Smart Contract Security Audit Report

The SlowMist Security Team received the PURIEVER team's application for smart contract security audit of the PURE on Aug. 20, 2020. The following are the details and results of this smart contract security audit:

Token name :

PURE

The Contract address :

0x2904b9b16652d7d0408EcCfA23A19D4A3358230f

Link address :

https://etherscan.io/address/0x2904b9b16652d7d0408EcCfA23A19D4A3358230f

The audit items and results :

(Other unknown security vulnerabilities are not included in the audit responsibility scope)

No.	Audit Items	Audit Subclass	Audit Subclass Result
1	Overflow Audit	-	Passed
2	Race Conditions Audit	-	Passed
3	Authority Control Audit	Permission vulnerability audit	Passed
		Excessive auditing authority	Passed
4	Safety Design Audit	Zeppelin module safe use	Passed
		Compiler version security	Passed
		Hard-coded address security	Passed
		Fallback function safe use	Passed
		Show coding security	Passed
		Function return value security	Passed
		Call function security	Passed
5	Denial of Service Audit	-	Passed
6	Gas Optimization Audit	-	Passed
7	Design Logic Audit	-	Passed
8	“False Deposit” vulnerability Audit	-	Passed

9	Malicious Event Log Audit	-	Passed
10	Scoping and Declarations Audit	-	Passed
11	Replay Attack Audit	ECDSA's Signature Replay Audit	Passed
12	Uninitialized Storage Pointers Audit	-	Passed
13	Arithmetic Accuracy Deviation Audit	-	Passed

Audit Result : **Passed**

Audit Number : 0X002008250003

Audit Date : Aug. 25, 2020

Audit Team : SlowMist Security Team

(Statement : SlowMist only issues this report based on the fact that has occurred or existed before the report is issued, and bears the corresponding responsibility in this regard. For the facts occur or exist later after the report, SlowMist cannot judge the security status of its smart contract. SlowMist is not responsible for it. The security audit analysis and other contents of this report are based on the documents and materials provided by the information provider to SlowMist as of the date of this report (referred to as "the provided information"). SlowMist assumes that: there has been no information missing, tampered, deleted, or concealed. If the information provided has been missed, modified, deleted, concealed or reflected and is inconsistent with the actual situation, SlowMist will not bear any responsibility for the resulting loss and adverse effects. SlowMist will not bear any responsibility for the background or other circumstances of the project.)

Summary: This is a token contract that does not contain the tokenVault section. The total amount of contract tokens can be changed, the owner can burn his token via the admin_tokenBurn function. The Owner can lock the user's account through the add_blockedAddress function. SafeMath security module is used, which is a commendable approach. The contract does not have the Overflow and the Race Conditions issue.

The source code:

```
/**
 *Submitted for verification at Etherscan.io on 2020-08-25
*/

//SlowMist// The contract does not have the Overflow and the Race Conditions issue

pragma solidity ^0.5.17;

//SlowMist// SafeMath security Module is used, which is a recommend approach
```

```
library SafeMath
{
  function add(uint256 a, uint256 b) internal pure returns (uint256)
  {
    uint256 c = a + b;

    assert(c >= a); //SlowMist// It is recommended to replace "assert" with "require" to optimize Gas

    return c;
  }
}


contract Variable
{
  string public name;
  string public symbol;
  uint256 public decimals;
  uint256 public totalSupply;
  address public owner;
  address public watchdog;

  uint256 internal _decimals;
  bool internal transferLock;

  mapping (address => bool) public blockedAddress;

  mapping (address => uint256) public balanceOf;

  address public newWatchdog;
  address public newOwner;

  constructor() public
  {
    name = "PURIEVER";
    symbol = "PURE";
    decimals = 18;
    _decimals = 10 ** uint256(decimals);
    totalSupply = _decimals * 1200000000;
    transferLock = false;
    owner =  msg.sender;
    balanceOf[owner] = totalSupply;
    watchdog = 0x1F5Ad54c24635b6c9728078a88142C0467a2FC11;
```

```
        newWatchdog = address(0);
        newOwner = address(0);
    }
}

contract Modifiers is Variable
{

    modifier isOwner
    {
        assert(owner == msg.sender); //SlowMist// It is recommended to replace "assert" with "require" to optimize Gas
        _;
    }

    modifier isValidAddress
    {
        assert(address(0) != msg.sender); //SlowMist// It is recommended to replace "assert" with "require" to optimize Gas
        _;
    }

    modifier isWatchdog
    {
        assert(watchdog == msg.sender); //SlowMist// It is recommended to replace "assert" with "require" to optimize Gas
        _;
    }

    function transferOwnership(address _newOwner) public isWatchdog
    {
        newOwner = _newOwner;
    }

    function transferOwnershipWatchdog(address _newWatchdog) public isOwner
    {
```

```
        newWatchdog = _newWatchdog;
    }

    function acceptOwnership() public isOwner
    {
        require(newOwner != address(0)); //SlowMist// This check is quite good in avoiding losing control of the contract caused by user mistakes
        owner = newOwner;
        newOwner = address(0);
    }

    function acceptOwnershipWatchdog() public isWatchdog
    {
        require(newWatchdog != address(0));
        watchdog = newWatchdog;
        newWatchdog = address(0);
    }
}

contract Event
{
    event Transfer(address indexed from, address indexed to, uint256 value);
    event TokenBurn(address indexed from, uint256 value);
}

contract manageAddress is Variable, Modifiers, Event
{
    //SlowMist// The Owner can lock the user's account through the add_blockedAddress function
    function add_blockedAddress(address _address) public isOwner
    {
        require(_address != owner);
        blockedAddress[_address] = true;
    }
    function delete_blockedAddress(address _address) public isOwner
    {
        blockedAddress[_address] = false;
    }
}
contract Admin is Variable, Modifiers, Event
```

```
{
    function admin_tokenBurn(uint256 _value) public isOwner returns(bool success)
    {
        require(balanceOf[msg.sender] >= _value);
        balanceOf[msg.sender] -= _value;
        totalSupply -= _value;
        emit TokenBurn(msg.sender, _value);
        return true;
    }
}
contract Get is Variable, Modifiers
{
    function get_transferLock() public view returns(bool)
    {
        return transferLock;
    }
    function get_blockedAddress(address _address) public view returns(bool)
    {
        return blockedAddress[_address];
    }
}

contract Set is Variable, Modifiers, Event
{
    //SlowMist// Suspending all transactions upon major abnormalities is a recommended approach
    function setTransferLock(bool _transferLock) public isOwner returns(bool success)
    {
        transferLock = _transferLock;
        return true;
    }
}

contract PURE is Variable, Event, Get, Set, manageAddress
{
    using SafeMath for uint256;

    function() external payable
    {
        revert();
    }
```

```
  //SlowMist// This function has no return value, it is recommended to increase the return value to comply with the EIP20 specification
  function transfer(address _to, uint256 _value) public isValidAddress
  {
    require(transferLock == false);
    require(!blockedAddress[msg.sender] && !blockedAddress[_to]);
    require(balanceOf[msg.sender] >= _value && _value > 0);
    require((balanceOf[_to].add(_value)) >= balanceOf[_to] );

    balanceOf[msg.sender] -= _value;
    balanceOf[_to] += _value;
    emit Transfer(msg.sender, _to, _value);
  }
}
```

慢雾科技
slow mist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account