SCOUTCHAIN SECURITY ASSESSMENT REPORT

JAN. 14 - JAN. 18, 2019

DISCLAIMER

• This document is based on a security assessment conducted by a blockchain security company SOOHO. This document describes the detected security vulnerabilities and also discusses the code quality and code license violations.

• This security assessment does not guarantee nor describe the usefulness of the code, the stability of the code, the suitability of the business model, the legal regulation of the business, the suitability of the contract, and the bug-free status. Audit document is used for discussion purposes only.

• SOOHO does not disclose any business information obtained during the review or save it through a separate media.

• SOOHO presents its best endeavors in smart contract security assessment.

SOOHO

SOOHO with the motto of “Audit Everything” researches and provides technology for reliable blockchain ecosystem. SOOHO verifies vulnerabilities through entire development life-cycle with Aegis, a vulnerability analyzer created by SOOHO, and open source analyzers. SOOHO is composed of experts including Ph.D researchers in the field of automated security tools and white-hackers verifying contract codes and detected vulnerabilities in depth. Professional experts in SOOHO secure partners’ contracts from known to zero-day vulnerabilities.

INTRODUCTION

SOOHO conducted a security assessment of ScoutChain's smart contract from Jan. 14 to Jan. 18, 2019. The following tasks were performed during the audit period:

• Performing and analyzing the results of Aegis, a static analyzer of SOOHO.

• Performing and analyzing the results of open source analyzers - Oyente, Mythril, and Osiris.

• Writing Exploit codes on suspected vulnerability in contract.

• Recommendations on codes based on best practices and the Secure Coding Guide.

A total of two security experts participated in vulnerability analysis of the ScoutChain Contract. The experts are professional hackers with Ph.D academic backgrounds and experiences of receiving awards from national/international hacking competitions such as Defcon, Nuit du Hack, White Hat, SamsungCTF, and etc.

We scanned about 3,000 vulnerable code signatures detected through SOOHO's Aegis in contracts. We have also conducted a more diverse security vulnerability detecting process with useful security tools mainly used in Ethereum community such as Oyente, Mythril, and Osiris.

The detected vulnerabilities are as follows: Note 2. However, most of the codes are found out to be compliant with all the best practices. In addition, we confirmed that all the vulnerabilities are resolved in deployed contracts with source code. It is recommended to promote the stability of service through continuous code audit and analyze potential vulnerabilities.

ANALYSIS TARGET

The following projects were analyzed from Jan. 14 to Jan. 18:

Project	scoutTkn	**White Paper**	v1.1	**Contract Addr.**	0xb862C
Commit	3f72f9e	**MD5**	b1d08c0		
# of Files	9				
# of Lines	591				

KEY AUDIT POINTS & PROCESS

ScoutChain is a trust-based, interactive and decentralized recruitment platform. ScoutChain Token ('SCT') is utility token for ScoutChain platform. SCT is ERC20 compatible and has the ability to freeze the specific account and pause token tradings. Accordingly, we mainly reviewed common vulnerabilities in ERC tokens and possible hacking scenarios.

For example, the following scenarios are included: whether arbitrary users can access to token mint/burn, whether intentional validation skip is possible, whether race conditions are considered, and whether handling transaction results is well processed. However, we did not take any internal hackings by administrators into account.

1 **Automated Vulnerability Analysis**

2 **Manual Code Analysis**

The followings are considered:

- Preferential analysis of codes with greater risks.
- Supervision of Access Control management.
- Analyze whether the code is written under the client's intention.

3 **Review of Exploitability and PoC Code**

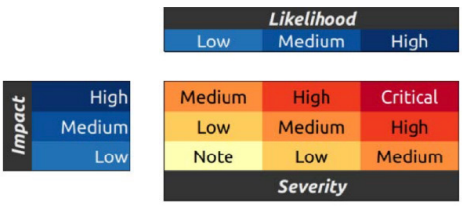
The followings are considered:

- Dynamic analysis through code execution.
- Examine possible financial gain by misusing detected vulnerabilities. (e.g., infinite withdrawal)
- Examine the possibility of adverse effect to the token service by misusing detected vulnerabilities. (e.g., Mint)

RISK RATING OF VULNERABILITY

Detected vulnerabilities are listed on the basis of the risk rating of vulnerability.

The risk rating of vulnerability is set based on OWASP's Impact & Likelihood Risk Rating Methodology as seen on the right. Some issues were rated vulnerable aside from the corresponding model and the reasons are explained in the following results.



ANALYSIS RESULTS

Analysis results are categorized into Critical, High, Medium, Low, and Note. SOOHO recommends upgrades on every detected issue.

REDUNDANT CONDITION Note

Additional resources and comments

File Name : `SCTtoken.sol`
File Location : `scoutTkn/contracts`
└── `SCTtoken.sol`
MD5 : 495cccf1b1fb639532a2b5670b070035

```
95    function freeze(uint256 _value) public  returns (bool success) {
96      require(_balances[msg.sender] >= _value && _value > 0);
97
98      _balances[msg.sender] = SafeMath.sub(_balances[msg.sender], _value);
```

```
104   function unfreeze(uint256 _value) public returns (bool success) {
105     require(freezeOf[msg.sender] >= _value && _value > 0);
106
107     freezeOf[msg.sender] = SafeMath.sub(freezeOf[msg.sender], _value);
```

The analysis result is related to gas optimization rather vulnerability detection

Details Condition statements `_balances[msg.sender] >= _value` in function `freeze` and `freezeOf[msg.sender] >= _value` in function `unfreeze` is redundant. Because each of the following statements using SafeMath guard such situtation. We recommend to remove conditions to optimize gas consumption.

OWNERSHIP CAN BE RELEASED Note

Additional resources and comments

File Name : `Ownable.sol`
File Location : `scoutTkn/contracts/helper`
└── `Ownable.sol`
MD5 : e1bc392b03129cad5bfe90f8003141b8

```
10  contract SCTtoken is ERC20, ERC20Detailed , Pausable{
```

```
10  contract Pausable is Ownable {
```

```
50      function renounceOwnership() public onlyOwner {
51          emit OwnershipTransferred(_owner, address(0));
52          _owner = address(0);
53      }
```

Details `SCTtoken` inherits `Pausable`. And, `Pausable` inherits `Ownable`. The function `renounceOwnership` declared in `Ownable` reinitialize the value of `_owner`. It makes it impossible to permanently manage the ability to freezing account and pausing token tradings which controlled by `onlyOwner`. You must either override the function in `SCTtoken` to prohibit it or remove it from `Ownable`.

ADDITIONAL ANALYSIS RESULTS

Additional analysis results include key issues that are not vulnerable but have been highlighted in the vulnerability analysis process.

TOKEN WILL NOT MINT OR BURN ✓

Additional resources and comments

File Name : `SCTtoken.sol`
File Location : `scoutTkn/contracts`
`└── SCTtoken.sol`
MD5 : 495cccf1b1fb639532a2b5670b070035

```
10  contract SCTtoken is ERC20, ERC20Detailed, Pausable {
```

```
152     function _mint(address account, uint256 value) internal {
153         require(account != address(0));
154
155         _totalSupply = _totalSupply.add(value);
156         _balances[account] = _balances[account].add(value);
157         emit Transfer(address(0), account, value);
158     }
```

Details According to white paper 22p, "The total number of SCT issued is 1 billion, and no further issue will be made." In fact, `SCTtoken` inherits `ERC20` which contains function `_mint` that can issue more token. But it is declared as `internal`, and no other function calls it. Therefore, the total number of the token will keep as the whitepaper explained.

APPROPRIATE TOTAL SUPPLY ✓

Additional resources and comments

File Name : `SCTtoken.sol`
File Location : `scoutTkn/contracts`
`└── SCTtoken.sol`
MD5 : 495cccf1b1fb639532a2b5670b070035

Details The value of totalSupply in `SCTtoken.sol` is same as whitepaper.

The total number of SCT issued is 1 billion

VERIFIED ✓

Additional resources and comments

File Name : `SCTtoken.sol`
File Location : `scoutTkn/contracts`
`└── SCTtoken.sol`
MD5 : 495cccf1b1fb639532a2b5670b070035

Details Functions of `SCTtoken.sol` have a right access control.

Ownership applied in pausing account, freezing token tradings, transferring ownership.

ADDITIONAL ANALYSIS RESULTS

Additional analysis results include key issues that are not vulnerable but have been highlighted in the vulnerability analysis process.

VERIFIED ✓

Additional resources and comments

File Name : `ERC20.sol`
File Location : `scoutTkn/contracts/helper`
└── `ERC20.sol`
MD5 : 38f48ffa4f7efa4c53e4a11beb48e5aa

Details Functions of `ERC20.sol` are safe.

VERIFIED ✓

Additional resources and comments

File Name : `ERC20Detailed.sol`
File Location : `scoutTkn/contracts/helper`
└── `ERC20Detailed.sol`
MD5 : c495589974b833f34dbe55c9c97daa3d

Details Functions of `ERC20Detailed.sol` are safe

VERIFIED ✓

Additional resources and comments

File Name : `IERC20.sol`
File Location : `scoutTkn/contracts/helper`
└── `IERC20.sol`
MD5 : f522419ba826d20c38fdc472447a8e75

Details Functions of `IERC20.sol` are safe.

VERIFIED ✓

Additional resources and comments

File Name : `Pausable.sol`
File Location : `scoutTkn/contracts/helper`
└── `Pausable.sol`
MD5 : 916946c4ef28c9c5e3c7f5e9bdb33c93

Details Functions of `Pausable.sol` are safe.

VERIFIED - MYTHRIL ✓

Additional resources and comments

Details We analyzed all detected vulnerabilities with Mythril. Most of the results were false positives.

CONCLUSION

The source code of the ScoutChain is easy to read and very well organized. We have to remark that contracts are developed the same as their whitepaper. The detected vulnerabilities are as follows: Note 2. However, most of the codes are found out to be compliant with all the best practices. It is recommended to promote the stability of ScoutChain service through continuous code audit and analyze potential vulnerabilities.

Project	scoutTkn
Version	1.0.0
# of Files	9
# of Lines	591

File Tree

```
scoutTkn
└── contracts
    ├── Migrations.sol
    ├── SCTtoken.sol Note
    └── helper
        ├── ERC20.sol
        ├── ERC20Detailed.sol
        ├── IERC20.sol
        ├── Ownable.sol Note
        ├── Pausable.sol
        └── SafeMath.sol
```