TrotelCoin Security Review

Version 1.0

02.04.2024

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	TrotelCoin
Repository	https://github.com/trotelcoin/trotelcoin-contracts
Commit hash	47bce136cb819f0f102f1aeb99b2a5e98f295cd2
Documentation	https://docs.trotelcoin.com
Methods	Manual review

Scope

trotelcoin-contracts/TrotelCoinEarlyNFT.sol
trotelcoin-contracts/TrotelCoinExpertNFT.sol
trotelcoin-contracts/TrotelCoinIntermediate.sol
trotelcoin-contracts/TrotelCoinLearning.sol
trotelcoin-contracts/TrotelCoinReferring.sol
trotelcoin-contracts/TrotelCoinStakingV1.sol
trotelcoin-contracts/TrotelCoinV1.sol

Issues Found

Critical risk	4
High risk	1
Medium risk	1
Low risk	1
Informational	0

5 Findings

5.1 Critical risk

5.1.1 Users can mint an infinite amount of expert and intermediate NFT's

Severity: *Critical risk*

Context: TrotelCoinIntermediateNFT.sol#L50TrotelCoinExpertNFT.sol#L50

Description: Inside `isEligibleForExpertNFT` we check if the user's TrotelCoin balance is >= than holdingRequirement

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

A user can mint an NFT, then send 10k/50k tokens to another address and mint another NFT. He can continue doing this indefinitely, completely breaking the economics behind TelCoin and its NFT's.

Recommendation: Mitigation may require complex accounting when tokens are transferred. Our suggestion is to integrate it with TrotelCoinLearning.sol contract and check rewards per person (which is equal to minted tokens from answering quizzes). This way only learners would be able to mint NFT, which makes the NFT more valuable.

Resolution: fixed

5.1.2 Users can spam `TrotelCoinReferring::refer` with the same address and receive multiple airdrops

Severity: *Critical risk*

Context: TrotelCoinReferring.sol#L34-L43

Description: User A can call `refer` with address X infinite times and receive rewards as if he has referred different addresses. Another problem is that users can use random addresses, without a guarantee that they are real/would use the system.

Recommendation:

- Implement some kind of confirm functionality, where referred addresses should call and mention the address who has invited them. Only then increment referralCounts[referredUser]
- Use OZ EnumerableSet to track referred addresses

Resolution: fixed

5.1.3 Users can spam `claimRewards` with different learner addresses and will claim rewards for quizes

Severity: *Critical risk*

Context: TrotelCoinLearning.sol#L103

Description:

The only requirements to claimRewards are: 1. `_quizId` has to be available (authorized) 2. `quizzesIdAnsweredPerLearner[_learner][_quizId]` has to be false. This is the crux of the problem. `quizzesIdAnsweredPerLearner[_learner][_quizId]` by default is `false`, meaning anyone can call `claimRewards` with whatever `_learner` address as long is it's != address(0) and an available `_quizId`

Example.

- There are 10 available quizzes with id's 1-10
- Alice creates 10 more EOA's. She will use them when calling `claimRewards`
- Alice calls `claimRewards(address(1), 1)`
- `_quizId = 1` is available so the first requirement is passed.
- `_learner` isn't in `isLearner` mapping, so `addLearner` is called, and address(1) becomes a learner.
- `quizzesIdAnsweredPerLearner[address(1)][1]` = false, since the default value of bool is false.
- address(1) gets minted rewards.
- Alice continues doing this for all 10 of her addresses and for all 10 quizzes. She can theoretically continue doing this until `trotelCoin` hit's it's totalSupply, completely draining the protocol of all its funds.

Recommendation:

- Use a modifier to `claimRewards` function to be called only by authorized address, or
- Implement `winners` mapping, where authorized address would update winners and then they can claim rewards, if their address is winner

Resolution: fixed

5.1.4 Hardcoded token amounts in raw format, instead of being multiplied by `10 ** tokenDecimals`

Severity: *Critical risk*

Context: TrotelCoinReferring.sol#L46

Description:

Places: - TrotelCoinReferring.sol#L46 - TrotelCoinIntermediate.sol#L10 - TrotelCoinExpertNFT.sol#L10

The reward amount is not scaled by the token's decimals, which are 18, this results in the wrong reward calculation, or exploited NFT minting.

Recommendation:

```
-       uint256 rewardAmount = 2500;
+       uint256 rewardAmount = 2500 * 10 ** trotelCoin.decimals();
```

Resolution: fixed

5.2 High risk

5.2.1 APR is not calculated correct. Long term users loose value

Severity: *High risk*

Context: TrotelCoinStakingV1.sol#L87-L90

Description: If a user chooses to stake 12 * 1 month, he would win >2X than staking the same amount for 12 months. The problem is that `rewards` are not calculated correctly.

If we have stakeAmount = 1000 After 12 months we would have -> `1000 * 15 / 100 = 150` as rewards If we stake each month the same 1000, we would have `12 * (1000 * 3 / 100)= 12 * 30 = 360` Also if we choose a smaller duration, we can restake winnings and earn even more.

The impact is that long-term users are not incentivized to stake.

Recommendation: When calculating reward, you should take `duration` into calculation as APR is `annual percentage rate` and formula should look like this: reward amount = (staked amount * duration * apr) / 1 year

```
-        uint256 rewards = (userStaking.amount *          stakingPeriods[
    userStaking.duration]) / 100;
+uint256 oneYear = 365 days;
+        uint256 rewards = (userStaking.amount * userStaking.duration *
    stakingPeriods[userStaking.duration]) / 100 / secondsInYear;

       trotelToken.mint(msg.sender, userStaking.amount + rewards);
  trotelToken.burn(userStaking.amount);
```

** Note: You should also change APR calculationg inside getUserStakingDetails Better implement internal function `calculateReward(uint256 amount, uint256 duration, uint256 apr)` and use it in both places

Resolution: fixed

5.3 Medium risk

5.3.1 `referralCounts[referredUser]` should be reset after `distributeRewardTokens`

Severity: *Medium risk*

Context: rotelCoinReferring.sol#L40-L42

Description: *NOTE - This may be design decision

Depending on the protocol decision whether the user can earn a referral reward only 1 time (after referring 3 other users), or for each 3 referrals, the following remediation would be different, but the current implementation is wrong because after user A has already referred 3 addresses, he would receive reward on each new referral.

Recommendation: If you want to reward user for each 3 referred addresses, reset referralCounts[referredUser] inside distributeRewardTokens for that user.

Resolution: fixed

5.4 Low risk

5.4.1 When staking period for user has ended `getUserStakingDetails` will revert

Severity: *Low risk*

Context: TrotelCoinStakingV1.sol#L109-L111

Description: NOTE: We don't have the context and following vulnerability may be a design choice In `Template721` there is a `MAX_BATCH` variable, which is checked when `publicMint` is called, but it is not inside `privateMint`.

Recommendation: Check `qty_ > MAX_BATCH` for `privateMint` and revert in such scenario

Resolution: fixed