SOC Token Audit

The SOC Group team asked us to review and audit their ERC20 SOC Token contract. We looked at the code and now publish our results.

The audited code is located in the allsports/blob/master/allSportsCoin.sol.

The version used for this report is commit f8ac4930c4cd573f0d5c1cf52efa02085e4f9882.

Here is our assessment and recommendations, in order of importance.

Critical severity

No critical severity issues were found.

High severity

No high severity issues were found.

Medium severity

No medium severity issues were found.

Low severity

No low severity issues were found.

Code

```
/*
  Copyright 2017 AllSportsChain Foundation.

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at
```

```
  http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.

*/

/// @title AllSportsCoin Token contract
/// For AllSportsCoin Project: http://allsportschain.com/
/// @author allsportschain@gmail.com
pragma solidity ^0.4.16;

contract owned {
    address public owner;

    function owned() public {
        owner = msg.sender;
    }

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

}


contract TokenERC20 {
    // Public variables of the token
    string public name;
    string public symbol;
    uint8 public decimals = 18;
    // 18 decimals is the strongly suggested default, avoid changing it
    uint256 public totalSupply;

    // This creates an array with all balances
    mapping (address => uint256) public balanceOf;

    // This generates a public event on the blockchain that will notify clients
    event Transfer(address indexed from, address indexed to, uint256 value);

    /**
     * Constrctor function
     *
     * Initializes contract with initial supply tokens to the creator of the contra
```

```
ct
     */
    function TokenERC20(
        uint256 initialSupply,
        string tokenName,
        string tokenSymbol
    ) public {
        totalSupply = initialSupply * 10 ** uint256(decimals);  // Update total sup
ply with the decimal amount
        balanceOf[msg.sender] = totalSupply;                // Give the creator all
 initial tokens
        name = tokenName;                                   // Set the name for dis
play purposes
        symbol = tokenSymbol;                               // Set the symbol for d
isplay purposes
    }

    /**
     * Internal transfer, only can be called by this contract
     */
    function _transfer(address _from, address _to, uint _value) internal {
        // Prevent transfer to 0x0 address. Use burn() instead
        require(_to != 0x0);
        // Check if the sender has enough
        require(balanceOf[_from] >= _value);
        // Check for overflows
        require(balanceOf[_to] + _value > balanceOf[_to]);
        // Save this for an assertion in the future
        uint previousBalances = balanceOf[_from] + balanceOf[_to];
        // Subtract from the sender
        balanceOf[_from] -= _value;
        // Add the same to the recipient
        balanceOf[_to] += _value;
        Transfer(_from, _to, _value);
        // Asserts are used to use static analysis to find bugs in your code. They
should never fail
        assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
    }

    /**
     * Transfer tokens
     *
     * Send `_value` tokens to `_to` from your account
     *
     * @param _to The address of the recipient
     * @param _value the amount to send
     */
    function transfer(address _to, uint256 _value) public {
```

```
        _transfer(msg.sender, _to, _value);
    }


}

/******************************************/
/*       ADVANCED TOKEN STARTS HERE       */
/******************************************/

contract AllSportsCoin is owned, TokenERC20 {

    /* Initializes contract with initial supply tokens to the creator of the contra
ct */
    function AllSportsCoin(
    ) TokenERC20(1500000000, "All Sports Coin", "SOC") public {}

    /* Internal transfer, only can be called by this contract */
    function _transfer(address _from, address _to, uint _value) internal {
        require (_to != 0x0);                               // Prevent transfer to
0x0 address. Use burn() instead
        require (balanceOf[_from] >= _value);               // Check if the sender
has enough
        require (balanceOf[_to] + _value > balanceOf[_to]); // Check for overflows
        balanceOf[_from] -= _value;                         // Subtract from the se
nder
        balanceOf[_to] += _value;                           // Add the same to the
recipient
        Transfer(_from, _to, _value);
    }

}
```


Compilation

Warnings:

```
AllSportsCoin.sol:26:5: Warning: Defining constructors as functions with the same n
ame as the contract is deprecated. Use "constructor(...) { ... }" instead.

AllSportsCoin.sol:57:5: Warning: Defining constructors as functions with the same n
ame as the contract is deprecated. Use "constructor(...) { ... }" instead.
```

```
AllSportsCoin.sol:111:5: Warning: Defining constructors as functions with the same
name as the contract is deprecated. Use "constructor(...) { ... }" instead.

AllSportsCoin.sol:84:1: Warning: Invoking events without "emit" prefix is deprecate
d.

AllSportsCoin.sol:121:1: Warning: Invoking events without "emit" prefix is deprecat
ed.
```

Testing process

```
 Contract: AllSportsCoin
    owner
      when the contract is created
        ✓ should have an owner
    total supply
      ✓ returns the total amount of tokens (81ms)
    balanceOf
      when the requested account has no tokens
        ✓ returns zero (80ms)
      when the requested account has some tokens
        ✓ returns the total amount of tokens (90ms)
    transfer
      when the recipient is not the zero address
        when the sender does not have enough balance
          ✓ reverts (109ms)
        when the sender has enough balance
          ✓ transfers the requested amount (152ms)
          ✓ emits a transfer event (167ms)
      when the recipient is the zero address
        ✓ reverts (110ms)


  8 passing (2s)

--------------------|----------|----------|----------|----------|----------------|
File                |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
--------------------|----------|----------|----------|----------|----------------|
 contracts/         |      100 |    83.33 |      100 |      100 |                |
  AllSportsCoin.sol |      100 |    83.33 |      100 |      100 |                |
--------------------|----------|----------|----------|----------|----------------|
All files           |      100 |    83.33 |      100 |      100 |                |
--------------------|----------|----------|----------|----------|----------------|
```

```
Istanbul coverage reports generated
Cleaning up...
Done.
```

Classic case test result

Use safe math

```
PASSED!
```

It may not be possible to stake tokens on an invalid outcome

```
PASSED!
```

Integer index types are unnecessarily small

```
PASSED!
```

Unbound iteration in arrays

```
PASSED!
```


Reentrancy risk

```
PASSED!
```

Naming issues

```
PASSED!
```

Unused boolean return values

```
PASSED!
```

Unsolved TODO comments

```
PASSED!
```

Inconsistent usage of getter functions and state variables

```
PASSED!
```

Use a standard toolchain for building contracts

```
PASSED!
```

No assertions for detecting broken invariants

```
PASSED!
```

Attack vector

```
PASSED!
```


Conclusion

No critical or high severity issues were found. Some changes were proposed to follow best practices and reduce potential attack surface.

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the ERC20 SOC Token contract. The above should not be construed as investment advice.

(Statement: Armors Labs reports only on facts that have occurred or existed before this report is issued and assumes corresponding responsibilities. Armors Labs is not able to determine the security of its smart contracts and is not responsible for any subsequent or existing facts after this report is issued. The security audit analysis and other content of this report are only based on the documents and information provided by the information provider to Armors Labs at the

time of issuance of this report (" information provided " for short). Armors Labs postulates that the information provided is not missing, tampered, deleted or hidden. If the information provided is missing, tampered, deleted, hidden or reflected in a way that is not consistent with the actual situation, Armors Labs shall not be responsible for the losses and adverse effects caused.)

No. : 0X201806070008

Date : 2018-06-07

Team : Armors Labs

Result : PASSED

wechat :

website: armors.io

email : audit@armors.io