BEOSIN
Blockchain Security

haya finance

Smart Contract Security Audit

No. 202405011013

May 1st, 2024

SECURING BLOCKCHAIN ECOSYSTEM

WWW.BEOSIN.COM

Contents

Summary of Audit Results

After auditing, 2 Low-risk and 1 Info were identified in the haya 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:

Business overview

The haya finance consists of multiple contracts, as follows：

`Controller`: The Controller contract manages all system contracts such as `setTokens`, modules, factories, resources (like price oracles), and protocol fee configurations in this protocol.

`IntegrationRegistry`: The `IntegrationRegistry` contract is used to manage adapters of modules. In the current version, `Integrations` has no other business linkages, and modules will be added in the future to connect the functions of this code.

`PriceOracle`: The `PriceOracle` contract will provide the function of querying token prices. There are currently three ways to obtain: (1) Obtain directly using the oracle contract of the token pair; (2) Convert the relative price by obtaining the token and `MasterQuote` prices; (3) Get the token price from the adapter list.

`SetToken`: The `SetToken` contract is an ERC20 token, but it has expanded some functions: (1) Multiple Components can be added, each Component can have a `DefaultPosition` and multiple `ExternalPositions`, and they have different `virtualUnits` and data; (2) Multiple Components can be added Module contracts, these Modules can implement different functions to manage the assets and system parameters in the `SetToken` contract.

`SetTokenCreator`: `SetTokenCreator` is a factory contract used to create `SetToken` contracts. When this contract creates a `SetToken` contract, it will record the corresponding address into the Controller contract.

`SetValuer`: The `SetValue` contract has only one function, `calculateSetTokenValuation`, which is used to calculate the valuation of `SetToken`. Specifically, it will count the total value of all components of `SetToken`.

`AuctionRebalanceModule`: This is a rebalance auction contract. The administrator can set rebalance components and amounts (including components and rewards that need to be spent) and start the auction. Users can participate in the auction and send set tokens to increase the bidding price. This mechanism uses price In the priority auction mode, when the manager of the Set token contract announces that the auction is valid, the winner of the auction will be automatically determined in order from high to low bids. Regardless of whether the participating users win the auction, they need to call the claim function to claim the assets. Users who do not win the bid will have the bidding assets

returned in full, and the winning users will receive the rewards. During this process, both the number of Component tokens and the number of Set tokens in the Set token contract will be rebalanced.

`BasicIssuanceModule`: This is a module for converting between Set token and components. In this contract, users can deposit components to obtain Set tokens, or destroy Set tokens to obtain tokens corresponding to components. Of course, the exchange ratio is determined by the `virtualUnit` in the Set token contract.

1 Overview

1.1 Project Overview

Project Name	haya finance
Project Language	Solidity
Platform	Arbitrum
Code Base	https://github.com/haya-finance/haya-smart-contracts
Commit Id	f6e8561e6b7f11800dc648f2de9a4d4dc52767c4 afa47cba8927c7c2714765ed394a757cf644bcc9 9d85ce83b1a0357e55a8b7544d8512317225fc88

1.2 Audit Overview

Audit work duration: Apr 18, 2024 – May 1, 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
haya finance-01	Negative parameters may cause logic errors	Low	Fixed
haya finance-02	The logic to update MaxTick is missing in _cancelBid function	Low	Fixed
haya finance-03	Missing event	Info	Fixed

Finding Details:

[haya finance-01] Negative parameters may cause logic errors

Severity Level	Low
Type	Business Security
Lines	contracts/protocol/modules/AuctionRebalanceModule.sol
Description	In the `AuctionRebalanceModule` contract, `_virtualAmount` represents the user's bidding amount, which should not be negative. However, since the contract only checks `_virtualAmount >= minBidVirtualAmount`, and `minBidVirtualAmount` can be negative, `_virtualAmount` may be negative. In this case, there are problems with many logics. For example: when the target price is positive, the normal logic should indicate that the bidder pays Set tokens to increase the winning rate. However, since `_virtualAmount` is negative, the calculated `setsTokenAmountNeeded` is negative, there is no need to send Set tokens; similarly, the processing logic of components will be reversed. `require( _virtualAmount >= info.minBidVirtualAmount && _virtualAmount % info.minBidVirtualAmount == 0, "Virtual quantity not meeting the requirements" );`
Recommendation	It is recommended to add a check on `minBidVirtualAmount`, requiring it to be greater than 0 to avoid the above problems caused by incorrect management settings.
Status	**Fixed.** This issue has been fixed in the commit of ab7ebce4fdf7ed6db878fc5724b5033ea21285cd, which now requires `minBidVirtualAmount` to be greater than 0. Then `_virtualAmount` will always be positive. `require(_minBidVirtualAmount > 0, "Min virtual amount must be bigger than 0");`

[haya finance-02] The logic to update MaxTick is missing in _cancelBid function

Severity Level	Low
Type	Business Security
Lines	contracts/protocol/modules/AuctionRebalanceModule.sol #L670-698
Description	In the `_cancelBid` function of the `AuctionRebalanceModule` contract, if all auctions for the specified tick are deleted, only the initialization status of the tick is updated, but the maximum tick of the `serialId` is not updated. `if (afterRollback == 0) { mapping(int16 => uint256) storage tickBitmap = tickBitmaps[ _setToken ][serialId]; tickBitmap.flipTick(_tick, 1); }`
Recommendation	Although this issue does not cause security issues and the tick is skipped in the `nextInitializedTickWithinOneWord` function, it is still recommended to add logic to update the maximum tick as it seems more logical.
Status	**Fixed.** The logic of updating the maximum tick has been added to the new version of the code. `if (afterRollback == 0) { mapping(int16 => uint256) storage tickBitmap = tickBitmaps[ _setToken ][serialId]; tickBitmap.flipTick(_tick, 1); }`

[haya finance-03] Missing event

Severity Level	Info
Type	Coding Conventions
Lines	contracts/protocol/SetToken.sol #L364-378
Description	The lock and unlock functions of the `SetToken` contract modify the contract's locking status without triggering the corresponding events. This could inconvenience users seeking to obtain the contract's locking status.

```
    function lock() external onlyModule {
        require(!isLocked, "Must not be locked");
        locker = msg.sender;
        isLocked = true;
    }
    function unlock() external onlyModule {
        require(isLocked, "Must be locked");
        require(locker == msg.sender, "Must be locker");
        delete locker;
        isLocked = false;
    }
```

Recommendation	It is recommended that these two functions trigger corresponding events.
Status	**Fixed.** Corresponding function events have been added in the new version of the code.

```
    function lock() external onlyModule {
        require(!isLocked, "Must not be locked");
        locker = msg.sender;
        isLocked = true;
        emit Lock(msg.sender, true);
    }
    function unlock() external onlyModule {
        require(isLocked, "Must be locked");
        require(locker == msg.sender, "Must be locker");
        delete locker;
        isLocked = false;
        emit Lock(msg.sender, false);
    }
```

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 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
Blockchain Security

Official Website
https://www.beosin.com

Telegram
https://t.me/beosin

Twitter
https://twitter.com/Beosin_com

Email
service@beosin.com