TOKENBOOST SECURITY ASSESSMENT REPORT

sooho.

SEPT. 19 - OCT. 04, 2018

DISCLAIMER

• This document is based on a security assessment conducted by a smart contract 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.

INTRODUCTION

Sooho conducted a security assessment of Tokenboost's smart contract from September 19 to October 4, 2018. 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 Manticore, and open service SmartDec.

• Writing Exploit codes on suspected vulnerability in Contract.

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

A total of six security experts participated in vulnerability analysis of the Tokenboost 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 460 vulnerable code signatures detected through Sooho's Aegis in Tokenboost 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, Remix IDE, and Manticore.

The detected vulnerabilities are as follows: Critical 2, High 3, and Medium 1. However, most of the codes are found out to be compliant with all the best practices. It is recommended to promote the stability of Tokenboost service through continuous code audit and analyze potential vulnerabilities.

sooho.
contact@sooho.io

Sooho with the motto of "Contract With Confidence" researches and provides technology for secure smart contract 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.

ANALYSIS TARGET

The following projects were analyzed from September 19 to October 4:

Project	tokenboost-solidity	**Project**	tokenboost-solidity-erc20sale	**Project**	tokenboost-solidity-erc20token
Commit	3d7bad1	**Commit**	c94634a	**Commit**	2fac179
# of Files	30	**# of Files**	15	**# of Files**	16
# of Lines	1,855	**# of Lines**	689	**# of Lines**	855

KEY AUDIT POINTS & PROCESS

Tokenboost is an easy and secure decentralized token launcher which is fundamental for blockchain business. It is divided into three concepts: Restry, Template and Contract considering the upgradability. Accordingly, we mainly reviewed common vulnerabilities in ERC tokens and possible hacking scenarios during the upgrades.

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, whether handling transaction results is well processed, and whether the memory corruption occurs during the upgrades. 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 Tokenboost's intention.

3 **Review of Exploitability and PoC Code**

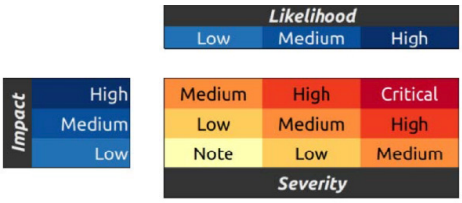
The followings are considered:

- 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.

OWNER CAN BE NULL Critical

File Name : `Contract.sol`
File Location : `tokenboost-solidity/contracts`
`└── Contract.sol`
MD5: 02e71866a33c5cf0122cb7e699aa0e9b

```
16      constructor(address _owner) public {
17          template = Template(msg.sender);
18          owner = _owner;
19
20          _registerInterface(InterfaceId_Contract);
21      }
22  }
```

Additional resources and comments

Generally, validation of the address is required by the context. We have concluded that it has greater impact considering its influence on the ownership.

Details Validation is missing in assigning the value to the `owner` in the constructor. Once the value of the `owner` becomes `NULL`, the contracts inheriting the `Contract.sol` cannot process managerial `onlyOwner` functions. Validating logics like `require(_owner != address(0))` should be added.
This following vulnerability is currently affecting tokenboost-solidity-erc20sale besides the tokenbookst-solidarity project.
Assigning `msg.sender` to `owner` may be recommended.

TOKEN CAN BE NULL Critical

File Name : `ERC20Sale.sol`
File Location : `tokenboost-solidity-erc20sale/contracts/sale/erc20`
`└── ERC20Sale.sol`
MD5 : b0979462164a4b7e37b309370b12fade

```
40      ) public Sale(_owner, _projectName, _name) {
41          token = _token;
42
43          _registerInterface(InterfaceId_ERC20Sale);
44      }
```

Additional resources and comments

Although it doesn't directly affect ownership, it is related to the token. We have concluded that it has more impact considering its influence.

```
63              _websiteUrl,
64              _whitepaperUrl,
65              _name
66          );
67          token = _token;
68      }
```

(Above) constructor
(Below) update function

Details Validation is missing in assigning the value to the `token` in the constructor and the `update` function. If the value of the `token` becomes `NULL`, the contract may not operate as intended.
Validating logics like `require(_token != address(0)` should be added.

ANALYSIS RESULTS

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

ACCESS CONTROL High

File Name : `ERC20SaleRenderer.sol`
File Location : `tokenboost-solidity-erc20sale`
`└── contracts/sale/erc20/widget`
`└── ERC20SaleRenderer.sol`
MD5 : 4315d30bba59ffc64d76f288a6875b93

File Name : `ERC20TokenRenderer.sol`
File Location : `tokenboost-solidity-erc20token`
`└── contracts/token/erc20/widget`
`└── ERC20TokenRenderer.sol`
MD5 : 0f0306715192d7af7d7c50484de6da34

```
14    function setAdminWidgetRenderers(ERC20SaleWidgetRenderer[] _renderers) public {
15        adminWidgetRenderers = _renderers;
16    }
26    function setUserWidgetRenderers(ERC20SaleWidgetRenderer[] _renderers) public {
27        userWidgetRenderers = _renderers;
28    }
38    function setInputsRenderer(ERC20SaleInputsRenderer _renderer) public {
39        inputsRenderer = _renderer;
40    }
```

Additional resources and comments

It is vulnerable ERC20SaleRenderer.sol file. ERC20TokenRenderer.sol file also discovered vulnerability similar to the ERC20SaleRenderer.

Details Any arbitrary users can access to these following functions: `setAdminWidgetRenderers`, `setUserWidgetRenderers`, `setInputsRenderer` due to its public visibility. There is a possible threat of modification of `adminWidgets`, `userWidgets`, `inputs` in "`tokenboost-solidity-erc20sale/contracts/sale/erc20/RenderableERC20Sale.sol`". Access control in `ERC20SaleRenderer` is needed.

ACCESS CONTROL High

File Name : `ClaimableTokenDistributionStrategy.sol` (Out of Scope)
File Location : `tokenboost-solidity-erc20sale-strategies/`
`└── contracts/strategy/sale/erc20`
`└── claimable-token-distribution`
`└── ClaimableTokenDistributionStrategy.sol`
MD5 : 93d6370bbe03f2aa2353f1d56187bf06

```
61    function receivesTokens(address _purchaser, uint256 _tokenAmount) public returns (bool) {
62        if (claimableTokensOfPurchaser[_purchaser] == 0) {
63            purchasers.push(_purchaser);
64        }
65        claimableTokensOfPurchaser[_purchaser] = claimableTokensOfPurchaser[_purchaser].add(_tokenAmount);
66        return true;
67    }
```

Additional resources and comments

ClaimableTokenDistributionStrategy.sol file is not our target but vulnerability has been discovered during the analysis.

Details Arbitrary users can increase claimableTokensOfPurchaser through receivesTokens functions and refund through claimTokens functions. Access control is needed.

ANALYSIS RESULTS

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

INTEGER UNDERFLOW High

File Name : `Raiser.sol`
File Location : `tokenboost-solidity/contracts`
`└── Raiser.sol`
MD5 : c43709aac723fcf7bc3c5396e5859b95

```
16      uint256 public constant MAX_HALVING_ERA = 20;
```

```
136         while (amount > 0) {
137             uint256 a = _min(amount, weiUntilNextHalving);
138             boosts = boosts.add(a.mul(2 ** (MAX_HALVING_ERA - newRewardEra).div(1000)));
139             amount = amount.sub(a);
140             newWeiUntilNextHalving = newWeiUntilNextHalving.sub(a);
141             if (newWeiUntilNextHalving == 0) {
```

Additional resources and comments

This is the only part that missed SafeMath in these projects. Other assignments are safe.

Details Integer Underflow may occur in `MAX_HALVING_ERA - newRewardEra` statement. It is due to the operation between constant 20, `MAX_HALVING_ERA`, and `uint8 newRewardEra`. SafeMath like `MAX_HALVING_ERA.sub(newRewardEra)` is recommended.

REINITIALIZE Medium

File Name : `ERC20Token.sol`
File Location : `tokenboost-solidity-erc20token`
`└── contracts/token/erc20`
`└── ERC20Token.sol`
MD5 : 75fcf57445183d32c47d3b81ed1a742b

```
76      function activate() public returns (bool) {
77          totalSupply_ = initialSupply;
78          balances[owner] = totalSupply_;
79          return super.activate();
80      }
```

Additional resources and comments

Similar vulnerabilities reported in CVE scored as High severity. But this has a lower severity because only authorized users can run the activate function.

Details Since `whenNotActivated` modifier is not applied to the `activate` function, the owner can reinitialize the `totalSupply_` and `balances[owner]` in case of accidental execution of the `activate` function.

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 : `Registry.sol`
File Location : `tokenboost-solidity/contracts/registry`
`└── Registry.sol`
MD5 : da76d35f75bba7f80dbbba123a5fc995

```
33      function register(string _identifier, uint _version, Template _template) public {
34          require(opened || msg.sender == owner);
35
36          // InterfaceId_ERC165
37          require(_template.supportsInterface(0x01ffc9a7));
38          // InterfaceId_Template
39          require(_template.supportsInterface(0xd48445ff));
40
41          address registrant = registrantOfIdentifier[_identifier];
42          require(registrant == address(0) || registrant == msg.sender, "identifier alre
43          if (registrant == address(0)) {
44              identifiers.push(_identifier);
45              registrantOfIdentifier[_identifier] = msg.sender;
46          }
47
48          uint[] storage versions = versionsOfIdentifier[_identifier];
49          if (versions.length > 0) {
50              require(_version > versions[versions.length - 1], "new version must be gre
51          }
52          versions.push(_version);
53          templateOfVersionOfIdentifier[_identifier][_version] = _template;
54      }
```

We verified the value of the interface based on the comments.

Details `Registry.sol`'s `register` function has public visibility. But `require` statement ensures the function executed only if the `opened` or `owner`.

VERIFIED ✓

Additional resources and comments

File Name : `Sale.sol`
File Location : `tokenboost-solidity/contracts/sale`
`└── Sale.sol`
MD5 : eb3f5aa4f99090942214e1ce4e017b92

```
165     function withdraw() onlyOwner whenActivated public returns (bool) {
166         require(!withdrawn);
167         require(finished());
168         require(successful());
169
170         withdrawn = true;
171         msg.sender.transfer(weiRaised);
172
173         return true;
174     }
```

Solidity official document recommends the Checks-Effects-Interactions patterns. It is well-adopted in this case.

Details `withdraw` function saves the state of withdrawn before executing the `transfer`. `claimRefund` function change before executing the `transfer`. `increasePaymentOf` function uses SafeMath. Public visibility functions have proper access control.

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 : `Sale.sol`
File Location : `tokenboost-solidity/contracts/sale`
`└── Sale.sol`
MD5 : eb3f5aa4f99090942214e1ce4e017b92

```
19      function instantiate(bytes _bytecode, bytes _args) public paya
20          ERC20Token token = ERC20Token(super.instantiate(_bytecode,
21          // InterfaceId_ERC20Token
22          require(token.supportsInterface(0x98b0f321));
23          return token;
24      }
```

Detail `instantiate` is safe despite its public visibility.

VERIFIED ✓

Additional resources and comments

File Name : `ERC20Sale.sol`
File Location : `tokenboost-solidity-erc20sale/contracts/sale/erc20`
`└── ERC20Sale.sol`
MD5 : b0979462164a4b7e37b309370b12fade

```
93      function _getTokenAmount(uint256 _weiAmount) private view retur
94          uint256 tokenAmount = _weiAmount;
95          for (uint i = 0; i < activatedStrategies.length; i++) {
96              ERC20SaleStrategy strategy = ERC20SaleStrategy(activate
97              uint256 rate = strategy.tokenRate(msg.sender, _weiAmoun
98              require(rate > 0);
99
100             tokenAmount = tokenAmount.mul(rate);
101         }
102         return tokenAmount;
103     }
```

Details `_getTokenAmount` function uses SafeMath during the multiplication. Every function has private visibility except for fallback function, which has external visibility.

VERIFIED - SMARTDEC, MYTHRIL ✓

Additional resources and comments

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

CONCLUSION

The source code of the Tokenboost is easy to read and very well organized. We have to remark that contracts are considering upgradability. The detected vulnerabilities are as follows: Critical 2, High 3, and Medium 1. However, most of the codes are found out to be compliant with all the best practices. It is recommended to promote the stability of Tokenboost service through continuous code audit and analyze potential vulnerabilities.

Project	tokenboost-solidity
Commit	3d7bad1
# of Files	30
# of Lines	1,855

Project	tokenboost-solidity-erc20sale
Commit	c94634a
# of Files	15
# of Lines	689

Project	tokenboost-solidity-erc20token
Commit	2fac179
# of Files	16
# of Lines	855

```
tokenboost-solidity/contracts
├── Activatable.sol
├── Boost.sol
├── Contract.sol Critical
├── Migrations.sol
├── Raiser.sol High
├── registry
│   ├── Registry.sol
│   ├── SaleRegistry.sol
│   ├── StrategyRegistry.sol
│   └── TokenRegistry.sol
├── sale
│   └── Sale.sol
├── strategy
│   ├── Strategy.sol
│   └── sale
│       └── SaleStrategy.sol
├── template
│   ├── Template.sol
│   ├── sale
│   │   └── SaleTemplate.sol
│   ├── strategy
│   │   ├── StrategyTemplate.sol
│   │   └── sale
│   │       └── SaleStrategyTemplate.sol
│   └── token
│       └── TokenTemplate.sol
├── token
│   └── Token.sol
├── utils
│   ├── AddressUtils.sol
│   ├── BoolUtils.sol
│   ├── ByteUtils.sol
│   ├── StringUtils.sol
│   ├── UintUtils.sol
│   └── strings.sol
└── widget
    ├── Actions.sol
    ├── Elements.sol
    ├── Localizable.sol
    ├── Renderable.sol
    ├── Tables.sol
    └── Widgets.sol
```

```
tokenboost-solidity-erc20sale/contracts
├── Migrations.sol
├── registry
│   └── ERC20SaleStrategyRegistry.sol
├── sale/erc20
│   ├── ERC20Sale.sol Critical
│   ├── RenderableERC20Sale.sol
│   └── widget
│       ├── ClaimRefundWidgetRenderer.sol
│       ├── ERC20SaleInputsRenderer.sol
│       ├── ERC20SaleRenderer.sol High
│       ├── ERC20SaleWidgetRenderer.sol
│       ├── FundsRaisedWidgetRenderer.sol
│       ├── SaleAddressWidgetRenderer.sol
│       ├── TokenInfoWidgetRenderer.sol
│       └── WithdrawFundsWidgetRenderer.sol
├── strategy/sale/erc20
│   └── ERC20SaleStrategy.sol
└── template
    ├── sale/erc20
    │   └── ERC20SaleTemplate.sol
    └── strategy/sale/erc20
        └── ERC20SaleStrategyTemplate.sol
```

```
tokenboost-solidity-erc20token/contracts
├── Migrations.sol
├── template/token/erc20
│   └── ERC20TokenTemplate.sol
└── token/erc20
    ├── ERC20Token.sol Medium
    ├── RenderableERC20Token.sol
    └── widget
        ├── BurnTokensWidgetRenderer.sol
        ├── ERC20TokenInputsRenderer.sol
        ├── ERC20TokenRenderer.sol High
        ├── ERC20TokenWidgetRenderer.sol
        ├── MaxMintableWidgetRenderer.sol
        ├── MintTokensWidgetRenderer.sol
        ├── MyBalanceWidgetRenderer.sol
        ├── PauseWidgetRenderer.sol
        ├── TokenAddressWidgetRenderer.sol
        ├── TotalSupplyWidgetRenderer.sol
        ├── TransferWidgetRenderer.sol
        └── UnpauseWidgetRenderer.sol
```