Smart Contract Security Audit Report

The SlowMist Security Team received the BFC team's application for smart contract security audit of the BFC on Sep. 13, 2020. The following are the details and results of this smart contract security audit:

Token name :

BFC

The Contract address :

0x4d31200e6D7854C2F664aF7Fc38a21600960F74D

Link address :

https://etherscan.io/address/0x4d31200e6D7854C2F664aF7Fc38a21600960F74D

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	Low Risk
		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 : Low Risk

Audit Number : 0X002009160001

Audit Date : Sep. 16, 2020

Audit Team : SlowMist Security Team

Summary: 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. The contract does not have the Overflow and the Race Conditions issue. During the audit, we found that the transfer function has no return value. According to the specification of EIP20, this function should return true or false.

The source code:

```
/**
 *Submitted for verification at Etherscan.io on 2020-08-10
*/

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

pragma solidity ^0.4.16;
```

```
interface tokenRecipient { function receiveApproval(address _from, uint256 _value, address _token, bytes _extraData)
external; }

contract TokenERC20 {

    string public name;
    string public symbol;
    uint8 public decimals = 18;

    uint256 public totalSupply;



    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;
   event Transfer(address indexed from, address indexed to, uint256 value);

    event Burn(address indexed from, uint256 value);

    constructor (uint256 initialSupply, string tokenName, string tokenSymbol) public {
        totalSupply = initialSupply * 10 ** uint256(decimals);   // Update total supply with the decimal amount
        balanceOf[msg.sender] = totalSupply;                     // Give the creator all initial tokens
        name = tokenName;                                        // Set the name for display purposes
        symbol = tokenSymbol;                                    // Set the symbol for display purposes
    }

    function _transfer(address _from, address _to, uint _value) internal {


        require(_to != 0x0); //SlowMist// This kind of check is very good, avoiding user mistake leading to
the loss of token during transfer

        require(balanceOf[_from] >= _value);
        require(balanceOf[_to] + _value > balanceOf[_to]);
        uint previousBalances = balanceOf[_from] + balanceOf[_to];
        balanceOf[_from] -= _value;
        balanceOf[_to] += _value;
        emit Transfer(_from, _to, _value);
        assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
    }


    //SlowMist// According to the specification of EIP20, this function should return true or false
```

```
    function transfer(address _to, uint256 _value) public {
        _transfer(msg.sender, _to, _value);
    }

    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success) {
        require(_value <= allowance[_from][msg.sender]);     // Check allowance
        allowance[_from][msg.sender] -= _value;
        _transfer(_from, _to, _value);
        return true; //SlowMist// The return value conforms to the EIP20 specification
    }

    function approve(address _spender, uint256 _value) public
        returns (bool success) {
        allowance[msg.sender][_spender] = _value;
        return true; //SlowMist// The return value conforms to the EIP20 specification
    }

    function approveAndCall(address _spender, uint256 _value, bytes _extraData)
        public
        returns (bool success) {
        tokenRecipient spender = tokenRecipient(_spender);
        if (approve(_spender, _value)) {
            spender.receiveApproval(msg.sender, _value, this, _extraData);
            return true;
        }
    }


    function burn(uint256 _value) public returns (bool success) {
        require(balanceOf[msg.sender] >= _value);   // Check if the sender has enough
        balanceOf[msg.sender] -= _value;            // Subtract from the sender
        totalSupply -= _value;                      // Updates totalSupply
        emit  Burn(msg.sender, _value);
        return true;
    }

    //SlowMist// Because burnFrom() and transferFrom() share the allowance amount of approve(),
if the agent be evil, there is the possibility of malicious burn
```

```
    function burnFrom(address _from, uint256 _value) public returns (bool success) {
        require(balanceOf[_from] >= _value);                // Check if the targeted balance is enough
        require(_value <= allowance[_from][msg.sender]);    // Check allowance
        balanceOf[_from] -= _value;                         // Subtract from the targeted balance
        allowance[_from][msg.sender] -= _value;             // Subtract from the sender's allowance
        totalSupply -= _value;                              // Update totalSupply
        emit Burn(_from, _value);
        return true;
    }
}
```

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

Github
https://github.com/slowmist