BELT.FI SECURITY ASSESSMENT REPORT

FEB. 24 ~ MAR. 9, 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 Ozys's BELT smart contract from Feb. 24 until Mar. 9. 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 1. It is recommended to promote the stability of service through continuous code audit and analyze potential vulnerabilities.

ANALYSIS TARGET

The following projects were analyzed until Sep. 18.

Project	belt-contract
Commit #	16f3174a
# of Files	16
# of Lines	12,256

KEY AUDIT POINTS & PROCESS

BELT is an AMM protocol that incorporates multi-strategy yield optimizing on Binance Smart Chain (BSC). BELT team design and develop the system based on the BSC which is EVM compatible blockchain. Accordingly, we mainly reviewed common vulnerabilities in DeFi services and possible hacking scenarios.

For example, the following scenarios are included: draining the contract's funds, freezing funds, breaking pools. However, we did not take any internal hackings by administrators into account. Additionally, although not mentioned in this report, we would like to suggest the customer's interest in the stability of external services as well. Most 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

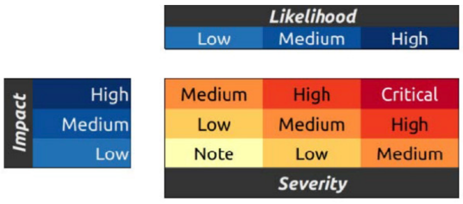
The followings are considered:

• 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.

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.



		Likelihood		
		Low	Medium	High
Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Note	Low	Medium
		Severity		

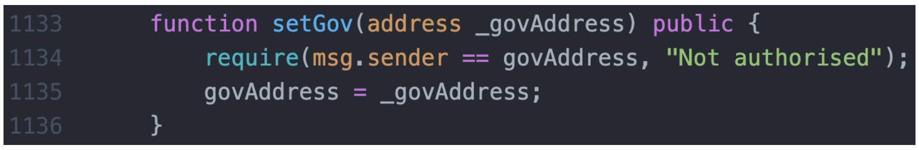
ANALYSIS RESULTS

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

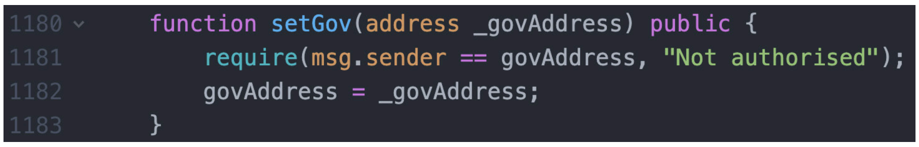
INSUFFICIENT INPUT VALIDATION Note

Additional resources and comments

File Name : `managerDataStorage.sol`
File Location : `belt-contract/contracts/earn/strategies`
```
└── Strategy*.sol
```



```
    function setGov(address _govAddress) public {
        require(msg.sender == govAddress, "Not authorised");
        govAddress = _govAddress;
    }
```



```
    function setGov(address _govAddress) public {
        require(msg.sender == govAddress, "Not authorised");
        govAddress = _govAddress;
    }
```

Details We have confirmed that the `setGov` functions that update `govAddress` in strategy-related contracts do not check. `govAddress` is an important account with critical privileges. The system can lead to unexpected situations because of the corrupted account, so it is recommended to validate the input value.

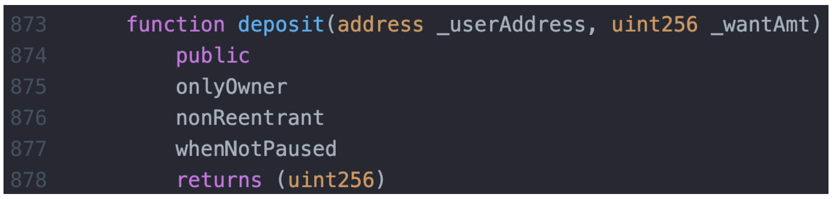
ADDITIONAL ANALYSIS RESULTS

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

REENTRANCY ✓

Additional resources and comments

File Name : `interestModel.sol`
File Location : `belt-contract/contracts/earn`
```
└── strategies/*.sol
└── tokens/*.sol
```



```
    function deposit(address _userAddress, uint256 _wantAmt)
        public
        onlyOwner
        nonReentrant
        whenNotPaused
        returns (uint256)
```

Details We analyzed the deposit and withdraw implementations of major contracts. It has been confirmed that the reentrancy that occurs mainly is protected through the guard statement.

ADDITIONAL ANALYSIS RESULTS

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

OVERFLOWS ✓

Additional resources and comments

File Name : `BeltToken.sol`
File Location : `belt-contract/contracts`
```
└── BeltToken.sol
```
MD5: 05fe2d8af27d963e2d71532eb030a03d

Details We analyzed the implementation of the `BeltToken` contract. We have confirmed that the theoretical integer overflow may occur, but it is unlikely to occur in practice.

BEP2 SPECIFICATIONS ✓

Additional resources and comments

File Name : `BeltLPToken.vy`
File Location : `belt-contract/contracts/swap/BeltLPToken.vy`
MD5: 2dd46daf622e2b2f426d336ee98b858d

Details We analyzed the implementation of the `BeltLPToken.vy` contract. We have confirmed that the contract complies with the BEP-2 proposals of the Binance Chain.

INFINITE APPROVE ✓

Additional resources and comments

File Name : `BeltLPToken.vy`
File Location : `belt-contract/contracts/swap/BeltLPToken.vy`
MD5: 2dd46daf622e2b2f426d336ee98b858d

```
@public
def transferFrom(_from : address, _to : address, _value : uint256) -> bool:
    self.balanceOf[_from] -= _value
    self.balanceOf[_to] += _value
    if msg.sender != self.minter:
        self.allowances[_from][msg.sender] -= _value
    log.Transfer(_from, _to, _value)
    return True
```

Details We analyzed the implementation of the `BeltLPToken.vy` contract. We found that the transfer can proceed regardless of allowance when the user is a minter. If it is not intended, we recommend to remove it.

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. Due to time limitations, we have partially covered the possibility of arithmetic rounding. Our preliminary results showed a negligible impact.

CONCLUSION

The source code of the BELT Finance 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 detected vulnerabilities are as follows: Note 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 service through continuous code audit and analyze potential vulnerabilities.

Project	belt-contract
Commit #	16f3174a
# of Files	16
# of Lines	12,256

File Tree

```
belt-contract/contracts
├── BeltToken.sol
├── BeltView.sol
├── earn
│   ├── MasterBelt.sol
│   ├── strategies  Note
│   │   ├── StrategyVenusBUSD.sol
│   │   ├── StrategyVenusDAI.sol
│   │   ├── StrategyVenusUSDC.sol
│   │   ├── StrategyVenusUSDT.sol
│   │   ├── VaultBPool.sol
│   │   └── VaultCakePool.sol
│   └── tokens
│       ├── bBUSD.sol
│       ├── bDAI.sol
│       ├── bUSDC.sol
│       └── bUSDT.sol
└── swap
    ├── BeltLPToken.vy
    ├── DepositB.vy
    ├── StableSwapB.vy
    └── pooldata.json
```