Smart Contract Security Audit Report

The SlowMist Security Team received the Stream Protocol team's application for smart contract security audit of the STPL on July 15, 2020. The following are the details and results of this smart contract security audit:

Token name :

STPL

The Contract address :

0x9b5C2BE869a19e84BDBcb1386dAD83a2ec8DAe82

Link address :

https://etherscan.io/address/0x9b5C2BE869a19e84BDBcb1386dAD83a2ec8DAe82

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 : 0X002007170002

Audit Date : July 17, 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, users can burn their tokens through the burn function. SafeMath security module is used, which is a commendable approach. The contract does not have the Overflow and the Race Conditions issue. The comprehensive evaluation contract is no risk.

The source code:

```
/**
 *Submitted for verification at Etherscan.io on 2020-07-03
*/

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

pragma solidity ^0.5.0;


// ----------------------------------------------------------------------
// 'STPL' 'Stream Protocol' token contract
//
```

```
// Symbol      : STPL
// Name        : Stream Protocol
// Total supply: 2,000,000,000.000000000000000000
// Decimals    : 18
//
// Enjoy.
//
// (c) Sam Jeong / SendSquare Co. 2020. The MIT Licence.
// ----------------------------------------------------------------------------



// ----------------------------------------------------------------------------
// Safe maths
// ----------------------------------------------------------------------------
```

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

```
library SafeMath {
        function add(uint a, uint b) internal pure returns (uint c) {
                c = a + b;
                require(c >= a);
        }
        function sub(uint a, uint b) internal pure returns (uint c) {
                require(b <= a);
                c = a - b;
        }
        function mul(uint a, uint b) internal pure returns (uint c) {
                c = a * b;
                require(a == 0 || c / a == b);
        }
        function div(uint a, uint b) internal pure returns (uint c) {
                require(b > 0);
                c = a / b;
        }
}



// ----------------------------------------------------------------------------
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
// ----------------------------------------------------------------------------
contract ERC20Interface {
        function totalSupply() public view returns (uint);
```

```
        function balanceOf(address tokenOwner) public view returns (uint balance);
        function allowance(address tokenOwner, address spender) public view returns (uint remaining);
        function transfer(address to, uint tokens) public returns (bool success);
        function approve(address spender, uint tokens) public returns (bool success);
        function transferFrom(address from, address to, uint tokens) public returns (bool success);

        event Transfer(address indexed from, address indexed to, uint tokens);
        event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}


// ----------------------------------------------------------------------
// Contract function to receive approval and execute function in one call
//
// Borrowed from MiniMeToken
// ----------------------------------------------------------------------
contract ApproveAndCallFallBack {
        function receiveApproval(address from, uint256 tokens, address token, bytes memory data) public;
}


// ----------------------------------------------------------------------
// Owned contract
// ----------------------------------------------------------------------
contract Owned {
        address public owner;

        constructor() public {
                owner = msg.sender;
        }

        modifier onlyOwner {
                require(msg.sender == owner);
                _;
        }
}


// ----------------------------------------------------------------------
// ERC20 Token, with the addition of symbol, name and decimals and a
// fixed supply
// ----------------------------------------------------------------------
```

```
contract StreamProtocol is ERC20Interface, Owned {
    using SafeMath for uint;

    string public symbol;
    string public name;
    uint8 public decimals;
    uint _totalSupply;
    bool _stopTrade;

    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;


    // ------------------------------------------------------------------------
    // Constructor
    // ------------------------------------------------------------------------
    constructor() public {
        symbol = "STPL";
        name = "Stream Protocol";
        decimals = 18;
        _totalSupply = 2000000000 * 10**uint(decimals);
        _stopTrade = false;
        balances[owner] = _totalSupply;
        emit Transfer(address(0), owner, _totalSupply);
    }


    // ------------------------------------------------------------------------
    // Total supply
    // ------------------------------------------------------------------------
    function totalSupply() public view returns (uint) {
        return _totalSupply.sub(balances[address(0)]);
    }


    // ------------------------------------------------------------------------
    // Stop Trade
    // ------------------------------------------------------------------------
```

//SlowMist// Suspending all transactions upon major abnormalities is a recommended approach

```
    function stopTrade() public onlyOwner {
        require(_stopTrade != true);
```

```
        _stopTrade = true;
    }


    // ------------------------------------------------------------------------
    // Start Trade
    // ------------------------------------------------------------------------
    function startTrade() public onlyOwner {
        require(_stopTrade == true);
        _stopTrade = false;
    }


    // ------------------------------------------------------------------------
    // Get the token balance for account `tokenOwner`
    // ------------------------------------------------------------------------
    function balanceOf(address tokenOwner) public view returns (uint balance) {
        return balances[tokenOwner];
    }


    // ------------------------------------------------------------------------
    // Transfer the balance from token owner's account to `to` account
    // - Owner's account must have sufficient balance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    function transfer(address to, uint tokens) public returns (bool success) {
        require(_stopTrade != true);

        require(to > address(0)); //SlowMist// This kind of check is very good, avoiding user
mistake leading to the loss of token during transfer


        balances[msg.sender] = balances[msg.sender].sub(tokens);
        balances[to] = balances[to].add(tokens);
        emit Transfer(msg.sender, to, tokens);

        return true; //SlowMist// The return value conforms to the EIP20 specification

    }


    // ------------------------------------------------------------------------
```

```
    // Token owner can approve for `spender` to transferFrom(...) `tokens`
    // from the token owner's account
    //
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
    // recommends that there are no checks for the approval double-spend attack
    // as this should be implemented in user interfaces
    // ------------------------------------------------------------------------
    function approve(address spender, uint tokens) public returns (bool success) {
        require(_stopTrade != true);

        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);

        return true; //SlowMist// The return value conforms to the EIP20 specification
    }


    // ------------------------------------------------------------------------
    // Transfer `tokens` from the `from` account to the `to` account
    //
    // The calling account must already have sufficient tokens approve(...)-d
    // for spending from the `from` account and
    // - From account must have sufficient balance to transfer
    // - Spender must have sufficient allowance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    function transferFrom(address from, address to, uint tokens) public returns (bool success) {
        require(_stopTrade != true);
        require(from > address(0));

        require(to > address(0)); //SlowMist// This kind of check is very good, avoiding user
mistake leading to the loss of token during transfer

        balances[from] = balances[from].sub(tokens);
        if(from != to && from != msg.sender) {
            allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
        }
        balances[to] = balances[to].add(tokens);
        emit Transfer(from, to, tokens);

        return true; //SlowMist// The return value conforms to the EIP20 specification
```

```
    }


    // ------------------------------------------------------------------------
    // Returns the amount of tokens approved by the owner that can be
    // transferred to the spender's account
    // ------------------------------------------------------------------------
    function allowance(address tokenOwner, address spender) public view returns (uint remaining) {
        require(_stopTrade != true);

        return allowed[tokenOwner][spender];
    }


    // ------------------------------------------------------------------------
    // Token owner can approve for `spender` to transferFrom(...) `tokens`
    // from the token owner's account. The `spender` contract function
    // `receiveApproval(...)` is then executed
    // ------------------------------------------------------------------------
    function approveAndCall(address spender, uint tokens, bytes memory data) public returns (bool success) {
        require(msg.sender != spender);

        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, address(this), data);
        return true;
    }


    // ------------------------------------------------------------------------
    // Don't accept ETH
    // ------------------------------------------------------------------------
    function () external payable {
        revert();
    }


    // ------------------------------------------------------------------------
    // Owner can transfer out any accidentally sent ERC20 tokens
    // ------------------------------------------------------------------------
    function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner returns (bool success) {
        return ERC20Interface(tokenAddress).transfer(owner, tokens);
```

```
        }

        event Burn(address indexed burner, uint256 value);

        // ------------------------------------------------------------------------
        // Burns a specific amount of tokens
        // ------------------------------------------------------------------------
        function burn(uint256 _value) public {
                require(_value <= balances[msg.sender]);

                address burner = msg.sender;
                balances[burner] = balances[burner].sub(_value);
                _totalSupply = _totalSupply.sub(_value);
                emit Burn(burner, _value);
        }
}
```

慢雾科技
slow mist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account