



# Smart contracts security assessment

Final report

[Tariff: Standard](#)

## Pulse Rate Staking

July 2023



[0xguard.com](https://0xguard.com)



[hello@0xguard.com](mailto:hello@0xguard.com)

## Contents

1. Introduction	3
2. Contracts checked	3
3. Procedure	3
4. Known vulnerabilities checked	4
5. Classification of issue severity	5
6. Issues	5
7. Conclusion	8
8. Disclaimer	9
9. Slither output	10

## Introduction

The report has been prepared for **Pulse Rate Staking**.

The Pulse Rate staking allows users to lock PRATE tokens for selected period of time in exchange for rewards in native Pulse Chain PLS currency.

The code is available at the @pulserate/pulserate-contracts Github repo and was audited in the [27c995f](#) commit. Only singleStake.sol file was in the scope of this audit.

The updated code was rechecked after the commit [2a20807](#).

Name	Pulse Rate Staking
Audit date	2023-07-19 - 2023-07-20
Language	Solidity
Platform	Pulse Chain

## Contracts checked

Name	Address
SingleStake	

## Procedure

We perform our audit according to the following procedure:

### Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

## Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

## Known vulnerabilities checked

Title	Check result
<u>Unencrypted Private Data On-Chain</u>	passed
<u>Code With No Effects</u>	passed
<u>Message call with hardcoded gas amount</u>	passed
<u>Typographical Error</u>	passed
<u>DoS With Block Gas Limit</u>	passed
<u>Presence of unused variables</u>	passed
<u>Incorrect Inheritance Order</u>	passed
<u>Requirement Violation</u>	passed
<u>Weak Sources of Randomness from Chain Attributes</u>	passed
<u>Shadowing State Variables</u>	passed
<u>Incorrect Constructor Name</u>	passed
<u>Block values as a proxy for time</u>	passed
<u>Authorization through tx.origin</u>	passed
<u>DoS with Failed Call</u>	passed
<u>Delegatecall to Untrusted Callee</u>	passed
<u>Use of Deprecated Solidity Functions</u>	passed
<u>Assert Violation</u>	passed
<u>State Variable Default Visibility</u>	passed

<u>Reentrancy</u>	passed
<u>Unprotected SELFDESTRUCT Instruction</u>	passed
<u>Unprotected Ether Withdrawal</u>	passed
<u>Unchecked Call Return Value</u>	passed
<u>Floating Pragma</u>	passed
<u>Outdated Compiler Version</u>	passed
<u>Integer Overflow and Underflow</u>	passed
<u>Function Default Visibility</u>	passed

## Classification of issue severity

<b>High severity</b>	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
<b>Medium severity</b>	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
<b>Low severity</b>	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

## Issues

## High severity issues

No issues were found

## Medium severity issues

### 1. Logic error in endAllStakes (SingleStake)

Status: Fixed

The user's `activeStakes` list reduces its length during the `_endStake` execution, making it impossible to end all active stakes in a single `endAllStakes` call.

```
function endAllStakes() public nonReentrant {
    EnumerableSet.UintSet storage activeStakes = users[_msgSender()].activeStakes;
    for (uint256 i = 0; i < activeStakes.length(); i++) {
        uint256 stakeID = activeStakes.at(i);
        if (block.timestamp >= stakes[stakeID].stakeEndTime) {
            _endStake(stakeID);
        }
    }
}

function _endStake(uint256 _stakeID) private {
    ...
    user.activeStakes.remove(_stakeID);
}
```

**Recommendation:** Cache `activeStakes.length()` and fix position in `stakeID = activeStakes.at(0)`.

## Low severity issues

### 1. Constant and immutable variables (SingleStake)

Status: Fixed

The `maxStakeTime` and `stakingRoundDuration` variables should be marked as constants and renamed according to Solidity naming conventions.

The `startTime` variable is immutable wince it's not changed anywhere after the constructor.

```
uint256 public maxStakeTime = 4 weeks;
uint256 public stakingRoundDuration = 1 weeks;
uint256 public startTime;

constructor() {
    startTime = block.timestamp;
    ...
}
```

### 2. Explorer view method may be inaccurate (SingleStake)

Status: Fixed

Ended stakes can't be checked with the `pendingPLS` view function since positive `rewardDebt` is subtracted from zero. Checking these stakes in explorer may confuse user since he won't be provided with a specific error.

**Recommendation:** Return 0 reward for ended stakes either by nullifying `rewardDebt` during stake withdraw or by explicitly checking stake's status in `pendingPLS` function.

## Conclusion

Pulse Rate Staking SingleStake contract was audited. 1 medium, 2 low severity issues were found.  
1 medium, 2 low severity issues have been fixed in the update.



## Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

## Slither output

INFO:Detectors:

solc-0.8.17 is not recommended for deployment

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

INFO:Slither:. analyzed (16 contracts with 88 detectors), 1 result(s) found

