BEOSIN
Web3 Security & Compliance

jasper-vaults

Smart Contract Security Audit

No. 202411281630

Nov 28th, 2024



SECURING BLOCKCHAIN ECOSYSTEM

WWW.BEOSIN.COM

Contents

1 Overview 6

1.1 Project Overview 6

1.2 Audit Overview 6

1.3 Audit Method 6

2 Findings 8

[jasper-vaults-01] The VerifyOrder function verification error 9

[jasper-vaults-02] Lack of events 10

3 Appendix 11

3.1 Vulnerability Assessment Metrics and Status in Smart Contracts 11

3.2 Audit Categories 14

3.3 Disclaimer 16

3.4 About Beosin 17

Summary of Audit Results

After auditing,1 Medium, 1 Info item was identified in the jasper-vaults project. Specific audit details will be presented in the Findings section. Users should pay attention to the following aspects when interacting with this project:

Medium	Fixed : 1 Acknowledged: 0
Info	Fixed : 1 Acknowledged: 0

- **Project Description:**

Jasper Vault is a peer-to-peer Bitcoin options trading platform designed to provide a safe, simple, and efficient way for users to trade with leverage without the worry of liquidation. The platform offers flexible options, including 30-minute trades via its Telegram Mini App, 2-hour expirations for degen options, and user-adjustable 0DTE (zero days to expiration) in advanced mode. Operating on multiple chains, Jasper Vault enhances the trading experience with NFTs and GameFi elements like the Moonlight Box and Infinity Stones, which boost user engagement and flexibility. By enabling direct trading from self-managed wallets, Jasper Vault ensures security, transparency, and a seamless experience for both beginners and advanced traders.

This audit focuses specifically on the OptionModuleV2, OptionService, and OptionLiquidateService contracts within the Jasper Vault project.

➢ OptionModuleV2:

An upgradable options management contract that supports the creation and management of call/put option orders, order signature verification, fee settlement, asset transfers, and other related features. Users can pay premiums (premiumAsset) using ETH or ERC20 tokens and access flexible options configuration and querying capabilities. The contract includes management for price oracle whitelists and fee discount whitelists, allowing users holding specified NFTs to receive fee discounts. Option fees are distributed proportionally at settlement, with a portion paid to the option writer and the remaining portion allocated as platform fees.

➢ OptionService:

This contract handles the creation and management of option orders through OptionModuleV2. It validates orders, locks assets, and ensures secure transfers. The contract also enables liquidation functionality, invoking the OptionLiquidateService contract during the liquidation process.

➢ OptionLiquidateService:

This contract manages the liquidation of options, including exercise, non-exercise, or profit settlement. It supports various settlement modes, such as physical delivery and profit settlement, to process options trades. The contract ensures that all assets (e.g., ETH, tokens, NFTs) are transferred according to specified rules during liquidation. Additional features include maximum asset loss limits, whitelist

validation, and price verification. The contract integrates with external platforms such as the Pyth price oracle and LiquidateHelper for real-time price verification and liquidation validation.

1 Overview

1.1 Project Overview

Project Name	jasper-vaults
Project language	Solidity
Platform	Arbitrum,BASE,Bitlayer,Sei Chain
Base code	https://github.com/jaspervault/jasper-vaults-contracts
Audit Scope	OptionModuleV2.sol OptionService.sol OptionLiquidateService.sol
Commit	f0f08d6280d8196e64811cd4213831fba27df85c(initial) 4aae5cb4ecb31a789c5321e2ed32bc15e60a0ef1 7c0c59c0f849e31c5670815e8a8575ad3b72dd83(final)

1.2 Audit Overview

Audit work duration: Nov 25, 2024 – Nov 28, 2024

Update time: Dec 3, 2024

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 method 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
jasper-vaults-01	The VerifyOrder function verification error	Medium	Fixed
jasper-vaults-02	Lack of events	Info	Fixed

Finding Details:

[jasper-vaults-01] The VerifyOrder function verification error

Severity Level	Medium
Type	Business Security
Lines	OptionService.sol#L215
Description	In the `verifyOrder` function of the `OptionService` contract, the value of `optionLockAmount` is first calculated based on quantity and `lockAmount`. The function then checks the type of locked asset: for platform coins or ERC20 tokens, it verifies that the writer's balance is sufficient to fulfill the `optionLockAmount`; for NFT-type locked assets, the current verification relies only on the `lockAmount`, which may result in the user only having to provide an NFT that fulfills the `lockAmount` requirement, thus bypassing the balance verification. This may result in users bypassing the balance validation by simply providing an NFT that meets the `lockAmount` requirement.
Recommendation	It is recommended to modify `lockAmount` to `optionLockAmount` for better clarity and consistency.
Status	**Fixed.** The project owner has changed the parameter to `optionLockAmount`.

```
require(
    uint256(liquidity) >= _verifyOrder.lockAmount,
    "OptionService:lockAmount not enough"
);
```

[jasper-vaults-02] Lack of events

Severity Level	Info
Type	Coding Conventions
Lines	OptionService.sol#L45-50
Description	When modifying important parameters in a contract, it is recommended to trigger the corresponding event for transparent and traceable recording on the blockchain. `function setPriceOracle(IPriceOracle _priceOracle) external onlyOwner{ priceOracle=_priceOracle; } function setLiquidateSerivce(IOptionLiquidateService _addr) external onlyOwner{ liquidateSerivce = _addr; }`
Recommendation	It is recommended to add event triggering to the corresponding functions.
Status	**Fixed**. The project has been deleted.

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.

Likelihood \ Impact	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	Deprecated Items
		Redundant Code
		require/assert Usage
		Default Values
2	General Vulnerability	Insufficient Address Validation
		Lack Of Address Normalization
		Variable Override
		DoS (Denial Of Service)
		Function Call Permissions
		Call/Delegatecall Security
		Tx.origin Usage
		Returned Value Security
		Mathematical Risk
		Overriding Variables
3	Business Security	Business Logics
		Business Implementations
		Manipulable Token Price
		Centralized Asset Control
		Arbitrage Attack
		Access Control

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 Rust 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.

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 the first institution in the world specializing in the construction of blockchain security ecosystem. The core team members are all professors, postdocs, PhDs, and Internet elites from world-renowned academic institutions. Beosin has more than 20 years of research in formal verification technology, trusted computing, mobile security and kernel security, with overseas experience in studying and collaborating in project research at well-known universities. Through the security audit and defense deployment of more than 2,000 smart contracts, over 50 public blockchains and wallets, and nearly 100 exchanges worldwide, Beosin has accumulated rich experience in security attack and defense of the blockchain field, and has developed several security products specifically for blockchain.

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/