BEOSIN
Web3 Security & Compliance

Kuant.ai

Smart Contract Security Audit

No. 202511251702

Nov 25th, 2025

SECURING BLOCKCHAIN ECOSYSTEM

WWW.BEOSIN.COM

Contents

Summary of Audit Results

After auditing, 2 Low risks , 1 Info were identified in the Kuant.ai project. Specific audit details will be presented in the Findings section. Users should pay attention to the following aspects when interacting with this project:

Low	Fixed : 0	Acknowledged: 2
Info	Fixed : 0	Acknowledged: 1

- **Project Description:**

This contract implements a centralized, perpetual contract margin pool where users can deposit margin (such as USDT), and withdrawals can only be triggered by a designated withdrawal administrator. The contract records each user's total deposits and withdrawals, and uses withdrawHash to prevent duplicate withdrawals. It also allows administrators to pool funds into a vault and modify key addresses such as administrator, transaction fee, and margin token addresses. The overall logic leans towards centralized control, with core functions including token custody, administrator-deducted transaction fees for withdrawals, and basic deposit/withdrawal record management.

1 Overview

1.1 Project Overview

Project Name	Kuant.ai
Project Language	Solidity
Platform	BNB Chain
Contract address	0xf6ae4e36a14da4be1988911d5e03544dc35dff3a

1.2 Audit Overview

Audit work duration: Nov 25, 2025 - Nov 25, 2025

Audit team: Beosin Security Team

1.3 Audit Method

The audit methods are as follows:

1. Formal Verification

Formal verification is a technique that uses property-based approaches for testing and verification. Property specifications define a set of rules using Beosin's library of security expert rules. These rules call into the contracts under analysis and make various assertions about their behavior. The rules of the specification play a crucial role in the analysis. If the rule is violated, a concrete test case is provided to demonstrate the violation.

2. Manual Review

Using manual auditing methods, the code is read line by line to identify potential security issues. This ensures that the contract's execution logic aligns with the client's specifications and intentions, thereby safeguarding the accuracy of the contract's business logic.

The manual audit is divided into three groups to cover the entire auditing process:

The Basic Testing Group is primarily responsible for interpreting the project's code and conducting comprehensive functional testing.

The Simulated Attack Group is responsible for analyzing the audited project based on the collected historical audit vulnerability database and security incident attack models. They identify potential attack vectors and collaborate with the Basic Testing Group to conduct simulated attack tests.

The Expert Analysis Group is responsible for analyzing the overall project design, interactions with third parties, and security risks in the on-chain operational environment. They also conduct a review of the entire audit findings.

3. Static Analysis

Static analysis is a function of examining code during compilation or static analysis to detect issues. Beosin-VaaS can detect more than 100 common smart contract vulnerabilities through static analysis, such as reentrancy and block parameter dependency. It allows early and efficient discovery of problems to improve code quality and security.

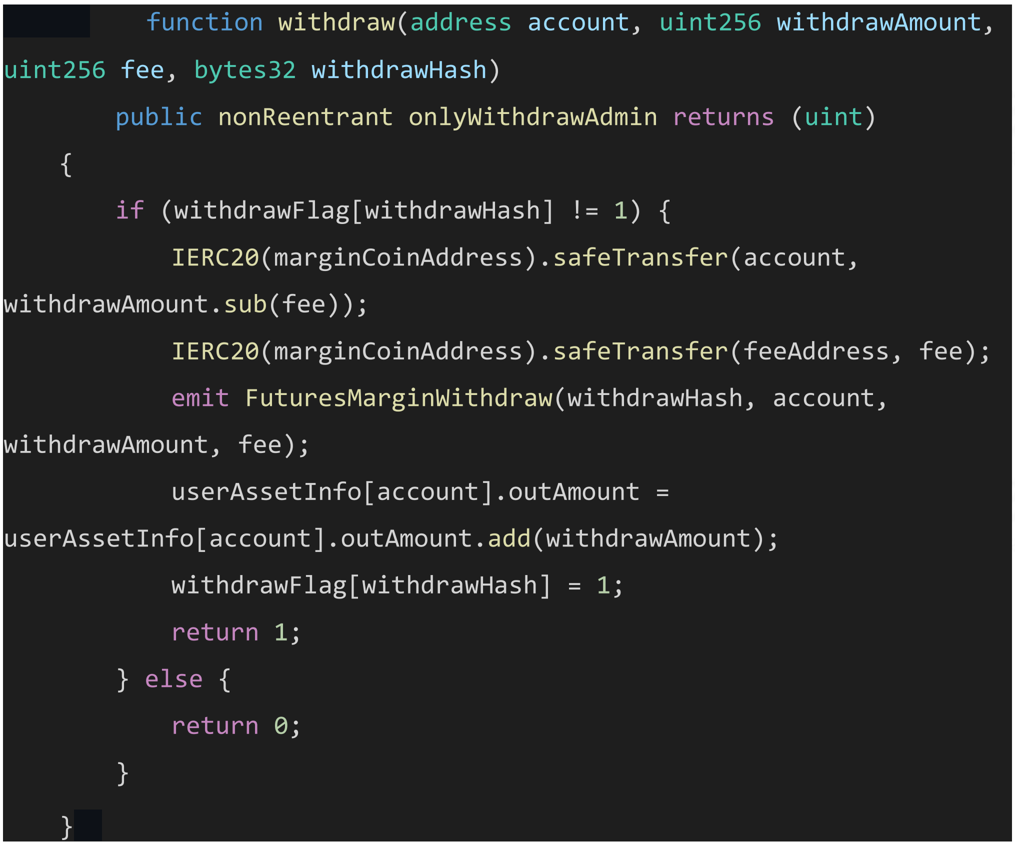
2 Findings

Index	Risk description	Severity level	Status
Kuant.ai-01	Excessive centralized authority	Low	Acknowledged
Kuant.ai-02	Inconsistent extraction amount	Low	Acknowledged
Kuant.ai-03	Missing event trigger	Info	Acknowledged

Finding Details:

[Kuant.ai-01] Excessive centralized authority

Severity Level	Low
Type	Business Security
Lines	Kuant.ai.sol#L686-708
Description	In the Kuant.ai contract, withdrawals can only be executed by the `withdrawAdmin` address, which is currently an EOA. If the private key of this EOA is compromised, all user funds deposited in the contract could be stolen. Additionally, the admin account has unrestricted access to the `withdrawAdminFun` function, allowing it to transfer the entire contract balance to the vaults address at any time. These two roles possess excessive centralized privileges, creating significant single-point-of-failure risks for user funds.



```
     function withdraw(address account, uint256 withdrawAmount,
uint256 fee, bytes32 withdrawHash)
        public nonReentrant onlyWithdrawAdmin returns (uint)
    {
        if (withdrawFlag[withdrawHash] != 1) {
            IERC20(marginCoinAddress).safeTransfer(account,
withdrawAmount.sub(fee));
            IERC20(marginCoinAddress).safeTransfer(feeAddress, fee);
            emit FuturesMarginWithdraw(withdrawHash, account,
withdrawAmount, fee);
            userAssetInfo[account].outAmount =
userAssetInfo[account].outAmount.add(withdrawAmount);
            withdrawFlag[withdrawHash] = 1;
            return 1;
        } else {
            return 0;
        }
    }
```

Recommendation	It is recommended to use a multisignature wallet for managing access addresses.
Status	**Acknowledged.**

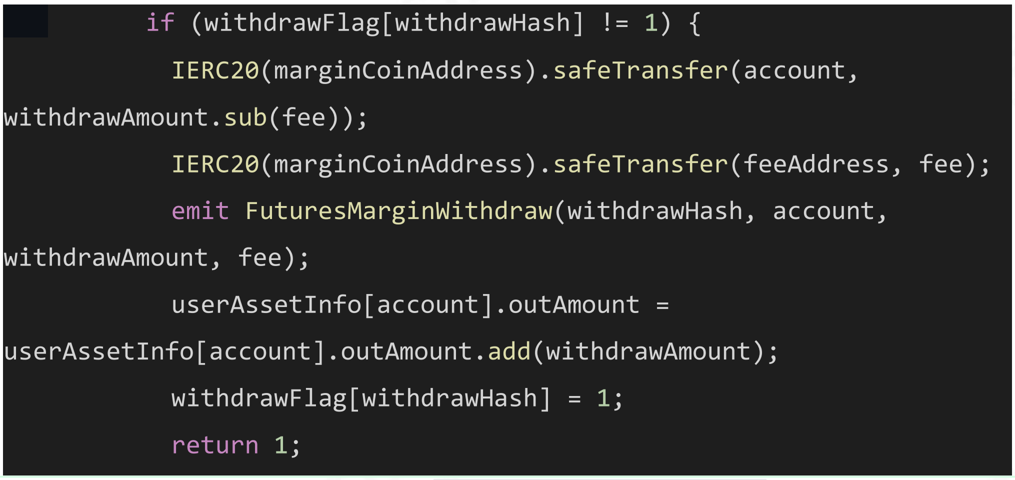
[Kuant.ai-02] Inconsistent extraction amount

Severity Level Low

Type Business Security

Lines Kuant.ai.sol#L696

Description The contract should store the user's actual withdrawn amount (net of fees), so `outAmount` must be updated with `withdrawAmount.sub(fee)` rather than `withdrawAmount`.



```
        if (withdrawFlag[withdrawHash] != 1) {
            IERC20(marginCoinAddress).safeTransfer(account,
withdrawAmount.sub(fee));
            IERC20(marginCoinAddress).safeTransfer(feeAddress, fee);
            emit FuturesMarginWithdraw(withdrawHash, account,
withdrawAmount, fee);
            userAssetInfo[account].outAmount =
userAssetInfo[account].outAmount.add(withdrawAmount);
            withdrawFlag[withdrawHash] = 1;
            return 1;
```

Recommendation It is recommended to change it to `withdrawAmount.sub(fee)`.

Status **Acknowledged.**

[Kuant.ai-03] Missing event trigger

Severity Level	Info
Type	Coding Conventions
Lines	Kuant.ai.sol#L710-730
Description	In the Kuant.ai contract, the following function does not trigger an event when modifying important parameters.

```
    function modifyMarginAddress(address _marginCoinAddress) public onlyAdmin {
        require(_marginCoinAddress != address(0), "FuturesMarginPool/MARGIN_COIN_ERROR");
        marginCoinAddress = _marginCoinAddress;
    }
    function modifyWithdrawAdmin(address _withdrawAdmin) public onlyAdmin {
        require(_withdrawAdmin != address(0), "FuturesMarginPool/WITHDRAW_ADMIN_ERROR");
        withdrawAdmin = _withdrawAdmin;
    }
    function modifyVaultsAddress(address _vaults) public onlyAdmin {
        require(_vaults != address(0), "FuturesMarginPool/VAULTS_ERROR");
        vaults = _vaults;
    }
    function modifyFeeAddress(address _feeAddress) public onlyAdmin {
        require(_feeAddress != address(0), "FuturesMarginPool/FEE_ADDRESS_ERROR");
        feeAddress = _feeAddress;
    }
    function modifyAdmin(address _admin) public onlyAdmin {
        require(_admin != address(0), "FuturesMarginPool/ADMIN_ERROR");
        admin = _admin;
    }
```

Recommendation	It is recommended to add event triggering to the above functions.
Status	**Acknowledged.**

3 Appendix

3.1 Vulnerability Assessment Metrics and Status in Smart Contracts

3.1.1 Metrics

In order to objectively assess the severity level of vulnerabilities in blockchain systems, this report provides detailed assessment metrics for security vulnerabilities in smart contracts with reference to CVSS 3.1(Common Vulnerability Scoring System Ver 3.1).

According to the severity level of vulnerability, the vulnerabilities are classified into four levels: "critical", "high", "medium" and "low". It mainly relies on the degree of impact and likelihood of exploitation of the vulnerability, supplemented by other comprehensive factors to determine of the severity level.

Impact Likelihood	Severe	High	Medium	Low
Probable	Critical	High	Medium	Low
Possible	High	Medium	Medium	Low
Unlikely	Medium	Medium	Low	Info
Rare	Low	Low	Info	Info

3.1.2 Degree of impact

- **Critical**

Critical impact generally refers to the vulnerability can have a serious impact on the confidentiality, integrity, availability of smart contracts or their economic model, which can cause substantial economic losses to the contract business system, large-scale data disruption, loss of authority management, failure of key functions, loss of credibility, or indirectly affect the operation of other smart contracts associated with it and cause substantial losses, as well as other severe and mostly irreversible harm.

- **High**

High impact generally refers to the vulnerability can have a relatively serious impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a greater economic loss, local functional unavailability, loss of credibility and other impact to the contract business system.

- **Medium**

Medium impact generally refers to the vulnerability can have a relatively minor impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a small amount of economic loss to the contract business system, individual business unavailability and other impact.

- **Low**

Low impact generally refers to the vulnerability can have a minor impact on the smart contract, which can pose certain security threat to the contract business system and needs to be improved.

3.1.3 Likelihood of Exploitation

- **Probable**

Probable likelihood generally means that the cost required to exploit the vulnerability is low, with no special exploitation threshold, and the vulnerability can be triggered consistently.

- **Possible**

Possible likelihood generally means that exploiting such vulnerability requires a certain cost, or there are certain conditions for exploitation, and the vulnerability is not easily and consistently triggered.

- **Unlikely**

Unlikely likelihood generally means that the vulnerability requires a high cost, or the exploitation conditions are very demanding and the vulnerability is highly difficult to trigger.

- **Rare**

Rare likelihood generally means that the vulnerability requires an extremely high cost or the conditions for exploitation are extremely difficult to achieve.

3.1.4 Fix Results Status

Status	Description
Fixed	The project party fully fixes a vulnerability.
Partially Fixed	The project party did not fully fix the issue, but only mitigated the issue.
Acknowledged	The project party confirms and chooses to ignore the issue.

3.2 Audit Categories

No.	Categories	Subitems
1	Coding Conventions	Compiler Version Security
		Deprecated Items
		Redundant Code
		require/assert Usage
		Gas Consumption
2	General Vulnerability	Integer Overflow/Underflow
		Reentrancy
		Pseudo-random Number Generator (PRNG)
		Transaction-Ordering Dependence
		DoS (Denial of Service)
		Function Call Permissions
		call/delegatecall Security
		Returned Value Security
		tx.origin Usage
		Replay Attack
		Overriding Variables
		Third-party Protocol Interface Consistency
3	Business Security	Business Logics
		Business Implementations
		Manipulable Token Price
		Centralized Asset Control
		Asset Tradability
		Arbitrage Attack

Beosin classified the security issues of smart contracts into three categories: Coding Conventions, General Vulnerability, Business Security. Their specific definitions are as follows:

- **Coding Conventions**

Audit whether smart contracts follow recommended language security coding practices. For example, smart contracts developed in Solidity language should fix the compiler version and do not use deprecated keywords.

- **General Vulnerability**

General Vulnerability include some common vulnerabilities that may appear in smart contract projects. These vulnerabilities are mainly related to the characteristics of the smart contract itself, such as integer overflow/underflow and denial of service attacks.

- **Business Security**

Business security is mainly related to some issues related to the business realized by each project, and has a relatively strong pertinence. For example, whether the lock-up plan in the code match the white paper, or the flash loan attack caused by the incorrect setting of the price acquisition oracle.

*Note that the project may suffer stake losses due to the integrated third-party protocol. This is not something Beosin can control. Business security requires the participation of the project party. The project party and users need to stay vigilant at all times.

3.3 Disclaimer

The Audit Report issued by Beosin is related to the services agreed in the relevant service agreement. The Project Party or the Served Party (hereinafter referred to as the "Served Party") can only be used within the conditions and scope agreed in the service agreement. Other third parties shall not transmit, disclose, quote, rely on or tamper with the Audit Report issued for any purpose.

The Audit Report issued by Beosin is made solely for the code, and any description, expression or wording contained therein shall not be interpreted as affirmation or confirmation of the project, nor shall any warranty or guarantee be given as to the absolute flawlessness of the code analyzed, the code team, the business model or legal compliance.

The Audit Report issued by Beosin is only based on the code provided by the Served Party and the technology currently available to Beosin. However, due to the technical limitations of any organization, and in the event that the code provided by the Served Party is missing information, tampered with, deleted, hidden or subsequently altered, the audit report may still fail to fully enumerate all the risks.

The Audit Report issued by Beosin in no way provides investment advice on any project, nor should it be utilized as investment suggestions of any type. This report represents an extensive evaluation process designed to help our customers improve code quality while mitigating the high risks in blockchain.

3.4 About Beosin

Beosin is a leading blockchain security and compliance technology company established in 2018. Being focused on blockchain ecosystem security and compliance, it has developed a product matrix including Beosin KYT, Beosin Trace, and Stablecoin Monitor, which have obtained international certifications such as ISO 27001 and SOC 2. Beosin's core products have been applied for over 70 intellectual property rights, and the company has participated in the development of multiple international standards related to blockchain security. It was among the first batch of enterprises selected for the Cyberport Incubation Programme. Its business covers professional code security audit services for blockchain ecosystems, anti-money laundering compliance technology services for exchanges, financial institutions, and payment institutions, and virtual asset tracing and investigation services for law enforcement and regulatory authorities.

As one of the earliest companies to apply formal verification to blockchain security, Beosin offers professional blockchain and smart contract security audit services. Beosin has audited over 4,500 smart contracts and blockchain projects and has become the official security partner for several renowned blockchains, including BNB Chain, TON, Soneium, Manta Network, Sonic SVM, and SOON Network.

BEOSIN
Web3 Security & Compliance

Official Website
https://www.beosin.com

Telegram
https://t.me/beosin

X
https://x.com/Beosin_com

Email
service@beosin.com

LinkedIn
https://www.linkedin.com/company/beosin