BEOSIN
Web3 Security & Compliance

STAY token by Staynex

Smart Contract Security Audit

No. 202604211030

Apr 21st, 2026

SECURING BLOCKCHAIN ECOSYSTEM

WWW.BEOSIN.COM

Contents

Summary of Audit Results

After auditing, 2 Low risks were identified in the STAY token by Staynex 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 : 2

- **Project Description:**

This is a token vesting contract called STAY token by StaynexVesting, designed to manage the token unlock schedules for various groups (community members, KOLs, community rewards, staking ecosystem, team and advisors, etc.) within the $STAY token ecosystem. One instance is deployed per distribution pool, and each instance has its own TGE (Token Generation Event) timestamp, cliff period, linear vesting duration, and upfront TGE release percentage — beneficiaries receive a portion of their tokens at TGE, while the remainder unlocks linearly over time after the cliff period. The contract uses a dual-role permission design: Owner (the treasury multisig wallet) handles critical configuration (setting the TGE timestamp, replacing the token address, pausing the contract, blacklisting wallets, emergency withdrawals, reclaiming unclaimed or surplus tokens, etc.), while Operator (the operational wallet) handles day-to-day tasks (adding, batch-adding, updating, and removing beneficiaries, funding the contract, and claiming on behalf of users). `Ownable2Step` requires ownership transfers to be explicitly accepted by the new owner, and `DEFAULT_ADMIN_ROLE` transfers atomically alongside ownership to prevent lingering privileges on old deployer keys. On the security side, it integrates `ReentrancyGuard` (prevents reentrancy), `Pausable` (emergency claim halt), and SafeERC20 (safe token transfers), and supports risk-control mechanisms such as revoking vesting (stops future unlocks while preserving already-vested portions) and blacklisting (blocks all claims, for legal or fraud response). It also exposes read-only functions like `claimable`, `fundingGap`, and `poolConfig` for use by frontends and block explorers.

1 Overview

1.1 Project Overview

Project Name	STAY token by Staynex
Project Language	Solidity
Platform	BNB Chain
GitHub	https://github.com/staynex-official/staynex-vesting/tree/main
Commit	c461df748dab76a4d43523ba3b72256abe307d46 7bf79a918d9e6e0e79001b645337573fc5ee65b1 b6f0cfae811f0ce7c3a3aa542e5362ff80171085

1.2 Audit Overview

Audit work duration: Apr 18 - Apr 21, 2026

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
STAY token by Staynex-01	The FundingGap calculation error	Low	**Fixed**
STAY token by Staynex-02	The withdrawSurplus calculation error	Low	**Fixed**

Finding Details:

[STAY token by Staynex-01] The FundingGap calculation error

Severity Level	Low
Type	Business Security
Lines	TokenStaking.sol#L619-622

Description	The variable `totalAllocatedTokens` does not decrease after the claim is processed, and the token balance in the contract has already paid for the amount of tokens claimed. Therefore, `totalAllocatedTokens` will be larger than the actual calculated value, causing the contract to require more tokens for computation.
	`function fundingGap() external view returns (uint256 gap) {`
	`uint256 bal = stayToken.balanceOf(address(this));`
	`if (bal >= totalAllocatedTokens) return 0;`
	`return totalAllocatedTokens - bal;`
	`}`

Recommendation	It is recommended to deduct the amount already claimed when calculating.

Status	**Fixed.** Specific changes made by the project team: https://github.com/staynex-official/staynex-vesting/commit/7bf79a918d9e6e0e79001b645337573fc5ee65b1.
	`function fundingGap() external view returns (uint256 gap) {`
	`// Outstanding obligation = total promised minus total already paid out.`
	`uint256 outstanding = totalAllocatedTokens - totalClaimedTokens;`
	`uint256 bal = stayToken.balanceOf(address(this));`
	`if (bal >= outstanding) return 0;`
	`return outstanding - bal;`
	`}`

[STAY token by Staynex-02] The withdrawSurplus calculation error

Severity Level	Low
Type	Business Security
Lines	StaynexVesting.sol#L248-254
Description	The `withdrawSurplus` function does not consider the portion already claimed by the user, which leads to inaccurate calculations.

```
    function withdrawSurplus() external onlyOwner {
        uint256 bal    = stayToken.balanceOf(address(this));
        uint256 needed = totalAllocatedTokens;
        require(bal > needed, "No surplus to withdraw");
        uint256 surplus = bal - needed;
        stayToken.safeTransfer(owner(), surplus);
        emit SurplusWithdrawn(surplus);
    }
```

Recommendation	It is recommended to deduct the amount already claimed when calculating.
Status	**Fixed.** Specific changes made by the project team: https://github.com/staynex-official/staynex-vesting/commit/b6f0cfae811f0ce7c3a3aa542e5362ff80171085.

```
    function withdrawSurplus() external onlyOwner {
        // Outstanding obligation = total promised minus total already paid out.
        // Mirrors {fundingGap} so surplus accurately reflects the unpaid amount.
        uint256 bal         = stayToken.balanceOf(address(this));
        uint256 outstanding = totalAllocatedTokens - totalClaimedTokens;
        require(bal > outstanding, "No surplus to withdraw");
        uint256 surplus = bal - outstanding;
        stayToken.safeTransfer(owner(), surplus);
        emit SurplusWithdrawn(surplus);
    }
```

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	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