MINT CLUB SECURITY ASSESSMENT REPORT

JUN. 7 ~ JUN. 18, 2021
JUN. 27 ~ JUN. 28, 2021
JUL. 12 ~ JUL. 13, 2021

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, Automatically” 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 Bourbonshake's Mint Clube smart contract from Jun. 7 until Jun. 18, 2021, Jun. 27 until Jun. 28, 2021, and Jul. 12 until Jul 13, 2021. The following tasks were performed during the audit period:

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

• Writing Exploit codes on suspected vulnerability in the contract.

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

A total of three security experts participated in a vulnerability analysis of the 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.

The detected vulnerabilities are as follows: Note 3. We have confirmed that all issues are resolved. We would like to remark that the Bourbanshake team has enough security expertise to patch critical vulnerabilities on their own. It is recommended to promote the stability of service through continuous code audit and analyze potential vulnerabilities.

ANALYSIS TARGET

The following projects were analyzed during period.

Project	`mint.club-contract`
Commit #	fc737f4b
# of Files	7
# of Lines	608

KEY AUDIT POINTS & PROCESS

Bourbonshake's Mint Club is a code-free token issuance service. The issued token is a smart token that provides instant liquidity. It is leveraging the bonding curve to removing the burden on the liquidity providing or market-making. In addition, the development of the Factory pattern has been applied to be able to issue tokens without code. Accordingly, we mainly reviewed common vulnerabilities in Factory pattern, Token contracts and Bonding curve implementations.

However, we did not take any internal hackings by administrators into account. Analyzes are about the functioning of the subject contract, given the safety of the system.

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:

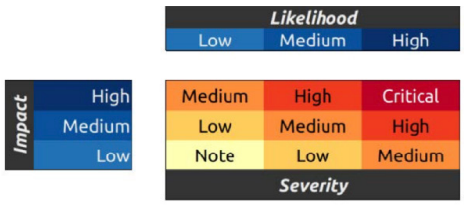
- Dynamic analysis through code execution.
- Examine possible financial gain by misusing detected vulnerabilities. (e.g., infinite withdrawal)
- Examine the possibility of adverse effect to the token service by misusing detected vulnerabilities. (e.g., Mint)

RISK RATING OF VULNERABILITY

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

Critical | High | Medium | Low | Note

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.

(RESOLVED) IMPLEMENTATION CAN BE NULL Note

Additional resources and comments

File Name : `MintClubFactory.sol`
File Location : `mint.club-contract/contracts`
└── `MintClubFactory.sol`
MD5 : 27a24a535edb50b46a974645f7891892

```
34      function updateTokenImplementation(address implementation) public onlyOwner {
35          tokenImplementation = implementation;
36      }
```

Details

`MintClubFactory` uses `tokenImplementation` for cloning. It needs to add null validation. Also, we recommend emitting Events to broadcast the changes.

Null validation and logging statements are add in fc1604ee

(RESOLVED) MAX SUPPLY SHOULD LARGER THAN ZERO Note

Additional resources and comments

File Name : `MintClubFactory.sol`
File Location : `mint.club-contract/contracts`
└── `MintClubFactory.sol`
MD5 : 27a24a535edb50b46a974645f7891892

```
50      function createToken(string memory name, string memory symbol, uint256 maxTokenSupply)
51          require(maxTokenSupply <= MAX_SUPPLY_LIMIT, 'MAX_SUPPLY_LIMIT_EXCEEDED');
52
53          address tokenAddress = _createClone(tokenImplementation);
```

```
69      function exists(address tokenAddress) external view
70          return maxSupply[tokenAddress] > 0;
71      }
```

Details

Tokens are allowing to be issued although the `maxTokenSupply` is zero value. The `exists` function implies that is checking the value is greater than zero, so tokens should be issued only and if the value is greater than zero. Accordingly, we recommend adding the condition logic to verify the value is positive.

Null validation is add in fc1604ee.

(RESOLVED) IMPLEMENTATION CAN BE NULL Note

Additional resources and comments

File Name : `MintClubBond.sol`
File Location : `mint.club-contract/contracts`
└── `MintClubBond.sol`
MD5 : 53f99ea5711d98a97066a6acb58fad84

```
42      function setDefaultBeneficiary(address beneficiary) external onlyOwner {
43          defaultBeneficiary = beneficiary;
44      }
```

Details

The taxes are collected to the `defaultBeneficiary`. But the address can be null. It needs to add null validation. Also, we recommend emitting Events to broadcast the changes.

Null validation and logging statements are add in fc1604ee

ADDITIONAL ANALYSIS RESULTS

Additional analysis results include a description of the main areas we looked at during the analysis.

ARITHMETIC ROUNDING ✓

Additional resources and comments

Details Abusing through arithmetic rounding and fund drain were analyzed, but none were found.

ARITHMETIC ISSUES ✓

Additional resources and comments

Details Arithmetic operations are using SafeMath

ERC-20 COMPATIBLE ✓

Additional resources and comments

Details Smart token issuing by the service comply with ERC-20 specifications.

CONCLUSION

The source code of the Mint Club developed by Bourbonshake is easy to read and very well organized. We have to remark that contracts are well architected and all the additional features are implemented. The team also has expertise in security that they can fix the vulnerability by themselves. **The detected vulnerabilities are as follows: Note 3. We have confirmed that all issues are resolved.** However, most of the codes are found out to be compliant with all the best practices. It is recommended to promote the stability of service through continuous code audit and analyze potential vulnerabilities.

Project	mint.club-contract
Commit #	fc737f4b
# of Files	7
# of Lines	608

File Tree

```
mint.club-contract/contracts
├── MintClubBond.sol ✓
├── MintClubFactory.sol ✓ ✓
├── MintClubToken.sol
├── lib
│   ├── ERC20Initializable.sol
│   └── Math.sol
├── mock
│   └── MintClubFactoryMock.sol
└── old
    └── MintToken.sol
```