THE ARCADIA GROUP

AUDIT

arcadiamgroup.com

Table of Contents

Executive Summary

A Representative Party of the NEON Protocol ("NEON") engaged The Arcadia Group ("Arcadia"), a software development, research, and security company, to conduct a review of the following NEON smart contracts on the NEON Protocol repo at Commit #`b06047c32bb8426ad3ddd52488552557935f10d6`.

NEONVault.sol

Presale.sol

NEONToken.sol

WARNING (EDIT:12/31/2020): The on-chain deployment has deployment decisions that should be taken into consideration with this project.

https://twitter.com/TheArcadiaGroup/status/1344686609451638784

Findings.

1. Rewards rounding issue

NEON-1
Severity: Low
Impact: Medium

Contract: NeonVault.sol
Category:
Finding Type: Dynamic
Lines: 451

In line 451 the devFeeAmount is calculated with a formula that leads to some rounding errors and can cause some rewards to be lost.
This issue will cause some rewards to not be able to be claimed by the staker in certain situations, so a better way would be to recalculate the rewards after knowing the devFeeAmount such as actualRewards= rewards.sub(devFeeAmount) in order to ensure that all reward will be sent to staker and dev.
This is especially important in the case of deflationary tokens and where the totalSupply is low in absolute terms as even the smallest decimals could have significant value.

```
*/
function _withdrawReward() internal {
(uint256 rewards, uint256 lastWithrewTime) = getReward(_msgSender());

require(rewards > 0, "No reward state");

uint256 devFeeAmount = rewards.mul(uint256(_devFee)).div(10000); //
problem when rounding. devFeeAmount. Rewards should be recalculated
based on devFeeAmount value due to possible rounding error

// Transfer reward tokens from contract to staker
require(
INEON(_neonAddress).transferWithoutFee(_msgSender(),
rewards.sub(devFeeAmount)),
"It has failed to transfer tokens from contract to staker."
```

```
);

// Transfer devFee tokens from contract to devAddress
require(
INEON(_neonAddress).transferWithoutFee(_devAddress,
devFeeAmount),
"It has failed to transfer tokens from contract to staker."
);

// update user's last withrew time better update the user's
lastWithrewTime before
_stakers[_msgSender()].lastWithrewTime = lastWithrewTime;

emit WithdrewReward(_msgSender(), rewards);
}
```

2. Update stakers' total amount and staker's amount after external calls

NEON-2
Severity: Low
Impact: Medium

Contract: NeonVault.sol
Category: Informational
Finding Type: Dynamic
Lines: 294-295 and 309

In function unstake() , it would be better to reassign the amount sent to a new variable and update the _totalStakedAmount and `_stakers[_msgSender()].totalStakedAmount` before the transferring the tokens (making external calls) in line 285

Also _stakers[_msgSender()].lastWithrewTime should be reassigned to another variable and set to zero before the external call.

```
function unstake() external onlyUnlocked {
require(!_isContract(_msgSender()), "Could not be a contract");
uint256 amount = _stakers[_msgSender()].totalStakedAmount;
require(amount > 0, "No running stake");

// Transfer LP tokens from contract to staker
require(
IUniV2Pair(_uniswapV2Pair).transfer(
_msgSender(),
amount),
"It has failed to transfer tokens from contract to staker."
);
_withdrawReward();

// Decrease the total staked amount //
_totalStakedAmount = _totalStakedAmount.sub(amount);
_stakers[_msgSender()].totalStakedAmount =
_stakers[_msgSender()].totalStakedAmount.sub(amount);
```

```
// Decrease the staker's amount
uint256 blockTime = block.timestamp;
uint256 lastWithrewTime = _stakers[_msgSender()].lastWithrewTime;
uint256 n = blockTime.sub(lastWithrewTime).div(_rewardPeriod); /

for (uint256 i = 0; i < n; i++) {
uint256 rewardTime = lastWithrewTime.add(_rewardPeriod.mul(i));
if (_userEpochStakedAmounts[rewardTime][_msgSender()] != 0) {
_userEpochStakedAmounts[rewardTime][_msgSender()] = 0;
}
}
// Initialize started time of user
_stakers[_msgSender()].lastWithrewTime = 0;

for (uint256 i = 0; i < _stakerList.length; i++) {
if (_stakerList[i] == _msgSender()) {
_stakerList[i] = _stakerList[_stakerList.length - 1];
_stakerList.pop();
break;
}
}

emit Unstaked(_msgSender(), amount);
}
```

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, The Arcadia Group and the Authors assume no responsibility for errors, omissions, or damages resulting from the use of the provided information. Additionally, Arcadia would like to emphasize that the use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period.

Document Information

Title	Smart Contract Audit of NEON
Client	NEON
Auditor(s)	Andrea Burzi
Reviewed by	Rasikh Morani
Approved by	Rasikh Morani
Contact Details	Rasikh Morani (972) 543-3886 rasikh@arcadiamgroup.com https://t.me/thearcadiagroup