bb1c7c8046818c206bde  CryptoArenaFarm.sol
4bf849fddb0c45f139f926439cd1463ca8af889ee75d2739eaca5eedd709fb97  mock/TokenMock.sol
3be91e2bea052d8b0d694c58bb70728f640cf6e7cc41e2ecaba7b8c045ade9c8  utils/Misc.sol
```

On **August 6** CryptoArena sent an updated version of the contract with Misc.sol and `Ownable` removed and fixes for the two issues found during the assessment.

The last commit reviewed is:

```
commit 75f663d66bcc4a6a8fc0b57095aed9afcc638ad9
Author: Pavlos Polianidis <ppo...com>
Date:   Fri Aug 6 19:37:36 2021 +0300

    Remove ownable
```

The last verification was performed on the following Solidity source files:

```
63081c8fa37bf9ad22e85ec0b77c7a9e7ea07c80308d1d808fd9e722b1925737  CryptoArenaFarm.sol
4bf849fddb0c45f139f926439cd1463ca8af889ee75d2739eaca5eedd709fb97  mock/TokenMock.sol
```

The contracts are specified to be compiled with Solidity compiler version 0.8.6, and this is the latest version at the time of the assessment.

The repository includes 28 tests. After some mangling with `truffle-config.js` it was verified that all tests pass. It is recommended to simplify the requirements for running tests to make the tests work out-of-the-box. The contract `CryptoArenaFarm` is an ERC20 contract with burning capabilities. The contract is created with a *farming period* that spans a number of blocks from a given starting

block, addresses for the *staking token* and the *reward token*, and a value specifying the number of reward tokens that are accumulated per block in the farming period.

Users can deposit staking tokens by approving the amount to the `CryptoArenaFarm` contract and calling function `stake()`. During this call CATS tokens (tokens of the `CryptoArenaFarm` contract) are minted and transferred to the user. At a later point in time, a user can unstake the tokens by calling `unstake()`, and during this call the corresponding number of synthetic tokens are burned, the original staking tokens are transferred to the user, and the reward tokens are also transferred to the user. Note that if the user's synthetic token balance is not enough (for example if the user transferred the tokens to another account) the call to `unstake()` will revert.

The function `emergencyUnstake()` allows a user to withdraw the user's stake without getting any rewards. This can be useful for emergency situations in case something goes wrong with the function `unstake()` preventing users from withdrawing their stakes. For example, if the `CryptoArenaFarm` contract has not enough balance in the reward token, the `unstake()` function might revert. The function `emergencyUnstake()` is simpler and relies on fewer assumptions and it always allows users to withdraw their staked tokens (as long as they have enough synthetic token balance to burn, otherwise the call to `emergencyUnstake()` reverts just like `unstake()`).

There's an undesirable discrepancy between `unstake()` and `emergencyUnstake()`: function **`emergencyUnstake()` forgets to decrease the `totalStaked`** storage variable. Because of this, the value of `accCatPerToken` computed in function `calcAccCatPerToken()` and set in function `updatePool()` can be less than it should be, and a part of the rewards end up locked in the contract instead of being distributed among the users. It is recommended to **change the function `emergencyUnstake()` to decrease `totalStaked` just like it is done in function `unstake()`**. (See CATS-001.)

Furthermore, it was found that the function **`stake()` does not check if the call to `safeTransferFrom()` actually transferred the specified amount**. Some "ERC20" contracts such as USDT can discount a fee on every transfer, and the final amount transferred to the destination address can be less than the amount specified in the call to `safeTransferFrom()`. This problem can result in some users losing funds, since calling `unstake` would transfer back to the user the amount originally

specified in the call to `stake()` and not the actual amount received by the `CryptoArenaFarm` contract, and there will be not enough balance of the staking token to transfer to the last user that calls `unstake()` or `emergencyUnstake()`. Instead of assuming that the staking token is well behaved and does not discount fees, it is recommended to **compute the actual amount transferred as the difference between balances before and after the transfer**. This computed amount should be used for minting synthetic tokens as well, instead of the amount specified in the call to `stake()`. (See CATS-002).

The contract `CryptoArenaFarm` inherits from OpenZeppelin's `Ownable`, `ReentrancyGuard` and `ERC20Burnable`, as well as from the `Misc` contract in `misc/Misc.sol` that solely provides the function `isContract()`. The function `isContract()` is never used, and **it is recommended to remove the `Misc` contract altogether**. Also, it is advisable to not inherit from `Ownable`, because its functions are never used.

3. Summary of Findings

ID	Description	Risk	Fixed
CATS-001	totalStaked not decreased on emergency unstake	Low	✔
CATS-002	Unverified transfer amount in function stake()	Medium	✔

4. Detailed Findings

CATS-001 totalStaked not decreased on emergency unstake

Total Risk	Impact	Location
Low	Low	CryptoArenaFarm.sol
Fixed	Likelihood	
✔	Low	

Description

Part of the **rewards end up locked in the contract** instead of being distributed among the users.

The function `emergencyUnstake()` allows a user to withdraw the user's stake without getting any rewards. This can be useful for emergency situations in case something goes wrong with the function `unstake()` preventing users from withdrawing their stakes. For example, if the `CryptoArenaFarm` contract has not enough balance in the reward token, the `unstake()` function might revert. The function `emergencyUnstake()` is simpler and relies on fewer assumptions, and it always allows users to withdraw their staked tokens.

However, there's an undesirable discrepancy between `unstake()` and `emergencyUnstake()`: function **`emergencyUnstake()` forgets to decrease the `totalStaked` storage variable**. Because of this, the value of `accCatPerToken` computed in function `calcAccCatPerToken()` and set in function `updatePool()` can be less than it should be, and a part of the rewards end up locked in the contract instead of being distributed among the users.

Recommendation

Decrease `totalStaked` in function `emergencyUnstake()` just as it is done in function `unstake()`.

Resolution

The issue was fixed following Coinspect's recommendation in commit `eac86cfe4b3115fa870c6bb8a9cd352a2913add0`.

CATS-002	Unverified transfer amount in function stake()		
Total Risk **Medium**	Impact Medium	Location CryptoArenaFarm.sol	
Fixed ✔	Likelihood Medium		

Description

Some users could receive less funds if ERC20 tokens that discount a fee are used. It was found that the function `stake()` does not check if the call to `safeTransferFrom()` actually transferred the specified amount. Some "ERC20" contracts such as USDT can discount a fee on every transfer, and the final amount transferred to the destination address can be less than the amount specified in the call to `safeTransferFrom()`. This problem can result in some users losing funds, since calling `unstake()` would transfer back to the user the amount originally specified in the call to `stake()` and not the actual amount received by the `CryptoArenaFarm` contract, and there will be not enough balance of the staking token to transfer to the last user that calls `unstake()` or `emergencyUnstake()`. Instead of assuming that the staking token is well behaved and doesn't discount fees.

Recommendation

Compute the actual amount transferred as the difference between balances before and after the transfer. This computed amount should be used for minting synthetic tokens as well, instead of the amount specified in the call to `stake()`.

Resolution

The issue was fixed following Coinspect's recommendation in commit `eac86cfe4b3115fa870c6bb8a9cd352a2913add0`.

5. Disclaimer

The information presented in this document is provided "as is" and without warranty. The present security audit does not cover any off-chain systems or frontends that communicate with the contracts, nor the general operational security of the organization that developed the code.