KAI PROTOCOL V2
SECURITY ASSESSMENT REPORT

AUG. 10 - AUG. 24, 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 KAI Protocol's updated smart contract from Aug. 10 to Aug. 24, 2021. Boost, Synthetics, and Governance is included in the updates. 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 Cube System 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 known vulnerable codes through SOOHO's Odin in contracts. We have also conducted a more diverse security vulnerability detecting process with useful security tools.

The detected vulnerabilities are as follows: High 1, and 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 from Aug. 10 to Aug. 24, 2021.

Project	kaiprotocol-contract-20210728
File Hash	89da843a
# of Files	54
# of Lines	5,766

KEY AUDIT POINTS & PROCESS

KAI Protocol is a algorithmic stable coin projects. This analysis is about newly added features of the V2 with vKai, Treasury V2, Boardroom V2, ksAssets, and so on. Accordingly, we mainly reviewed whether it is possible to manipulates synthetic assets, abused of vKai and possible hacking scenarios.

For example, the following scenarios are included: access control, input validation, token vesting logics, parameter validation. However, we did not take any internal hackings by administrators into account (e.g., Rug Pull). In addition, the design of protocol is also excluded from technical review.

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:

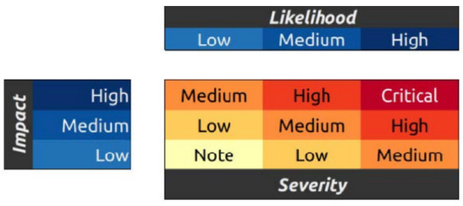
• Dynamic analysis through code execution.
• Examine possible financial gain by misusing detected vulnerabilities. (e.g., infinite withdrawal)
• Examine the possibility of adverse effect to the token service by misusing detected vulnerabilities. (e.g., Mint)

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.



Impact \ Likelihood	Low	Medium	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.

INITIALIZE CAN BE EXECUTE MULTIPLE TIMES High

Additional resources and comments

File Name : `TreasuryV2Impl.sol`
File Location : `kaiprotocol-contract-20210728`
`└── TreasuryV2Impl.sol`

```
    function initialize (
        address _kai,
        address _bkai,
        address _skai,
        uint256 _startTime
    ) external onlyAdmin {
```

Details Initialize function can be called multiple times. We recommend to add `initializer` or boolean variable that flags the initialization.

RECOMMEND TO EMIT EVENT Note

Additional resources and comments

File Name : `BoardroomV2Impl.sol`
File Location : `kaiprotocol-contract-20210728`
`└── BoardroomV2Impl.sol`

```
    function setTreasury(address _treasury) external onlyAdmin {
        require(_treasury != address(0), "address is zero");
        treasury = _treasury;
    }
```

Details The event does not emitted when `treasury` value changes. We recommend to fire the event when important variable changes.

ADDITIONAL ANALYSIS RESULTS

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

VERIFIED - VOTING PERIOD ✓

Additional resources and comments

```
    /// @notice The duration of voting on a proposal, in blocks
    function votingPeriod() public pure returns (uint) { return 60 * 60 * 24 * 7 / 1; }
```

Details It was confirmed that the period was well implemented as 7-day stated in the Medium announcement.

ADDITIONAL ANALYSIS RESULTS

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

VERIFIED - KAIVOTE ✓

File Name : `KAIVote.sol`
File Location : `kaiprotocol-contract-20210728/assets`

```
└── KAIVote.sol
```

Details We have confirmed that Kai Vote Token is well developed. The rights required for mint and burn are well managed, and the initial value is the same as the quantity stated in the notice.

Additional resources and comments

VERIFIED - GOVERNANCE ✓

File Name : `GovernorAlpha.sol`
File Location : `kaiprotocol-contract-20210728/governance`

```
└── GovernorAlpha.sol
```

Details We have confirmed that KAI Protocol Governor Alpha is well developed. We verified the voting conditions required for the proposal are not an issue. We have confirmed that proposals can be proposed and managed without problems.

Additional resources and comments

VERIFIED - SYNTHETICS ✓

File Name : `Kaitroller.sol`
File Location : `kaiprotocol-contract-20210728/governance`

```
└── Kaitroller.sol
```

Details We have confirmed that the controller and the ksAsset are well developed. The analysis is conducted based on documents from Synthetix and DeFiPieProtocol with known vulnerabilities.

Additional resources and comments

VERIFIED - MATHEMATICAL OPERATIONS ✓

Details We have confirmed that the mathematical operations are working well.

Additional resources and comments

VERIFIED - REENTRANCY ✓

Details We analyzed possible reentrancy attacks in the contracts.

Additional resources and comments

CONCLUSIONS

The source code of the KAI Protocol is easy to read and very well organized. We have to remark that contracts were well handling the possible situations, renaming ambiguous variables and writing test codes. Most of the codes are found out to be compliant with all the best practices. **The detected vulnerabilities are as follows: High 1, and Note 1.** It is recommended to promote the stability of service through continuous code audit and analyze potential vulnerabilities.

Project	kaiprotocol-contract-20210728
File Hash	89da843a
# of Files	54
# of Lines	5,766

```
kaiprotocol-contract-20210728
├── AirdropBoost.sol
├── AirdropOperator.sol
├── BBFund.sol
├── BBFundImpl.sol
├── BBFundStorage.sol
├── Boardroom.sol
├── BoardroomV2.sol
├── BoardroomV2Impl.sol  Note
├── BoardroomV2Storage.sol
├── OracleKlayswap.sol
├── Timelock.sol
├── TreasuryImpl.sol
├── TreasuryStorage.sol
├── TreasuryUni.sol
├── TreasuryV2Impl.sol High
├── assets
│   ├── KAI.sol
│   ├── KAIBond.sol
│   ├── KAIShare.sol
│   ├── KAIVote.sol
│   └── kERC20.sol
├── governance
│   └── GovernorAlpha.sol
├── interfaces
│   ├── IBoardroom.sol
│   ├── IBoardroomV2.sol
│   ├── IERC20Detailed.sol
│   ├── IKAIAsset.sol
│   ├── IKlayExchange.sol
│   ├── IKlayswapFactory.sol
│   ├── IKlayswapStore.sol
│   ├── IOracle.sol
│   └── ITreasury.sol
├── lib
│   ├── Babylonian.sol
│   ├── FixedPoint.sol
│   ├── Safe112.sol
│   ├── UQ112x112.sol
│   ├── UniswapV2Library.sol
│   └── UniswapV2OracleLibrary.sol
├── owner
│   ├── Operator.sol
│   └── Ownable.sol
├── synthetics
│   ├── KaiUnitroller.sol
│   ├── Kaitroller.sol
│   ├── KaitrollerInterface.sol
│   ├── KaitrollerStorage.sol
│   ├── PriceOracle.sol
│   ├── PriceOracleAlpha.sol
│   ├── SimplePriceOracle.sol
│   └── ksAsset.sol
└── utils
    ├── ContractGuard.sol
    └── Epoch.sol
```