TrotelCoin Security Review

Version 2.0

15.04.2024

Conducted by: nmirchev8 , SR
deth , SR

Table of Contents

1 About Egis Security

We are a team of experienced smart contract researchers, who strive to provide the best smart contract security services possible to DeFi protocols.

Both members of Egis Security have a proven track record on public auditing platforms such as Code4rena, Sherlock & Codehawks, uncovering more than 80 High/Medium severity vulnerabilities, with multiple 2nd, 5th, and 10th place finishes.

2 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

3 Risk classification

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

3.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

3.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

3.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

4 Executive summary

Overview

Project Name	TrotelCoinV2
Repository	https://github.com/TrotelCoin/trotelcoin-contracts/
Commit hash	ebd5de0c660cfc58c65f447215d74c29e4090335
Documentation	https://docs.trotelcoin.com
Methods	Manual review

Scope

nfts/intermediate/TrotelCoinIntermediateNFTV2.sol
nfts/expert/TrotelCoinExpertNFTV2.sol
shop/TrotelCoinShop.sol
token/TrotelCoinV2.sol

Issues Found

Critical risk	1
High risk	1
Medium risk	3
Low risk	10
Informational	1

5 Findings

5.1 Critical risk

5.1.1 Users can game staking by locking 1 wei and increasing the amount in the end of the staking period

Severity: *Critical risk*

Context: TrotelCoinStakingV2.sol#L76

Description: After the `userStaking.duration` has passed, users can call `unstake` and receive their rewards.

The issue is that `increaseStaking` doesn't check if the `userStake.duration` has passed, meaning that users can `increaseStaking` after their duration has passed.

This way a user can stake 1 wei, wait for the entire duration of his stake to finish, then `increaseStaking` with a huge amount of tokens. When he unstake he will reap the rewards of the tokens that he just staked, not of the original amount. This way users can lock up very few funds, but get rewards based on an amount that they never had to lock.

Also, Currently a user can call `increaseStaking` and increase his stake at any point during his staking duration. The issue is that there is no incentive for users to lock up their funds when they call `stake`. There is no penalty to calling `increaseStaking` after the original stake.

A user can stake 1 wei originally with stake, then wait a few seconds before their staking duration has ended and call `increaseStaking` and increase their `userStaking.amount`. The user will reap the rewards based on their new amount, even though they didn't lock up all the funds in the beginning of their stake.

Recommendation: **NOTE: This recommendation assume that you mint rewards to the staking reward at the moment that user is staking (See 5.3.1)

If `increaseStaking` is called use the following formula to calculate new rewards for user:

```
    struct UserStaking {
        uint256 amount;
        uint256 startTime;
        uint256 duration;
+       uint256 rewards;
    }
    function increaseStaking(uint256 amount) external {
        UserStaking storage userStaking = stakings[msg.sender];
        require(userStaking.amount > 0, "No staking found");
+         require(block.timestamp < userStaking.startTime + userStaking.duration, "
    Staking period has ended");
+         duration = (userStaking.startTime + userStaking.duration) - block.timestamp
    ;
+         uint256 rewardAPR = 0;

+         if(durations[0] < duration){
+           for(uint256 i = 0; i < durations.length;++i){
+             if(duration < durations[i] && i > 0){
+                rewardAPR = rewards[durations[i - 1]];
+             }
+           }
+          }
+         require(rewardAPR > 0, "No incetive of staking more");
+         rewards = amount.mul(rewardAPR).mul(duration).div(365 days).div(100);
+         userStaking.rewards += rewards;
+         trotelToken.mint(address(this), rewards); // Instantly mint rewards,
    because if ‘cap‘ is reached, we don’t allow more stakes
        trotelToken.transferFrom(msg.sender, address(this), amount);

        userStaking.amount = userStaking.amount.add(amount);
        emit Staked(msg.sender, userStaking.amount, userStaking.duration);
    }
```

Resolution: Fixed

5.2 High risk

5.2.1 `Category` and `item` may be overriden, if such is deleted and another one created

Severity: *High risk*

Context: TrotelCoinShop.sol#L104 TrotelCoinShop.sol#L162

Description: If we have 5 categories - totalCategories = 5 If admin deleted 3rd category, he calls removeCategory with `_categoryId = 3`. After tx totalCategories = 4 When admin wants to add new category and call addCategory, category with id = 5, would be overriden from current operation

NOTE The same is for items: https://github.com/TrotelCoin/trotelcoin-contracts/blob/ebd5de0c660cfc58c65f447215c
L164

Recommendation: Add `deleted` field in the structs and check it in the other functions

```
    struct Category {
        string name;
        uint256[] categoryItems;
+        bool disabled;
    }

    struct Item {
        string name;
        uint256 price;
        uint256 discount;
        string emoji;
        string description;
+        bool disabled;
    }
    function removeCategory(uint256 _categoryId) external onlyRole(
        DEFAULT_ADMIN_ROLE) {
        require(categories[_categoryId].categoryItems.length == 0, "Category not
            empty");
+       categories[_categoryId].disabled = true;
-       delete categories[_categoryId];
-       totalCategories--;
        emit CategoryRemoved(_categoryId);
    }

    function removeItem(uint256 _itemId) external onlyRole(DEFAULT_ADMIN_ROLE) {
        require(_itemId > 0 && _itemId <= totalItems, "Invalid item id");
+       items[_itemId].disabled = true;
-        delete items[_itemId];
-        totalItems--;
        emit ItemRemoved(_itemId);
    }
```

Resolution: Fixed

5.3 Medium risk

5.3.1 If `TrotelCoinV2::cap` is reached, stakers funds may be locked

Severity: *Medium risk*

Context: TrotelCoinStakingV2.sol#L96

Description: TrotelCoinV2 has a max cap, which is initially set to `1000000000 * 10 ** decimals();`. The problem inside `TrotelCoinStakingV2` is that there is no validation whether it would be possible to mint rewards token to a staker after his period has ended, which is severe, because not only he looses yield, but also his inital stake, because transaciton will revert here, if `_cap` is reached. Changing `_cap` from admin is an option to unlock user stakes, but this would change initial tokenomics, which is another concern.

Recommendation: Directly mint user reward tokens to `TrotelCoinStakingV2` itself, when `stake`, or `increaseStaking` is called. This way, user won't be able to stake, if it would be impossible to claim his rewards, because `cap` is reached. **NOTE: See 5.1.1 for example.

Resolution: Fixed

5.3.2 Changing rewards APR may not be unfair to stakers

Severity: *Medium risk*

Context: TrotelCoinStakingV2.sol#L131-L133

Description: Inside `TrotelCoinStakingV2.sol` APR per duration can be changed at any moment by admin, which may be unfair for users, which has decided to stake for old APR value. If user has staked for 30 days with APR of 5%, but team decides that 5% is too much and decides to decrease it to 2%. If this happens 1 day before the end duration of the staker, it would be very unpleasant for him.

Recommendation:

- Save APR amount at the time user stakes and use that one, when calculating rewards:

```
    struct UserStaking {
        uint256 amount;
        uint256 startTime;
        uint256 duration;
+        uint256 apr
    }
```

```
    function stake(uint256 amount, uint256 duration) external isValidDuration(
        duration) {
     ...
        stakings[msg.sender] = UserStaking({
            amount: amount,
            startTime: block.timestamp,
            duration: duration
+            apr : rewards[duration]
        });

        emit Staked(msg.sender, amount, duration);
    }
```

Pass `apr` to `calculateReward`

Resolution: Fixed

5.3.3 If item is removed, it cannot be used by user

Severity: *Medium risk*

Context: TrotelCoinShop.sol#L217

Description: When removing item, user inventory is not checked, which means that if user calls useItem, it will revert when it is last item, but user has already paid for it.

Recommendation: When removing item, check whether it exists in user inventory, or if you want to be able to remove it, but still allow user to use it, remove the following check

Resolution: Fixed

5.4 Low risk

5.4.1 The same `itemId` can be in 2 categories, but `removeItemFromCategory` stops on the first item

Severity: *Low risk*

Context: TrotelCoinShop.sol#L121-L133

Description: When `removeItemFromCategory` is called, it stops when it matches the itemId in the first category it finds it in. If an item is in 2 categories, then the function has to be called twice to completely remove it from all categories.

Recommendation: We recommend to pass a categoryId so that the code doesn't loop through all categories and their items. This will also make the function more gas-efficient and will decrease the risk of running out of gas when calling the functions when categories and categoryItems are very large.

Resolution: Fixed

5.4.2 NFT contracts `approveContract` function is redundant

Severity: *Low risk*

Context: TrotelCoinIntermediateNFTV2.sol#L52-L54

Description: Inside `TrotelCoinIntermediateNFTV2` and `TrotelCoinExpertNFTV2.sol` there is a function called `approveContract`, which will approve `NFT` contract as spender of the same `NFT` contract trotel coins, because `msg.sender` of `trotelCoin.approve` would be the `nft` contract ,which doesn't make sense and if called, it would only waste caller's gas.

Recommendation: Remove the function

Resolution: Fixed

5.4.3 `ModifyCategory` may be called with future `categoryId`, which may lead to problems

Severity: *Low risk*

Context: TrotelCoinShop.sol#L108-L111

Description: If `modifyCategory` is called with `id > totalCategories`, it would be overriden in future `addCategory`

Recommendation: Inside `modifyCategory` check if `_categoryId < totalCategories`

Resolution: Fixed

5.4.4 The feePercentage isn't constrained in initialize.

Severity: *Low risk*

Context: TrotelCoinShop.sol#L90

Description: The function changeFeePercentage applies a constraint on the value of _newFeePercentage . feePercentage is also called in initialize. There is no constraint on the value here.

```
function initialize(
        address _daoAddress,
        uint256 _feePercentage,
        address _tokenFeeAddress,
        address _upgrader
    ) public initializer {
        __AccessControl_init();
        __UUPSUpgradeable_init();

        _grantRole(UPGRADER_ROLE, _upgrader);
        _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);

        daoAddress = _daoAddress;
        feePercentage = _feePercentage;
        tokenFee = TrotelCoinV2(_tokenFeeAddress);
    }
```

Recommendation: Add the following to initialize

```
require(
            _feePercentage>= 0 && _feePercentage<= 100,
            "Fee percentage must be between 0 and 100"
        );
```

Resolution: Fixed

5.4.5 Users can stake with 0 amount, which makes it impossible to increase stake or unstake

Severity: *Low risk*

Context: TrotelCoinStakingV2.sol#L62

Description: Currently, it's possible to call stake with `amount = 0`. This will create a `UserStaking` with a `startTime` and a duration, but it will make it impossible to call `increaseStaking` or `unstake`, as both functions require that `userStaking.amount > 0`.

```
function increaseStaking(uint256 amount) external {
        UserStaking storage userStaking = stakings[msg.sender];
        require(userStaking.amount > 0, "No staking found");

        trotelToken.transferFrom(msg.sender, address(this), amount);

        userStaking.amount = userStaking.amount.add(amount);

        emit Staked(msg.sender, userStaking.amount, userStaking.duration);
    }

    function unstake() external {
        UserStaking storage userStaking = stakings[msg.sender];
        require(userStaking.amount > 0, "No staking found");
        require(
            block.timestamp >= userStaking.startTime + userStaking.duration,
            "Staking period not ended"
        );

        uint256 reward = calculateReward(userStaking.amount, userStaking.duration);
        trotelToken.mint(msg.sender, reward);
        trotelToken.transfer(msg.sender, userStaking.amount);

        emit Unstaked(msg.sender, userStaking.amount, reward);

        delete stakings[msg.sender];
    }
```

Recommendation: The only way to fix this is if a user calls stake again with `amount > 0`. This will reset the user's `startTime` and `duration`, but they will now have a valid stake and they can call `increaseStaking` and `unstake`.

Considering this as a Low, because it's a user error and they can unstuck themselves.

Resolution: Fixed

5.4.6 Use `_safeMint` instead of `mint`

Severity: *Low risk*

Context: TrotelCoinExpertNFTV2.sol#L62TrotelCoinExpertNFTV2.sol#L69 TrotelCoinIntermediateNFTV2.sol#L62 TrotelCoinIntermediateNFTV2.sol#L69

Description: Currently `_mint` is used when trying to mint NFT's. `_mint` doesn't check if the to address is a contract and can handle NFTs. This way a NFT can get minted to a contract and be stuck forever since the contract doesn't handle NFTs.

Recommendation: Use `_safeMint` instead

Resolution: Fixed

5.4.7 `removeCategory` can be called with invalid `_categoryId`

Severity: *Low risk*

Context: TrotelCoinShop.sol#L101

Description: If `removeCategory` is called with an invalid `_categoryId`, `totalCategories` will decrement. If then `addCategory` is called after this, the category that is at the last index, will have it's name changed, instead of adding a new category.

```
function removeCategory(uint256 _categoryId) external onlyRole(DEFAULT_ADMIN_ROLE) {
        require(categories[_categoryId].categoryItems.length == 0, "Category not
            empty");
        delete categories[_categoryId];
        totalCategories--;
        emit CategoryRemoved(_categoryId);
    }
```

Recommendation: Add this to `removeCategory`

```
require(_categoryId > 0 && _categoryId <= totalCategories, "Invalid category id");
```

Resolution: Fixed

5.4.8 `changeRewards` doesn't check if `duration` is valid

Severity: *Low risk*

Context: TrotelCoinStakingV2.sol#L131

Description: The function is used to change rewards based on duration. The admin can technically pass a duration that isn't in the durations array.

```
function changeRewards(uint256 duration, uint256 newAPR) external onlyRole(
    DEFAULT_ADMIN_ROLE) {
        rewards[duration] = newAPR;
    }
```

Recommendation: Consider adding the isValidDuration to the function.

Resolution: Fixed

5.4.9 The check for allowance inside `mint` for both Intermediate and Expert NFT's is redundant

Severity: *Low risk*

Context: TrotelCoinIntermediateNFTV2.sol#L59TrotelCoinExpertNFTV2.sol#L59

Description: Inside `mint`, the function checks if `msg.sender` has given allowance to `address(this)`.

```
function mint(address to) public {
        require(isEligibleForNFT(to), "Not eligible for the NFT");
        require(balanceOf(to) < 1, "Already claimed the NFT");
        require(trotelCoin.allowance(msg.sender, address(this)) >=
            holdingRequirement, "Contract not approved to spend tokens");

        trotelCoin.transferFrom(msg.sender, daoAddress, holdingRequirement);
        _mint(to, tokenIdCounter);
        tokenIdCounter++;
        mintLockTimestamp[msg.sender] = block.timestamp;
        emit NFTMinted(to, tokenIdCounter, holdingRequirement);
    }
```

The check is redundant, as if there is no allowance, the function will revert on the next line.

Recommendation: Remove `require(trotelCoin.allowance(msg.sender, address(this))>= holdingRequirement, "Contract not approved to spend tokens");` from `mint`

Resolution: Fixed

5.4.10 Calling `isEligibleForNFT` inside `mint` for intermediate/expert NFT's is redundant

Severity: *Low risk*

Context: TrotelCoinIntermediateNFTV2.sol#L57TrotelCoinExpertNFTV2.sol#L57

Description: The function checks if the user is eligible to mint the corresponding NFT. Either Intermediate or Expert.

```
function isEligibleForNFT(address user) public view returns (bool) {
        uint256 userBalance = trotelCoin.balanceOf(user);
        return userBalance >= holdingRequirement;
    }
```

The function is redundant, because inside mint, the function transfers the holdingRequirement to the daoAddress, if msg.sender doesn't have enough tokens, the function will simply revert.

Recommendation: Remove `isEligibleForNFT` from mint

Resolution: Fixed

5.5 Informational

5.5.1 Accept raw token amount for `setHoldingRequirement` to be more flexible

Severity: *Low risk*

Context: TrotelCoinIntermediateNFTV2.sol#L88

Description: It is generally accepted that functions, which accept tokens amount, it is in raw format (not scaled). This way it is more flexible to adjust this parameter

Resolution: Fixed