0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

Pulse Capital Presale

October 2024

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **Pulse Capital Presale**.

The audited contract is a sale contract that accepts any amounts of fixed ERC20 payment token in exchange for to be decided amounts of 2 unknown ERC20 tokens, marked as `pcap` and `stock`. The IERC20, IERC165, IERC1363, Errors, Address, SafeERC20 contracts are exact forks from the OpenZeppelin repository.

The code is available at the GitHub repository and was audited after the commit 6a844fce37efc9a5c965290fd5d039c0adb90fff.

Name	Pulse Capital Presale
Audit date	2024-10-24 - 2024-10-25
Language	Solidity
Platform	Pulse Chain

Contracts checked

Name	Address
sale.sol	

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Outdated Compiler Version	passed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 privileges (sale.sol)

Status: Open

Only the contract owner can finalize the sale, finalizations can't be forced if the ownership is lost.

The owner can set an arbitrary ERC20 token addresses as sold tokens and the sale amount is determined at the moment of the finalization, the total sold amount can be set as low as 1 token.

```
    // Function to end the deposit phase; only deployer can call this
    function end (address _pcap, address _stock) external onlyDeployer {
        require(endState == false, "Already ended");
        pcap = _pcap;
        stock = _stock;
        endState = true;
        totalPcapRaised = IERC20(pcap).balanceOf(address(this));
        totalStockRaised = IERC20(stock).balanceOf(address(this));
        require(totalPcapRaised > 0 && totalStockRaised > 0, "Owner should send the
rewards before calling the end function");
        emit ContractEnded(); // Emit End event
    }
```

Recommendation: Secure ownership. Consider using less centralized model.

Medium severity issues

1. Owner can't be changed (sale.sol)

Status: Open

The contract owner (as `deployer` address) has great privileges as he's the only authority to finalize the sale with correct parameters. The `deployer` address is set in the constructor and can't be changed later. If the `deployer` address is set to EOA and this EOA is compromised, there's no option to recover the sale, and it should be finalized immediately.

Recommendation: Secure ownership by using Ownable model or set the `deployer` address to the contract that allows changing authorities.

Low severity issues

1. Complex ERC20 transfer (sale.sol)

Status: Open

Unreasonable transfer path of the payment tokens: from the buyer to the sale contract and then to the deployer address.

```
    // Deposit function for users to transfer tokens and gain points
    function deposit(uint256 _id, uint256 _amount) external notEnded {
        . . .
        IERC20(tokenAddress).safeTransferFrom(msg.sender, address(this), _amount);
        IERC20(tokenAddress).safeTransfer(deployer, _amount);
        . . .
     }
```

Recommendation: Use `IERC20(tokenAddress).safeTransferFrom(msg.sender, deployer, _amount)` or pull payment model.

Conclusion

Pulse Capital Presale sale.sol contract was audited. 1 high, 1 medium, 1 low severity issues were found.

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:
TokenTracking.deposit(uint256,uint256) (contracts/sale.sol#48-66) contains a tautology
or contradiction:
        - require(bool,string)(_id >= 0 && _id <= MAX_ID,Invalid ID, must be between 0
and 19.) (contracts/sale.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction
INFO:Detectors:
TokenTracking.constructor(address)._tokenAddress (contracts/sale.sol#32) lacks a zero-
check on :
                - tokenAddress = _tokenAddress (contracts/sale.sol#34)
TokenTracking.end(address,address)._pcap (contracts/sale.sol#69) lacks a zero-check
on :
                - pcap = _pcap (contracts/sale.sol#71)
TokenTracking.end(address,address)._stock (contracts/sale.sol#69) lacks a zero-check
on :
                - stock = _stock (contracts/sale.sol#72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation
INFO:Detectors:
Reentrancy in TokenTracking.deposit(uint256,uint256) (contracts/sale.sol#48-66):
        External calls:
        - IERC20(tokenAddress).safeTransferFrom(msg.sender,address(this),_amount)
(contracts/sale.sol#53)
        - IERC20(tokenAddress).safeTransfer(deployer,_amount) (contracts/sale.sol#56)
        State variables written after the call(s):
        - idTotalAmount[_id] += _amount (contracts/sale.sol#59)
        - totalPoints += _amount (contracts/sale.sol#60)
        - userPoints[msg.sender] += _amount (contracts/sale.sol#63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2
INFO:Detectors:
Reentrancy in TokenTracking.claimPoints() (contracts/sale.sol#84-102):
        External calls:
        - IERC20(pcap).safeTransfer(msg.sender,pcapAmount) (contracts/sale.sol#98)
        - IERC20(stock).safeTransfer(msg.sender,stockAmount) (contracts/sale.sol#99)
        Event emitted after the call(s):
```

```
        - PointsClaimed(msg.sender,pcapAmount,stockAmount) (contracts/sale.sol#101)
Reentrancy in TokenTracking.deposit(uint256,uint256) (contracts/sale.sol#48-66):
        External calls:
        - IERC20(tokenAddress).safeTransferFrom(msg.sender,address(this),_amount)
(contracts/sale.sol#53)
        - IERC20(tokenAddress).safeTransfer(deployer,_amount) (contracts/sale.sol#56)
        Event emitted after the call(s):
        - DepositMade(msg.sender,_id,_amount) (contracts/sale.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3
INFO:Detectors:
TokenTracking.end(address,address) (contracts/sale.sol#69-81) compares to a boolean
constant:
        -require(bool,string)(endState == false,Already ended) (contracts/sale.sol#70)
TokenTracking.claimPoints() (contracts/sale.sol#84-102) compares to a boolean constant:
        -require(bool,string)(userClaimed[msg.sender] == false,User already claimed.)
(contracts/sale.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality
INFO:Detectors:
Version constraint ^0.8.20 contains known severe issues (https://
solidity.readthedocs.io/en/latest/bugs.html)
        - VerbatimInvalidDeduplication
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess.
It is used by:
        - ^0.8.20 (contracts/sale.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Parameter TokenTracking.deposit(uint256,uint256)._id (contracts/sale.sol#48) is not in
mixedCase
Parameter TokenTracking.deposit(uint256,uint256)._amount (contracts/sale.sol#48) is not
in mixedCase
Parameter TokenTracking.end(address,address)._pcap (contracts/sale.sol#69) is not in
mixedCase
Parameter TokenTracking.end(address,address)._stock (contracts/sale.sol#69) is not in
mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
INFO:Detectors:
```

```
TokenTracking.deployer (contracts/sale.sol#13) should be immutable
TokenTracking.tokenAddress (contracts/sale.sol#14) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-immutable
INFO:Slither:. analyzed (5 contracts with 93 detectors), 16 result(s) found
```

0x Guard