TrotelCoin Security Review

Version 1.0

19.04.2024

Conducted by:
MaslarovK, Independent Security Researcher

Table of Contents

1 About MaslarovK

MaslarovK is an independent security researcher from Bulgaria with 3 years of experience in Web2 development. His curiosity and love for decentralisation and transparency made him transition to Web3. He has secured various protocols through public contests and private audits.

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	55a9581f1146db40017047c6e17f8797a60cd376
Resolution	N/A
Documentation	N/A
Methods	Manual review & testing

Scope

staking/TrotelCoinStakingV2.sol

Issues Found

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

5 Findings

5.1 High risk

5.1.1 Wrong rewards calculation in the TrotelCoinStakingV2::unstake

Severity: *High risk*

Context: TrotelCoinStakingV2.sol#L136-L140

Description: In the `TrotelCoinStakingV2::unstake`

```
function unstake() external {
        UserStaking storage userStaking = stakings[msg.sender];
        require(userStaking.totalAmount > 0, "No staking found");
        require(
            getUserTimeLeft(msg.sender) == 0,
            "Staking duration not yet expired"
        );

        uint256 totalReward = 0;
        for (uint256 i = 0; i < userStaking.amounts.length; i++) {
            uint256 stakingTime;
            if (i == userStaking.amounts.length - 1) {
                stakingTime = block.timestamp - userStaking.times[i];
            } else {
                stakingTime = userStaking.times[i + 1] - userStaking.times[i];
            }
            totalReward = totalReward.add(
                calculateReward(
                    userStaking.amounts[i],
                    userStaking.duration,
                    stakingTime
                )
            );
        }


        uint256 mintAmount = userStaking.totalAmount.add(totalReward);

        trotelToken.mint(msg.sender, mintAmount);
        trotelToken.burn(userStaking.totalAmount);

        emit Unstaked(msg.sender, userStaking.totalAmount, totalReward);

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

the whole rewards calculation is wrong.

This if statement checks if the stake amount calculated is the last one

```
if (i == userStaking.amounts.length - 1) {
        stakingTime = block.timestamp - userStaking.times[i];
    }
```

and if it is, the staking time is calculated as follows: `stakingTime = block.timestamp - userStaking.times[i];`, this will work properly only if the user choses to unstake in the exact moment when the stake duration ends, otherwise it will generate more rewards than intended.

The other problem arises in the **`else`** statement

```
else {
        stakingTime = userStaking.times[i + 1] - userStaking.times[i];
    }
```

the staking time is calculated as follows: `userStaking.times[i + 1] - userStaking.times[i]`, which calculates the rewards for every stake not from `userStake.startTime until the end of the duration`, but from `userStake.startTime until increaseStake is called`. This will lead to significant loss of rewards for the user everytime when the user increases the stake amount except the final stake, in the final stake - which can lead to getting more rewards than intended.

Recommendation: Implement the following changes, I have described them in the comments:

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

        //cache endTime and use it for reward calculations
        uint256 endTime = getUserTimeLeft(msg.sender);
        require(
            endTime == 0,
            "Staking duration not yet expired"
        );

        uint256 totalReward = 0;
        for (uint256 i = 0; i < userStaking.amounts.length; i++) {
            uint256 stakingTime;

            //endTime = startTime of the first stake + duration
            //endTime - startTime of every stake should work just fine
            stakingTime = endTime - userStaking.times[i];

            totalReward = totalReward.add(
                calculateReward(
                    userStaking.amounts[i],
                    userStaking.duration,
                    stakingTime
                )
            );
        }

        uint256 mintAmount = userStaking.totalAmount.add(totalReward);

        trotelToken.mint(msg.sender, mintAmount);
        trotelToken.burn(userStaking.totalAmount);

        emit Unstaked(msg.sender, userStaking.totalAmount, totalReward);

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

Resolution: Aknowledged