BIFI SECURITY ASSESSMENT REPORT

NOV. 13, 2020

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 PiLab's BiFi smart contract until Nov. 13. 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.

We scanned the target contract about known vulnerable codes from the SOOHO's signature database collected until Nov. 13, through SOOHO's Odin. We have also conducted a more diverse security vulnerability detecting process with useful security tools.

The detected vulnerabilities are as follows: Medium 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	20201022-pilab-audit
checksum	3808f009
# of Files	22
# of Lines	3,112

KEY AUDIT POINTS & PROCESS

BiFi is the DeFi service that services deposit & loan in which users deposit their virtual assets and loan other types of assets as collateral. PiLab team design and develop the system based on the Ethereum blockchain. Accordingly, we mainly reviewed common vulnerabilities in DeFi services and possible hacking scenarios.

For example, the following scenarios are included: access control, input validation. However, we did not take any internal hackings by administrators into account. In addition, the analysis was mad on the premise that the Oracle information is correct except in certain conditions.

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.

Impact	Likelihood: Low	Likelihood: Medium	Likelihood: High
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.

AUTHORIZATION THROUGH TX.ORIGIN (SWC-115) Medium

File Name : `managerDataStorage.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└─contracts/marketHandler/marketHandlerDataStorage`
`└─handlerDataStorage.sol`
MD5: d48fbb460ab93cbfc2e780484111b292

```
44    modifier onlyOwner {
45      require((msg.sender == owner) || (tx.origin == owner), "onlyOwner function");
46      _;
47    }
```

Details In the modifier `onlyOwner`, `tx.origin` is used for the access control. Using the variable for authorization could make a contract vulnerable if an authorized account calls into a malicious contract. A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.

Additional resources and comments

See the details from SWC-115 in the Smart Contract Weakness Classification and Test Cases

VERIFIED - PROXY ✓

File Name : `callProxy.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└─front/callProxy.sol`
MD5: 64ef85a8382e6be9a6c265d057c84890

Details We analyzed the implementation of the `callProxy` contract. We have confirmed that the functions return the correct value according to its name.

분석 결과에 대한 추가적인 자료 및 코멘트

VERIFIED - TOKEN HANDLER MODEL ✓

File Name : `tokenHandler.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└─marketHandler/tokenHandler.sol`
MD5: 15ce858ceaf8f14143e64dc62d8648f3

Details We analyzed the implementation of the `tokenHandler` contract. We intensively analyzed whether there is a possibility for exploit due to token type confusion or re-entry attacks similar to the recent Akropolis hack.

분석 결과에 대한 추가적인 자료 및 코멘트

ANALYSIS RESULTS

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

VERIFIED - LIQUIDATION ✓

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `liquidationManager.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└── marketManager/liquidationManager.sol`
MD5: bbb9abe46848bff5e7c1565341a7829e

Details We analyzed the implementation of the `liquidationManager` contract. We have confirmed that the partial liquidation and liquidation judgment functions operate correctly.

VERIFIED - INTEREST MODEL ✓

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `interestModel.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└── interestModel/interestModel.sol`
MD5: 09d33da59e228b11682e6ecf96b39a5f

Details We analyzed the implementation of the `interestModel` contract. We focused on the logic of calculating the Interest Delta and applying interest. It has been confirmed that there are no issues such as duplicate calculations or mistaken calculations which mainly occur.

VERIFIED - COIN HANDLER ✓

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `coinHandler.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└── marketHandler/coinHandler.sol`
MD5: 2bb0060b5c87ea25bd31193da96792b4

Details We analyzed the implementation of the `coinHandler`, `tokenHandler` and storage contracts. Deposits and withdrawals are operated safely. In addition, it was confirmed that underflow may occur theoretically when calculating the deposit margin value, but it cannot actually occur.

VERIFIED - TOKEN MANAGER ✓

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `tokenManager.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└── marketHandler/tokenManager.sol`
MD5: ed135d65b7c2ab415d6ed4df4d600519

Details We analyzed the implementation of the `coinHandlertokenManager` contract. However, if the price obtained through Oracle changes significantly, the same situation as the bZx Oracle case could occur. It is recommended to check the differences in the price.

ANALYSIS RESULTS

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

CONCLUSION

The source code of the PiLab's BiFi 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: 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 service through continuous code audit and analyze potential vulnerabilities.

Project	20201022-pilab-audit
checksum	3808f009
# of Files	22
# of Lines	3,112

File Tree

```
20201022-pilab-audit/2. truffle/contracts
├── front/callProxy.sol
├── interestModel/interestModel.sol
├── interfaces
│   ├── IERC20.sol
│   ├── interestModelInterface.sol
│   ├── managerDataStorageInterface.sol
│   ├── marketHandlerDataStorageInterface.sol
│   ├── marketHandlerInterface.sol
│   ├── marketManagerInterface.sol
│   ├── oracleInterface.sol
│   ├── oracleProxyInterface.sol
│   ├── safeMathInterface.sol
│   └── tokenInterface.sol
├── marketHandler
│   ├── coinHandler.sol
│   ├── marketHandlerDataStorage
│   │   └── handlerDataStorage.sol Medium
│   └── tokenHandler.sol
├── marketManager
│   ├── liquidationManager.sol
│   ├── managerDataStorage
│   │   └── managerDataStorage.sol
│   └── tokenManager.sol
├── oracle
│   ├── oracle.sol
│   └── oracleProxy.sol
├── safeMath/safeMath.sol
└── tokenStandard/token.sol
```