Cryptonics

Smart Contract Audit Report

pNetwork DAO Staking and Voting Reward System

July 31, 2020

Table of Contents

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

This audit has been performed by

Stefan Beyer

Cryptonics Consulting S.L.
Ramiro de Maeztu 7
46022 Valencia
SPAIN

https://cryptonics.consulting/
info@cryptonics.consulting

Summary of Findings

No	Description	Severity	Status
1	Reward collection for an address will eventually fail due to block gas limit	Critical	Resolved
2	Staking operation does not check for allowance	Minor	Resolved
3	Gas-intensive loops over unbounded arrays in staking manager	Minor	Resolved
4	Ineffective require statements in VotingRewards contract	Minor	Resolved
5	Unused variables (error constants) in VotingRewards contract	Informational	Resolved

All issues encountered during the audit process have been addressed by the team.

Introduction

Purpose of this Report

Cryptonics Consulting has been engaged to perform a smart contract audit for the pNetwork DAO staking and voting reward smart contracts (https://p.network/).

The objectives of the audit are as follows:

1. Determine the correct functioning of the contract, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine contract bugs, which might lead to unexpected behavior.
4. Analyze, whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The code has been provided by the developers in the form of two GitHub repositories:

https://github.com/provable-things/aragon-voting-rewards/tree/develop

commit: `343f0eb519e86334de7758b33d20c39a22dc6537`

https://github.com/provable-things/aragon-staking-manager/tree/develop

commit: `159822a39ada2f098852c83a00c9fc9d9619161b`

https://github.com/provable-things/steroids/tree/develop

commit: `3f1ae1e7a3049422159dded544979ca535074e0b`

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the contract's intended purpose by reading the available documentation.
2. Automated scanning of the contract with static code analysis tools for security vulnerabilities and the use of best practice guidelines.
3. Manual line by line analysis of the contracts source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    - Reentrancy analysis
    - Race condition analysis
    - Front-running issues and transaction order dependencies
    - Time dependencies
    - Under- / overflow and math precision issues
    - Function visibility Issues
    - Possible denial of service attacks
    - Storage Layout Vulnerabilities
4. Report preparation

Project Overview

Intended Functionality

The submitted code is based on AragonOS (https://github.com/aragon/aragonOS) and implements staking and voting reward distributions for the pNetwork DAO. Users can lock (stake) ERC-tokens for a customizable amount of time in exchange for community token. The voting rewards contract allows users to receive rewards based on the number of votes performed in a specified time period.
The Steroid contracts is a specific staking contract, linked to a Uniswap liquidity pool. PNT holders can stake their tokens used to provide liquidity.

Smart Contracts Audited

The following files have been covered in the audit process:

```
StakingManager.sol
VotingRewards.sol
Steroids.sol
```

Detailed Findings

Issues Encountered

1. Reward collection for an address will eventually fail due to block gas limit

Severity: Critical

Function `collectRewardsFor()` loops over a an ever-growing array of rewards. This means that the transaction will eventually hit the block gas limit when the array grows too large. From this point, rewards the rewards, and any future rewards for an address will be stuck. The reason for this behavior is that the array can only grow, making the loop more gas-intensive every time.

Recommendation

Clear rewards from the award array when collected. In the case that a list of already claimed rewards needs to be kept, these should be moved to a separate data-structure.

Status: Resolved

Fixed in commit: `b8b83124d19fed01cfb7176a8a4a41c4af0fcf5e`

2. Staking operation does not check for allowance

Severity: Minor

The function `stake()` in the `StakingManager` contract uses `safeTransferFrom()` to transfer tokens form `msg.sender` to the vault. However, whilst the sender's balance is checked for sufficient funds, no check is performed whether the contract has been authorized to transfer sufficient funds on behalf of the user. This means that the transfer might fail with an unspecific error message.

Recommendation

Add a call the check the contracts allowance, in order to improve code and error message clarity.

Status: Resolved

Fixed in commit: `5abc542a919f1ee2a2b12001a960647754804a5b`

3. Gas-intensive loops over unbounded arrays in staking manager

Severity: Minor

The staking manager stores information in user stakes in arrays of `Lock` data structures, limited by the `maxLocks` variable, which can be set by an authorized caller. These arrays are searched using `for` loops during insertion and removal (staking and unstaking). It is generally considered dangerous to loop over unbounded or variable-sized arrays since it results in unpredictable gas consumption. In the extreme case, this can lead to the block gas limit being exceeded and certain transactions never completing. In this particular case, this may lead to stuck funds if `unstake()` fails due to the array becoming too large. This may occur if `maxLocks` is accidentally set too large. Note, that this issue has been classified as minor because `maxLocks` can only be set by a trusted account. However, accidental misconfiguration can still lead to stuck funds.

Recommendation

In general, loops over dynamic arrays in write transactions can and should often be avoided. In this particular case, the array content could be acquired through read-only calls and the iteration over the array could be performed off-chain. Insertions and deletions could then be performed at specific array indexes. Such a refactoring would also simplify the code by moving complexity off-chain. For instance, `unstake()` could be called with a slot index as a parameter several times, freeing up the specific slot amounts at a time.
At the very least, it is recommended to implement a safe maximum that `maxLocks` cannot exceed.

Status: Resolved

Fixed in commit: `cead223fd6cc810f24798526ae35f41c87dd6327`

4. Ineffective `require` statement in `VotingRewards` contract

Severity: Minor

The function `initialize()` in the `VotingRewards` contract contains the following precondition check:

```
require(_lockTime >= 0, ERROR_WRONG_VALUE);
```

This check will always return true since the variables checked are unsigned integers, which by definition must be larger or equal to 0.

Recommendation

Remove check.

Status: Resolved

Fixed in commit: `27239e57d83d56d2fdbdc9526a11509a16c2b2d1`

5. Unused variables (error constants) in `VotingRewards` contract

Severity: Informational

There are several state variables (declared as error constants) in `VotingRewards.sol` that are never used:

ERROR_VAULT_INSUFFICENT_TOKENS
ERROR_SENDER_NOT_VOTED
ERROR_EPOCH_WRONG_VALUE

Recommendation
Remove constants.

Status: Resolved

Fixed in commit: `27239e57d83d56d2fdbdc9526a11509a16c2b2d1`

Overall Code Quality

The overall quality of the code submitted for the audit is good.

Best practice recommendations have largely been followed. Existing, audited code has been used whenever possible in the form of the Aragon codebase ((https://github.com/aragon/aragonOS).

A safe math library has been used for arithmetic operations to avoid overflow and underflow issues.

Automated Vulnerability Scanning

In addition to the manual audit, automated vulnerability scanning has been performed.

The Slither static code analyzer reported some results. These have been checked individually and were found to be match issues reported above, false positives, or minor style issues. The full output of the scans is included as appendices at the end of this report.

Test Coverage

Unit tests have been submitted with the source code. Test coverage is very complete. The following is the output produced with `solidity-coverage` test instrumentation:

StakingManager:

```
---------------------|----------|----------|----------|----------|----------------|
File                 |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
---------------------|----------|----------|----------|----------|----------------|
 contracts/          |      100 |    89.29 |      100 |      100 |                |
  StakingManager.sol |      100 |    89.29 |      100 |      100 |                |
 contracts/test/     |        0 |        0 |        0 |        0 |                |
  Imports.sol        |      100 |      100 |      100 |      100 |                |
  StandardToken.sol  |        0 |        0 |        0 |        0 |... 143,150,159 |
---------------------|----------|----------|----------|----------|----------------|
All files            |    64.65 |     62.5 |    52.63 |    65.35 |                |
---------------------|----------|----------|----------|----------|----------------|
```

VotingRewards[1]:

```
---------------------|----------|----------|----------|----------|----------------|
File                 |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
---------------------|----------|----------|----------|----------|----------------|
 contracts/          |      100 |    81.67 |      100 |      100 |                |
  VotingRewards.sol  |      100 |    81.67 |      100 |      100 |                |
 contracts/test/     |     2.56 |        0 |     8.33 |     2.56 |                |
  ExecutionTarget.sol |        0 |      100 |        0 |        0 |         8,9,13 |
  Imports.sol        |      100 |      100 |      100 |      100 |                |
  StandardToken.sol  |        0 |        0 |        0 |        0 |... 143,150,159 |
  VotingMock.sol     |      100 |      100 |      100 |      100 |                |
---------------------|----------|----------|----------|----------|----------------|
All files            |    73.43 |    68.06 |    62.07 |    73.43 |                |
---------------------|----------|----------|----------|----------|----------------|
```

Steroids[2]:

```
--------------|----------|----------|----------|----------|----------------|
File          |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
--------------|----------|----------|----------|----------|----------------|
 contracts/   |    97.87 |       85 |    92.31 |    97.92 |                |
  Steroids.sol |    97.87 |       85 |    92.31 |    97.92 |        216,217 |
--------------|----------|----------|----------|----------|----------------|
All files     |    97.87 |       85 |    92.31 |    97.92 |                |
--------------|----------|----------|----------|----------|----------------|
```

[1] Two of the tests failed under coverage instrumentation (due to time simulation), so actual branch coverage is higher than indicated.

[2] One of the test failed under coverage instrumentation

Appendix - Automated Vulnerability Scanning Output

StakingManager.sol

```
INFO:Detectors:
Reentrancy in StakingManager.stake(uint256,uint64,address) (contracts/StakingManager.sol#100-142):
        External calls:
        - require(bool,string)(ERC20(depositToken).balanceOf(msg.sender) >=
_amount,ERROR_INSUFFICENT_TOKENS) (contracts/StakingManager.sol#105-108)
        - wrappedTokenManager.mint(_receiver,_amount) (contracts/StakingManager.sol#120)
        State variables written after the call(s):
        - addressStakeLocks[_receiver][emptyIndex] = Lock(lockDate,_duration,_amount)
(contracts/StakingManager.sol#129-133)
        - addressStakeLocks[_receiver].push(Lock(lockDate,_duration,_amount))
(contracts/StakingManager.sol#135-137)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in StakingManager.stake(uint256,uint64,address) (contracts/StakingManager.sol#100-142):
        External calls:
        - require(bool,string)(ERC20(depositToken).balanceOf(msg.sender) >=
_amount,ERROR_INSUFFICENT_TOKENS) (contracts/StakingManager.sol#105-108)
        - wrappedTokenManager.mint(_receiver,_amount) (contracts/StakingManager.sol#120)
        Event emitted after the call(s):
        - Staked(msg.sender,_receiver,_amount,_duration,lockDate)
(contracts/StakingManager.sol#140)
Reentrancy in StakingManager.unstake(uint256) (contracts/StakingManager.sol#149-160):
        External calls:
        - wrappedTokenManager.burn(msg.sender,_amount) (contracts/StakingManager.sol#155)
        - vault.transfer(depositToken,msg.sender,_amount) (contracts/StakingManager.sol#156)
        Event emitted after the call(s):
        - Unstaked(msg.sender,_amount) (contracts/StakingManager.sol#158)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StakingManager._updateStakedTokenLocks(address,uint256) (contracts/StakingManager.sol#213-262) uses
timestamp for comparisons
        Dangerous comparisons:
        - timestamp >= stakedLocks[i].lockDate.add(stakedLocks[i].duration) && !
_isWrapLockEmpty(stakedLocks[i]) (contracts/StakingManager.sol#229-231)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version^0.4.24 (contracts/StakingManager.sol#1) allows old versions
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter StakingManager.initialize(address,address,address,uint64,uint64)._tokenManager
(contracts/StakingManager.sol#74) is not in mixedCase
Parameter StakingManager.initialize(address,address,address,uint64,uint64)._vault
(contracts/StakingManager.sol#75) is not in mixedCase
Parameter StakingManager.initialize(address,address,address,uint64,uint64)._depositToken
(contracts/StakingManager.sol#76) is not in mixedCase
```

```
Parameter StakingManager.initialize(address,address,address,uint64,uint64)._minLockTime
(contracts/StakingManager.sol#77) is not in mixedCase
Parameter StakingManager.initialize(address,address,address,uint64,uint64)._maxLocks
(contracts/StakingManager.sol#78) is not in mixedCase
Parameter StakingManager.stake(uint256,uint64,address)._amount (contracts/StakingManager.sol#101) is
not in mixedCase
Parameter StakingManager.stake(uint256,uint64,address)._duration (contracts/StakingManager.sol#102)
is not in mixedCase
Parameter StakingManager.stake(uint256,uint64,address)._receiver (contracts/StakingManager.sol#103)
is not in mixedCase
Parameter StakingManager.unstake(uint256)._amount (contracts/StakingManager.sol#149) is not in
mixedCase
Parameter StakingManager.changeMinLockTime(uint64)._minLockTime (contracts/StakingManager.sol#166)
is not in mixedCase
Parameter StakingManager.changeMaxAllowedStakeLocks(uint64)._maxLocks
(contracts/StakingManager.sol#178) is not in mixedCase
Parameter StakingManager.changeVaultContractAddress(address)._vault
(contracts/StakingManager.sol#190) is not in mixedCase
Parameter StakingManager.getStakedLocks(address)._address (contracts/StakingManager.sol#204) is not
in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-convent
ions
INFO:Slither:. analyzed (60 contracts with 46 detectors), 18 result(s) found
```

VotingReward.sol

```
INFO:Detectors:
VotingRewards.initialize(address,address,address,address,uint64,uint256,uint64,uint256)
(contracts/VotingRewards.sol#124-155) contains a tautology or contradiction:
        - require(bool,string)(_lockTime >= 0,ERROR_WRONG_VALUE) (contracts/VotingRewards.sol#139)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
VotingRewards.collectRewardsFor(address) (contracts/VotingRewards.sol#409-436) has external calls
inside a loop: rewardsVault.transfer(rewardsToken,_beneficiary,rewards[i].amount)
(contracts/VotingRewards.sol#423-427)
VotingRewards._calculateReward(address,uint64,uint64,uint256) (contracts/VotingRewards.sol#447-497)
has external calls inside a loop: (None,None,startBlock,None,None,None,None,None,None,None,None) =
dandelionVoting.getVote(voteId) (contracts/VotingRewards.sol#460)
VotingRewards._calculateReward(address,uint64,uint64,uint256) (contracts/VotingRewards.sol#447-497)
has external calls inside a loop: voterState = dandelionVoting.getVoterState(voteId,_beneficiary)
(contracts/VotingRewards.sol#463-464)
VotingRewards._calculateReward(address,uint64,uint64,uint256) (contracts/VotingRewards.sol#447-497)
has external calls inside a loop: votingTokenBalanceAtVote =
MiniMeToken(dandelionVoting.token()).balanceOfAt(_beneficiary,startBlock)
(contracts/VotingRewards.sol#469-472)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in VotingRewards.collectRewardsFor(address) (contracts/VotingRewards.sol#409-436):
        External calls:
        - rewardsVault.transfer(rewardsToken,_beneficiary,rewards[i].amount)
(contracts/VotingRewards.sol#423-427)
        Event emitted after the call(s):
```

```
        - RewardCollected(_beneficiary,rewards[i].amount) (contracts/VotingRewards.sol#430)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
VotingRewards.closeRewardsDistributionForCurrentEpoch() (contracts/VotingRewards.sol#187-199)
compares to a boolean constant:
        -require(bool,string)(isDistributionOpen ==
true,ERROR_EPOCH_REWARDS_DISTRIBUTION_NOT_OPENED) (contracts/VotingRewards.sol#191-194)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Pragma version^0.4.24 (contracts/VotingRewards.sol#1) allows old versions
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter
VotingRewards.initialize(address,address,address,address,uint64,uint256,uint64,uint256)._baseVault
(contracts/VotingRewards.sol#125) is not in mixedCase
Parameter
VotingRewards.initialize(address,address,address,address,uint64,uint256,uint64,uint256)._rewardsVaul
t (contracts/VotingRewards.sol#126) is not in mixedCase
Parameter
VotingRewards.initialize(address,address,address,address,uint64,uint256,uint64,uint256)._dandelionVo
ting (contracts/VotingRewards.sol#127) is not in mixedCase
Parameter
VotingRewards.initialize(address,address,address,address,uint64,uint256,uint64,uint256)._rewardsToke
n (contracts/VotingRewards.sol#128) is not in mixedCase
Parameter
VotingRewards.initialize(address,address,address,address,uint64,uint256,uint64,uint256)._epochDurati
on (contracts/VotingRewards.sol#129) is not in mixedCase
Parameter
VotingRewards.initialize(address,address,address,address,uint64,uint256,uint64,uint256)._percentageR
ewards (contracts/VotingRewards.sol#130) is not in mixedCase
Parameter
VotingRewards.initialize(address,address,address,address,uint64,uint256,uint64,uint256)._lockTime
(contracts/VotingRewards.sol#131) is not in mixedCase
Parameter
VotingRewards.initialize(address,address,address,address,uint64,uint256,uint64,uint256)._missingVote
sThreshold (contracts/VotingRewards.sol#132) is not in mixedCase
Parameter VotingRewards.openRewardsDistributionForEpoch(uint64)._fromBlock
(contracts/VotingRewards.sol#161) is not in mixedCase
Parameter VotingRewards.distributeRewardsToMany(address[])._beneficiaries
(contracts/VotingRewards.sol#206) is not in mixedCase
Parameter VotingRewards.collectRewardsForMany(address[])._beneficiaries
(contracts/VotingRewards.sol#226) is not in mixedCase
Parameter VotingRewards.changeEpochDuration(uint64)._epochDuration (contracts/VotingRewards.sol#236)
is not in mixedCase
Parameter VotingRewards.changeMissingVotesThreshold(uint256)._missingVotesThreshold
(contracts/VotingRewards.sol#250) is not in mixedCase
Parameter VotingRewards.changeLockTime(uint64)._lockTime (contracts/VotingRewards.sol#262) is not in
mixedCase
Parameter VotingRewards.changeBaseVaultContractAddress(address)._baseVault
(contracts/VotingRewards.sol#275) is not in mixedCase
Parameter VotingRewards.changeRewardsVaultContractAddress(address)._rewardsVault
(contracts/VotingRewards.sol#289) is not in mixedCase
Parameter VotingRewards.changeDandelionVotingContractAddress(address)._dandelionVoting
(contracts/VotingRewards.sol#303) is not in mixedCase
Parameter VotingRewards.changePercentageReward(uint256)._percentageRewards
(contracts/VotingRewards.sol#318) is not in mixedCase
Parameter VotingRewards.changeRewardsTokenContractAddress(address)._rewardsToken
(contracts/VotingRewards.sol#332) is not in mixedCase
Parameter VotingRewards.getRewardsInfo(address)._beneficiary (contracts/VotingRewards.sol#346) is
```

```
not in mixedCase
Parameter VotingRewards.distributeRewardsTo(address)._beneficiary (contracts/VotingRewards.sol#361)
is not in mixedCase
Parameter VotingRewards.collectRewardsFor(address)._beneficiary (contracts/VotingRewards.sol#409) is
not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-convent
ions
INFO:Detectors:
VotingRewards.ERROR_VAULT_INSUFFICENT_TOKENS (contracts/VotingRewards.sol#45) is never used in
VotingRewards (contracts/VotingRewards.sol#14-498)
VotingRewards.ERROR_SENDER_NOT_VOTED (contracts/VotingRewards.sol#49) is never used in VotingRewards
(contracts/VotingRewards.sol#14-498)
VotingRewards.ERROR_EPOCH_WRONG_VALUE (contracts/VotingRewards.sol#55) is never used in
VotingRewards (contracts/VotingRewards.sol#14-498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Slither:. analyzed (72 contracts with 46 detectors), 33 result(s) found
```

Steriods.sol

```
INFO:Detectors:
Reentrancy in Steroids._updateStakedTokenLocks(address,uint256,uint256,uint256)
(contracts/Steroids.sol#302-365):
        External calls:
        -
_adjustBalanceAndStakedLockOf(_unstaker,stakedLocks[i],_uniswapV2PairTotalSupply,_uniswapV2PairReser
ve0) (contracts/Steroids.sol#324-329)
                -
wrappedTokenManager.mint(_owner,adjustedOwnerWrappedTokenLockAmount.sub(currentOwnerWrappedTokenAmou
nt)) (contracts/Steroids.sol#397-402)
                -
wrappedTokenManager.burn(_owner,currentOwnerWrappedTokenAmount.sub(adjustedOwnerWrappedTokenLockAmou
nt)) (contracts/Steroids.sol#404-409)
        State variables written after the call(s):
        - stakedLocks[i].wrappedTokenAmount = totalAmountUnstakedSoFar.sub(_amountToUnstake)
(contracts/Steroids.sol#342-343)
        - stakedLocks[i].uniV2PairAmount =
stakedLocks[i].wrappedTokenAmount.mul(_uniswapV2PairTotalSupply).div(_uniswapV2PairReserve0)
(contracts/Steroids.sol#345-348)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Reentrancy in Steroids.stake(uint256,uint64,address) (contracts/Steroids.sol#116-174):
        External calls:
        - require(bool,string)(uniswapV2Pair.balanceOf(msg.sender) >=
_amount,ERROR_INSUFFICENT_TOKENS) (contracts/Steroids.sol#122-125)
        - require(bool,string)(uniswapV2Pair.allowance(msg.sender,this) >=
_amount,ERROR_TOKENS_NOT_APPROVED) (contracts/Steroids.sol#126-129)
        -
require(bool,string)(uniswapV2Pair.transferFrom(msg.sender,address(vault),_amount),ERROR_TOKEN_WRAP_
REVERTED) (contracts/Steroids.sol#130-133)
        - uniswapV2PairTotalSupply = uniswapV2Pair.totalSupply() (contracts/Steroids.sol#137)
        - (uniswapV2PairReserve0) = uniswapV2Pair.getReserves() (contracts/Steroids.sol#138)
        - wrappedTokenManager.mint(_receiver,wrappedTokenAmountToStake)
(contracts/Steroids.sol#145)
```

```
        State variables written after the call(s):
        - addressStakeLocks[_receiver][emptyIndex] =
Lock(lockDate,_duration,_amount,wrappedTokenAmountToStake) (contracts/Steroids.sol#153-158)
        -
addressStakeLocks[_receiver].push(Lock(lockDate,_duration,_amount,wrappedTokenAmountToStake))
(contracts/Steroids.sol#160-162)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Steroids._adjustBalanceAndStakedLockOf(address,Steroids.Lock,uint256,uint256)
(contracts/Steroids.sol#375-415):
        External calls:
        -
wrappedTokenManager.mint(_owner,adjustedOwnerWrappedTokenLockAmount.sub(currentOwnerWrappedTokenAmou
nt)) (contracts/Steroids.sol#397-402)
        -
wrappedTokenManager.burn(_owner,currentOwnerWrappedTokenAmount.sub(adjustedOwnerWrappedTokenLockAmou
nt)) (contracts/Steroids.sol#404-409)
        Event emitted after the call(s):
        - StakedLockAdjusted(_owner,adjustedOwnerWrappedTokenLockAmount)
(contracts/Steroids.sol#413)
Reentrancy in Steroids.stake(uint256,uint64,address) (contracts/Steroids.sol#116-174):
        External calls:
        - require(bool,string)(uniswapV2Pair.balanceOf(msg.sender) >=
_amount,ERROR_INSUFFICENT_TOKENS) (contracts/Steroids.sol#122-125)
        - require(bool,string)(uniswapV2Pair.allowance(msg.sender,this) >=
_amount,ERROR_TOKENS_NOT_APPROVED) (contracts/Steroids.sol#126-129)
        -
require(bool,string)(uniswapV2Pair.transferFrom(msg.sender,address(vault),_amount),ERROR_TOKEN_WRAP_
REVERTED) (contracts/Steroids.sol#130-133)
        - uniswapV2PairTotalSupply = uniswapV2Pair.totalSupply() (contracts/Steroids.sol#137)
        - (uniswapV2PairReserve0) = uniswapV2Pair.getReserves() (contracts/Steroids.sol#138)
        - wrappedTokenManager.mint(_receiver,wrappedTokenAmountToStake)
(contracts/Steroids.sol#145)
        Event emitted after the call(s):
        - Staked(msg.sender,_receiver,_amount,wrappedTokenAmountToStake,_duration,lockDate)
(contracts/Steroids.sol#165-172)
Reentrancy in Steroids.unstake(uint256) (contracts/Steroids.sol#181-205):
        External calls:
        - uniswapV2PairTotalSupply = uniswapV2Pair.totalSupply() (contracts/Steroids.sol#182)
        - (uniswapV2PairReserve0) = uniswapV2Pair.getReserves() (contracts/Steroids.sol#183)
        -
require(bool,string)(_updateStakedTokenLocks(msg.sender,_amount,uniswapV2PairTotalSupply,uniswapV2Pa
irReserve0),ERROR_NOT_ENOUGH_UNWRAPPABLE_TOKENS) (contracts/Steroids.sol#185-193)
                -
wrappedTokenManager.mint(_owner,adjustedOwnerWrappedTokenLockAmount.sub(currentOwnerWrappedTokenAmou
nt)) (contracts/Steroids.sol#397-402)
                -
wrappedTokenManager.burn(_owner,currentOwnerWrappedTokenAmount.sub(adjustedOwnerWrappedTokenLockAmou
nt)) (contracts/Steroids.sol#404-409)
        Event emitted after the call(s):
        - StakedLockAdjusted(_owner,adjustedOwnerWrappedTokenLockAmount)
(contracts/Steroids.sol#413)
                -
require(bool,string)(_updateStakedTokenLocks(msg.sender,_amount,uniswapV2PairTotalSupply,uniswapV2Pa
irReserve0),ERROR_NOT_ENOUGH_UNWRAPPABLE_TOKENS) (contracts/Steroids.sol#185-193)
Reentrancy in Steroids.unstake(uint256) (contracts/Steroids.sol#181-205):
        External calls:
        - uniswapV2PairTotalSupply = uniswapV2Pair.totalSupply() (contracts/Steroids.sol#182)
        - (uniswapV2PairReserve0) = uniswapV2Pair.getReserves() (contracts/Steroids.sol#183)
        -
```

```
require(bool,string)(_updateStakedTokenLocks(msg.sender,_amount,uniswapV2PairTotalSupply,uniswapV2Pa
irReserve0),ERROR_NOT_ENOUGH_UNWRAPPABLE_TOKENS) (contracts/Steroids.sol#185-193)
                -
wrappedTokenManager.mint(_owner,adjustedOwnerWrappedTokenLockAmount.sub(currentOwnerWrappedTokenAmou
nt)) (contracts/Steroids.sol#397-402)
                -
wrappedTokenManager.burn(_owner,currentOwnerWrappedTokenAmount.sub(adjustedOwnerWrappedTokenLockAmou
nt)) (contracts/Steroids.sol#404-409)
        - wrappedTokenManager.burn(msg.sender,_amount) (contracts/Steroids.sol#195)
        - vault.transfer(uniswapV2Pair,msg.sender,uniV2AmountToTransfer)
(contracts/Steroids.sol#201)
        Event emitted after the call(s):
        - Unstaked(msg.sender,uniV2AmountToTransfer,_amount) (contracts/Steroids.sol#203)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Steroids._updateStakedTokenLocks(address,uint256,uint256,uint256) (contracts/Steroids.sol#302-365)
uses timestamp for comparisons
        Dangerous comparisons:
        - timestamp >= stakedLocks[i].lockDate.add(stakedLocks[i].duration) && !
_isStakedLockEmpty(stakedLocks[i]) (contracts/Steroids.sol#320-322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version^0.4.24 (contracts/Steroids.sol#1) allows old versions
Pragma version^0.4.24 (contracts/interfaces/IUniswapV2Pair.sol#1) allows old versions
Pragma version0.4.24 (contracts/test/Imports.sol#1) allows old versions
Pragma version^0.4.24 (contracts/test/StandardToken.sol#1) allows old versions
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter Steroids.initialize(address,address,address,uint64,uint64)._tokenManager
(contracts/Steroids.sol#89) is not in mixedCase
Parameter Steroids.initialize(address,address,address,uint64,uint64)._vault
(contracts/Steroids.sol#90) is not in mixedCase
Parameter Steroids.initialize(address,address,address,uint64,uint64)._uniswapV2Pair
(contracts/Steroids.sol#91) is not in mixedCase
Parameter Steroids.initialize(address,address,address,uint64,uint64)._minLockTime
(contracts/Steroids.sol#92) is not in mixedCase
Parameter Steroids.initialize(address,address,address,uint64,uint64)._maxLocks
(contracts/Steroids.sol#93) is not in mixedCase
Parameter Steroids.stake(uint256,uint64,address)._amount (contracts/Steroids.sol#117) is not in
mixedCase
Parameter Steroids.stake(uint256,uint64,address)._duration (contracts/Steroids.sol#118) is not in
mixedCase
Parameter Steroids.stake(uint256,uint64,address)._receiver (contracts/Steroids.sol#119) is not in
mixedCase
Parameter Steroids.unstake(uint256)._amount (contracts/Steroids.sol#181) is not in mixedCase
Parameter Steroids.adjustBalanceOfMany(address[])._owners (contracts/Steroids.sol#212) is not in
mixedCase
Parameter Steroids.changeMinLockTime(uint64)._minLockTime (contracts/Steroids.sol#225) is not in
mixedCase
Parameter Steroids.changeMaxAllowedStakeLocks(uint64)._maxLocks (contracts/Steroids.sol#237) is not
in mixedCase
Parameter Steroids.changeVaultContractAddress(address)._vault (contracts/Steroids.sol#250) is not in
mixedCase
Parameter Steroids.getStakedLocks(address)._address (contracts/Steroids.sol#264) is not in mixedCase
Parameter Steroids.adjustBalanceOf(address)._owner (contracts/Steroids.sol#274) is not in mixedCase
Parameter StandardToken.allowance(address,address)._owner (contracts/test/StandardToken.sol#45) is
not in mixedCase
Parameter StandardToken.allowance(address,address)._spender (contracts/test/StandardToken.sol#45) is
not in mixedCase
```

```
Parameter StandardToken.transfer(address,uint256)._to (contracts/test/StandardToken.sol#58) is not
in mixedCase
Parameter StandardToken.transfer(address,uint256)._value (contracts/test/StandardToken.sol#58) is
not in mixedCase
Parameter StandardToken.approve(address,uint256)._spender (contracts/test/StandardToken.sol#76) is
not in mixedCase
Parameter StandardToken.approve(address,uint256)._value (contracts/test/StandardToken.sol#76) is not
in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._from
(contracts/test/StandardToken.sol#89) is not in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._to
(contracts/test/StandardToken.sol#90) is not in mixedCase
Parameter StandardToken.transferFrom(address,address,uint256)._value
(contracts/test/StandardToken.sol#91) is not in mixedCase
Parameter StandardToken.increaseApproval(address,uint256)._spender
(contracts/test/StandardToken.sol#112) is not in mixedCase
Parameter StandardToken.increaseApproval(address,uint256)._addedValue
(contracts/test/StandardToken.sol#112) is not in mixedCase
Parameter StandardToken.decreaseApproval(address,uint256)._spender
(contracts/test/StandardToken.sol#132) is not in mixedCase
Parameter StandardToken.decreaseApproval(address,uint256)._subtractedValue
(contracts/test/StandardToken.sol#132) is not in mixedCase
Parameter StandardToken.balanceOf(address)._owner (contracts/test/StandardToken.sol#158) is not in
mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-convent
ions
INFO:Detectors:
allowance(address,address) should be declared external:
        - StandardToken.allowance(address,address) (contracts/test/StandardToken.sol#45-51)
transfer(address,uint256) should be declared external:
        - StandardToken.transfer(address,uint256) (contracts/test/StandardToken.sol#58-65)
approve(address,uint256) should be declared external:
        - StandardToken.approve(address,uint256) (contracts/test/StandardToken.sol#76-80)
transferFrom(address,address,uint256) should be declared external:
        - StandardToken.transferFrom(address,address,uint256)
(contracts/test/StandardToken.sol#88-101)
increaseApproval(address,uint256) should be declared external:
        - StandardToken.increaseApproval(address,uint256)
(contracts/test/StandardToken.sol#112-121)
decreaseApproval(address,uint256) should be declared external:
        - StandardToken.decreaseApproval(address,uint256)
(contracts/test/StandardToken.sol#132-144)
totalSupply() should be declared external:
        - StandardToken.totalSupply() (contracts/test/StandardToken.sol#149-151)
balanceOf(address) should be declared external:
        - StandardToken.balanceOf(address) (contracts/test/StandardToken.sol#158-160)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared
-as-external
INFO:Slither:. analyzed (61 contracts with 46 detectors), 48 result(s) found
```