Guard

Smart contracts security assessment

Final report

Tariff: Standard

Pulse Rate

June 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

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

The Pulse Rate project is a Tomb Finance fork, allowing users to acquire PulseRate (PRATE) and PulseShare (PSHARE) tokens. Both PRATE and PSHARE tokens are ERC20 standard tokens, PRATE has privileged account allowed to mint.

PulseShare reward pool (PulseShareRewardPool contract) may charge a fee of up to 4% for each deposit.

The code is available at the @pulserate/pulserate-contracts Github repo and was audited in the 03145a9 commit.

The updated code was rechecked after the commit 2a20807.

Name	Pulse Rate
Audit date	2023-06-21 - 2023-06-23
Language	Solidity
Platform	Pulse Chain

Contracts checked

Name	Address
pulseRate.sol	
pulseShare.sol	
pulseBond.sol	
boardroom.sol	
pulseShareRewardPool.sol	
treasury.sol	
oracle.sol	

Centralization risks

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
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed

Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	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.
Medium severity	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.

Low severity	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

1. Owner capabilities (Centralization risks)

Status: Partially fixed

The project is fully centralized.

PRATE token is open for minting by operator account, which is assigned by the owner, other contracts are highly dependent on the owner's account.

Recommendation: Renounce ownership wherever possible and/or secure owner's account.

Comment from Pulse Rate team: Upon a smooth and successful launch $PRATE, $PSHARE and $PBOND contracts will be Renounced and the LP Burned!

Medium severity issues

1. Unlimited deposit fee (pulseShareRewardPool.sol)

Status: Fixed

Each pool has individual deposit fee which can be set from 0 to 2% (up to 4% in the updated version) during pool creation. However, the operator can update pool parameters after creation without any restrictions on the `depositFeeBP` parameter.

```
function add(
    uint256 _allocPoint,
    IERC20 _token,
    bool _withUpdate,
    uint256 _lastRewardTime,
```

```
    uint16 _depositFeeBP
  ) public onlyOperator {
    require(_depositFeeBP <= 200, "add: invalid deposit fee basis points");
    ...
}


function set(
    uint256 _pid,
    uint256 _allocPoint,
    uint16 _depositFeeBP
  ) public onlyOperator {
    massUpdatePools();
    PoolInfo storage pool = poolInfo[_pid];
    if (pool.isStarted)
      totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint);
    pool.allocPoint = _allocPoint;
    poolInfo[_pid].depositFeeBP = _depositFeeBP;
}
```

2. Possible reentrancy (pulseShareRewardPool.sol)
Status: Fixed

The `withdraw()` function is guarded by nonReentrant modifier, but it doesn't prevent reentrancy to the `deposit()` function. If `pool.token` or `pulseShare` tokens have user calls on transfers, withdraw-to-deposit reentrancy will cause a double reward for attacker since `user.rewardDebt` variable is updating in the very end.

```
function withdraw(uint256 _pid, uint256 _amount) public nonReentrant {
  ...
  uint256 _pending = user.amount
            .mul(pool.accTokensPerShare).div(1e18)
            .sub(user.rewardDebt);
  safePulseShareTransfer(_sender, _pending);
  pool.token.safeTransfer(_sender, _amount);
  user.rewardDebt = user.amount.mul(pool.accTokensPerShare).div(1e18);
}

function deposit(uint256 _pid, uint256 _amount, address referrer) public {
```

```
    ...
    uint256 _pending = user.amount
              .mul(pool.accTokensPerShare).div(1e18)
              .sub(user.rewardDebt);
    safePulseShareTransfer(_sender, _pending);
    ...
}
```

Low severity issues

No issues were found

Conclusion

Pulse Rate pulseRate.sol, pulseShare.sol, pulseBond.sol, boardroom.sol, pulseShareRewardPool.sol, treasury.sol, oracle.sol, Centralization risks contracts were audited. 1 high, 2 medium severity issues were found.

2 medium 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.

0x Guard