BEOSIN
Blockchain Security

Hypezion Finance

Smart Contract Security Audit

No. 202511241435

Dec 3rd, 2025

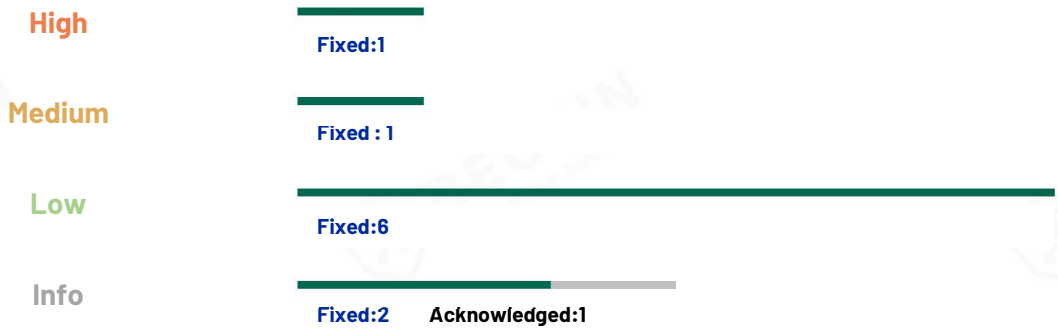
SECURING BLOCKCHAIN ECOSYSTEM

WWW.BEOSIN.COM

Contents

Summary of Audit Results

After auditing, 1 High-risk, 1 Medium-risk, 6 Low-risks, 3 Info items were identified for the Hypezion Finance project. Specific audit details will be presented in the Findings section. Users should pay attention to the following aspects when interacting with this project:



- **Project Description:**

The audited project is an innovative decentralized finance ecosystem built on HypeEVM — Hypezion Finance, comprising a Staking-Swap System and a Stability Maintenance System. The protocol achieves secure and reliable asset management and yield distribution through an upgradable smart contract architecture.

The Staking-Swap System serves as the core of the entire ecosystem, allowing users to stake HYPE tokens to mint the protocol's flagship assets: hzUSD and bullHYPE.

- hzUSD is a USD-pegged stablecoin designed to maintain a strict 1:1 peg with the US dollar, with its circulating supply tightly controlled through regulated minting and burning mechanisms. As the foundational asset of the ecosystem, hzUSD provides a stable medium of exchange and store of value.
- bullHYPE is the protocol's leveraged token, offering users who seek higher risk-adjusted returns an amplified investment vehicle. Holders of bullHYPE benefit from leveraged exposure, earning magnified gains when the price of HYPE rises.

Meanwhile, holders of hzUSD enjoy the stability-backed attributes of the peg, supported by the HYPE collateral staked in the pool and algorithmic interventions from the Stability Maintenance System.

The StakedHzUSD contract, the core yield-generating component, is implemented as an ERC-4626-compliant vault. Beyond standard deposit and withdrawal functionality, it introduces a unique intervention mechanism: when the system's collateralization ratio falls below a predefined threshold, it automatically triggers an asset conversion protection process to safeguard the overall stability of the Staking-Swap System. Users who stake hzUSD not only earn staking rewards but also provide liquidity for asset conversion, thereby contributing to the stability of the entire ecosystem.

The system integrates multiple oracle adapters (HyperCore, Hyperlend, and RedStone) to aggregate price feeds from diverse sources, ensuring accurate and reliable pricing data. The core Exchange contract incorporates a comprehensive set of risk-management parameters, including collateralization ratio monitoring, dynamic fee model adjustments, and emergency pause functionality, enabling the protocol to operate robustly under all market conditions.

Through this carefully engineered suite of components, Hypezion Finance delivers a secure, efficient, and innovative DeFi yield environment that seamlessly combines the characteristics of stablecoins and leveraged tokens, catering to the investment needs of users with varying risk appetites.

1 Overview

1.1 Project Overview

Project Name	Hypezion Finance
Project Language	Solidity
Platform	HyperEVM
Github Link	https://github.com/zion-lab-pjt/hypezion-contract

1.2 Audit Overview

Audit work duration:Nov 03, 2025 - Dec 03, 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 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
Hypezion Finance-01	Incorrect Withdrawal ID Query	High	Fixed
Hypezion Finance-02	Fee Unit Error	Medium	Fixed
Hypezion Finance-03	Collateral Bad Debt Issue	Low	Fixed
Hypezion Finance-04	Fees Cannot Be Withdrawn	Low	Fixed
Hypezion Finance-05	Upside Slippage Limit Error	Low	Fixed
Hypezion Finance-06	Recovery Mode Issue	Low	Fixed
Hypezion Finance-07	Reserve Amount Calculation Error	Low	Fixed
Hypezion Finance-08	stakedAmounts Calculation Error	Low	Fixed
Hypezion Finance-09	Deduction Logic Issue	Info	Fixed
Hypezion Finance-10	Uncallable Interfaces	Info	Acknowledged
Hypezion Finance-11	Redundant Code	Info	Fixed

Finding Details:

[Hypezion Finance-01] Incorrect Withdrawal ID Query

Severity Level	High
Type	Business Security
Location	KinetiqIntegration.sol#L91
Description	In the KinetiqIntegration contract, the recorded nextWithdrawalId starts from 1 and increments, whereas in the StakingManager contract, nextWithdrawalId has no initial value set and each staker's ID starts from 0 and increments independently. As a result, using the nextWithdrawalId recorded in KinetiqIntegration for queries will retrieve incorrect withdrawal data.
Recommendation	It is recommended to use the nextWithdrawalId recorded in the StakingManager contract for queries.
Status	**Fixed.**

```
function initialize() public initializer {
    __AccessControl_init();
    __ReentrancyGuard_init();
    __UUPSUpgradeable_init();
    _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
    _grantRole(OPERATOR_ROLE, msg.sender);
    nextWithdrawalId = 1; // Start withdrawal IDs at 1
}
```

[Hypezion Finance-02] Fee Unit Error

Severity Level	**Medium**
Type	Business Security
Location	HypeZionExchange.sol#L377-380

Description	In the HypeZionExchange contract, the `mintLevercoin` function calculates the accumulated value of accumulatedFees using `getZhypeNavInHYPE`, resulting in a unit denominated in HYPE rather than hzUSD. However, in the subsequent `collectFees` function, when the admin claims the fees, accumulatedFees is directly minted as hzUSD, leading to a fee unit mismatch. A similar issue exists in the `_executeRedeem` function.
	`zhypeMinted = (amountHYPE * PRECISION) / navBefore;`
	`uint256 fee = (zhypeMinted * feeBps) / BASIS_POINTS;`
	`zhypeMinted -= fee;`
	`accumulatedFees += (fee * navBefore) / PRECISION; // Convert fee to HYPE value`

Recommendation	It is recommended to unify the fee unit to hzUSD instead of HYPE.
Status	**Fixed.** The project team has unified the unit used for charging fees.

[Hypezion Finance-03] Collateral Bad Debt Issue

Severity Level	Low
Type	Business Security
Location	HypeZionExchange.sol#L785-789
Description	In the HypeZionExchange contract, totalHYPECollateral only increases when hzUSD and bullHYPE are minted. When users redeem/withdraw bullHYPE or hzUSD, the contract deducts the corresponding amount of collateral. Under the current mechanism, profits generated by the exchange are stored in the stable pool by minting hzUSD. However, if this minting is not performed in a timely manner, when a user redeems bullHYPE, they can claim a portion of the profit without a corresponding increase in collateral, leading to collateral bad debt.

```
        if (isZusd) {
            uint256 availableReserves = getAvailableReserveInHYPE();
            uint256 zusdLiabilities = getZusdLiabilitiesInHYPE();
            uint256 freeReserves = availableReserves > zusdLiabilities ?
                availableReserves - zusdLiabilities : 0;
            if (hypeAmount > freeReserves) {
                revert InsufficientReserves(hypeAmount, freeReserves);
            }
        } else {
            if (hypeAmount > totalHYPECollateral) {
                revert InsufficientReserves(hypeAmount,
totalHYPECollateral);
            }
        }
```

Recommendation	It is recommended to before calling `_executeRedeem`, settle/compound the vault's accrued yield first, or remove the total collateral cap restriction on user redemptions.
Status	**Fixed.**

[Hypezion Finance-04] Fees Cannot Be Withdrawn

Severity Level	Low
Type	Business Security
Location	HypeZionExchange.sol#L1022-1027
Description	In the HypeZionExchange contract, the `_executeSwapRedeem` function does not accumulate fees into accumulatedFees during fee calculation, and the contract lacks an admin interface to withdraw HYPE fees. As a result, fees generated via `_executeSwapRedeem` are locked in the contract and cannot be extracted.
Recommendation	It is recommended to accumulate fees into accumulatedFees, or record fees and add an admin withdrawal interface.
Status	**Fixed.** The project team has accumulated the fees into accumulatedFees.

```
        // Apply SwapRedeem fee (default 5% = 500 bps)
        uint256 fee = (hypeSwapped * swapRedeemFeeBps) / BASIS_POINTS;
        uint256 netHype = hypeSwapped - fee;
        // Update protocol balances
        totalKHYPEBalance -= khypeNeeded;
```

[Hypezion Finance-05] Upside Slippage Limit Error

Severity Level	Low
Type	Business Security
Location	HypeZionExchange.sol#L564-568

Description	In the HypeZionExchange contract, the restriction on expectedHypeFromKinetiq causes the slippage of expectedHype to only allow a 10% upward float. However, expectedHype should only enforce a minimum HYPE output and not cap the maximum output. This prevents users from successfully swapping if they cannot precisely specify minHypeOut.
	`if (expectedHypeFromKinetiq > expectedHype) {`
	`diff = expectedHypeFromKinetiq - expectedHype;`
	`} else {`
	`diff = expectedHype - expectedHypeFromKinetiq;`
	`}`

Recommendation	It is recommended to restrict only downside slippage.
Status	**Fixed.** The project team has removed the upward-floating slippage.

[Hypezion Finance-06] Recovery Mode Issue

Severity Level	Low
Type	Business Security
Location	HypeZionExchange.sol#L708-718

Description	In the HypeZionExchange contract, `exitRecoveryMode` only checks the system collateralization ratio before exiting recovery mode and does not verify it afterward. Additionally, the function can be called by anyone, allowing malicious actors to prevent the contract from returning to a healthy collateralization ratio.
	`// Burn hzHYPE from stability pool`
	`zhype.burn(address(stabilityPool), zhypeInPool);`
	`// Mint hzUSD to stability pool`
	`zusd.mint(address(stabilityPool), zusdToMint);`
	`// Update stability pool accounting`
	`stabilityPool.exitRecoveryMode(zhypeInPool, zusdToMint);`
	`// Update system state`
	`_updateSystemState();`

Recommendation	It is recommended to check the system collateralization ratio both before and after exiting recovery mode to ensure a healthy state post-exit.
Status	**Fixed.** The project team has added a system collateral ratio validation after the function call.

[Hypezion Finance-07] Reserve Amount Calculation Error

Severity Level	Low
Type	Business Security
Location	HypeZionExchange.sol#L351-370
Description	In the HypeZionExchange contract, the `mintLevercoin` function deducts the pre-transferred amountHYPE prematurely when calculating NAV via `getZhypeNavInHYPE`. However, `getZhypeNavInHYPE` uses the contract's totalKHYPE reserves for NAV computation and has no relation to HYPE. This premature deduction causes the reserve to appear smaller than actual, potentially triggering a revert.
Recommendation	It is recommended not to deduct amountHYPE during NAV calculation.
Status	**Fixed.** The project team has removed the calculation logic for amountHYPE.

```
    function mintLevercoin(uint256 amountHYPE) external payable
nonReentrant whenNotPaused returns (uint256 zhypeMinted) {
        if (msg.value != amountHYPE) revert IncorrectHYPEAmount();
        // Check our protocol minimum
        if (amountHYPE < minimumAmounts.mintHypeMin) revert
BelowMinimumAmount();
        // Also check Kinetiq minimum (if different)
        uint256 kinetiqMin = kinetiq.getMinStakingAmount();
        if (amountHYPE < kinetiqMin) {
            revert MinimumStakingAmountNotMet(amountHYPE, kinetiqMin);
        }
        // Check maximum total deposit limit
        uint256 newTotal = totalHypeDeposited + amountHYPE;
        if (newTotal > maxTotalDeposit) {
            revert MaximumDepositExceeded(newTotal, maxTotalDeposit);
        }
        // Calculate pre-deposit NAV (excludes current deposit)
        uint256 navBefore = getZhypeNavInHYPE(true, amountHYPE);
```

[Hypezion Finance-08] stakedAmounts Calculation Error

Severity Level	Low
Type	Business Security
Location	KinetiqIntegration.sol#L239-246

Description	In the KinetiqIntegration contract, withdrawalKHYPEAmounts records the kHYPE amount after fee deduction when queuing unstaking via queueUnstakeHYPE. However, in the subsequent `claimUnstake` function, stakedAmounts is reduced directly by the recorded withdrawalKHYPEAmounts, leaving the fee portion undeducted from stakedAmounts.
	`uint256 kHYPEAmount = withdrawalKHYPEAmounts[withdrawalId];`
	`if (stakedAmounts[hypeNovaExchange] >= kHYPEAmount) {`
	`stakedAmounts[hypeNovaExchange] -= kHYPEAmount;`
	`}`
	`// Clear withdrawal tracking`
	`delete withdrawalKHYPEAmounts[withdrawalId];`
	`delete withdrawalKHYPEFees[withdrawalId];`

Recommendation	It is recommended to deduct both the principal and the fee portion from stakedAmounts simultaneously.
Status	**Fixed.** The project team now deducts the fee portion from stakedAmounts at the same time.

[Hypezion Finance-09] Deduction Logic Issue

Severity Level	Info
Type	Business Security
Location	HypeZionExchange.sol#L1031-1033
Description	In the HypeZionExchange contract, the `claimRedemption` function calculates totalHYPECollateral. If totalHYPECollateral is less than hypeReceived, no deduction occurs and the if check is skipped, rendering the totalHYPECollateral restriction ineffective and preventing it from ever reaching zero. Similar if logic flaws exist in multiple places. `if (totalHYPECollateral >= hypeEquivalent) { totalHYPECollateral -= hypeEquivalent; }`
Recommendation	It is recommended that, when totalHYPECollateral is insufficient, it be set to zero instead of being ignored, or a revert be added to trigger an error and rollback.
Status	**Fixed.** The project team has added handling for the "less than" scenario.

[Hypezion Finance-10] Uncallable Interfaces

Severity Level	Info
Type	Coding Convention
Location	StakedHzUSD.sol#L182-190 ,L345-353
Description	In the StakedHzUSD contract, due to the absence of corresponding interface implementations in Protocol, the caller of `compoundYield` can never be protocol. A similar issue affects `revertIntervention`.
Recommendation	It is recommended to add the relevant interfaces to protocol, or modify the Protocol call conditions accordingly.
Status	**Acknowledged.**

[Hypezion Finance-11] Redundant Code

Severity Level	Info
Type	Coding Convention
Location	KinetiqYieldManager.sol#L470-522
Description	In the KinetiqYieldManager contract, the functions `_updateNAVInternal` and `_calculateAccruedYield` are currently unused and are internal functions. This constitutes redundant code that increases the contract's gas cost.
Recommendation	It is recommended to remove the redundant code.
Status	**Fixed.**

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

- **Severe**

Severe 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
Blockchain Security

Official Website
https://www.beosin.com

Telegram
https://t.me/beosin

Twitter
https://twitter.com/Beosin_com

Email
service@beosin.com