BEOSIN
Web3 Security & Compliance

Turboflow

Smart Contract Security Audit

No. 202510271213

Oct 27th, 2025

SECURING BLOCKCHAIN ECOSYSTEM

WWW.BEOSIN.COM

Contents

Summary of Audit Results

After auditing, 2 Medium risks and 2 Info items were identified in the Turboflow 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 : 2
Info	Fixed : 0 Acknowledged : 2

- **Project Description:**

The EVM part of the Turboflow, which primarily handles cross-chain asset transfer and management. The CommonBridge contract serves as the core module, designed to facilitate asset transfers and review processes across multiple chains. Users can initiate cross-chain deposits through the `deposit` function, locking specified tokens or native assets within the contract and generating corresponding withdrawal proposals (`proposal`). The contract employs a multi-role access control system, including `SUPER_ADMIN`, `NORMAL_ADMIN`, `proposeAdminAddress`, and `reviewAdminAddress`, which are responsible for system configuration, permission management, proposal creation, and proposal review, respectively. It supports management of target chain `domainID` and whitelisted bridge tokens, configuration of bridge fees and dynamic rates, and mechanisms to prevent duplicate proposal submissions. Additionally, it provides `pause`, `retry`, and emergency withdrawal functions to ensure the security and controllability of cross-chain fund operations.

The SVM part of Turboflow, which handles cross-chain asset transfers on Solana. The program employs a multi-role access control system, including roles such as `DEFAULT_ADMIN_ROLE`, proposer, and reviewer, which are responsible for system configuration, permission management, proposal creation, and proposal review, respectively. Users can initiate cross-chain deposits through the `deposit_native` and `deposit_token` functions, locking specified native assets or whitelisted SPL tokens. Subsequently, the proposer will submit a withdrawal proposal, and the generated review_account will record information such as `local_domain_id`, `remote_domain_id`, `destination_recipient_address`, etc. Finally, after review, the reviewer will call the review_token and review_native functions to transfer funds to the specified recipient_token_account. Among them, the cross-chain assets involved include two types, and the corresponding cross-chain logic is implemented separately using conditional compilation `#[cfg(not(feature = "enable-mint"))]` and `#[cfg(feature = "enable-mint")]`. The former represents cross-chain assets that are svm_token_mint generated by this contract, while the latter represents cross-chain assets that are externally introduced custody_token_mint.

1 Overview

1.1 Project Overview

Project Name	Turboflow
Project language	Solidity and Rust
Platform	Binance Smart Chain and SVM Multi-Chains
Github Link	https://github.com/Lumoz-protocol/evm-bridge-contract https://github.com/Lumoz-protocol/svm-bridge-contract/tree/main
Commit	EVM: 29ca0ec345f84f5973d4fa7b650fa13ffb057e40 1c0c9df60e6108505b2441f024fcdccffe7522c5 SVM: 3f78c0389dae0154f7f1aab1df8c99e5f3cb5aeb f61d6ce97f584aad97445adb7b9a380b5685f1dd

1.2 Audit Overview

Audit work duration:Oct 9 - Oct 27, 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
Turboflow-EVM-01	Stale Transaction Hash Causes Proposal Lock	Medium	Fixed
Turboflow-EVM-02	Incomplete parameter validation	Info	Acknowledged
Turboflow-EVM-03	Redundant codes	Info	Acknowledged
Turboflow-SVM-01	Missing Permission Checks	Medium	Fixed

Finding Details:

[Turboflow-EVM-01] Stale Transaction Hash Causes Proposal Lock

Severity Level	Medium
Type	Business Security
Lines	CommonBridge.sol#L353-415
Description	The main purpose of the propose function is to allow authorized administrators to initiate cross-chain proposals, recording information such as the proposal's source domain ID (`originDomainID`), source deposit transaction (`originDepositNonce`), transaction hash (`txHash`), token address, recipient address, and amount, which are used to trigger subsequent cross-chain operations on the destination chain. When an existing proposal is marked as Rejected, the function permits administrators to resubmit a new proposal with the same `originDepositNonce`. However, in this resubmission process, although the proposal content and `txHash` are updated, the old `proposeTxHashHistory` record is not cleared. As a result, the previous transaction hash remains permanently marked as true, causing the system to incorrectly determine that the hash has already been used. This prevents the same transaction hash from being resubmitted, leading to certain cross-chain proposals becoming locked or unable to be properly processed.
Recommendation	It is recommended to change the recipient address from sender to the originally set to address.
Status	**Fixed.** The project team deleted the `proposeTxHashHistory` record corresponding to oldTxHash.

[Turboflow-EVM-02] Incomplete parameter validation

Severity Level	Info
Type	Business Security
Lines	CommonBridge.sol#L353-415,L417-455
Description	The `propose` and `review` functions lack checks for the legitimacy of cross-chain tokens (token). Currently, administrators can specify any token address when submitting or executing a proposal, without verifying whether the token is included in the allowed bridge token list (`bridgeTokenEnable`) or checking the validity or compliance of the address. If an administrator mistakenly or maliciously specifies a non-whitelisted token address, it may result in incorrect cross-chain asset transfers, funds being sent to unsupported token contracts.
Recommendation	It is recommended to add legitimacy and whitelist checks for cross-chain tokens in the propose and review functions, ensuring that only supported tokens can participate in cross-chain operations, thereby enhancing fund security and system robustness.
Status	**Acknowledged.** The project team stated that this verification will be handled by the backend.

[Turboflow-EVM-03] Redundant codes

Severity Level	Info
Type	Coding Conventions
Lines	CommonBridge.sol#L87-88
Description	The contract defines the events `EmergencyWithdrawETH` and `EmergencyWithdrawToken`, but the corresponding function logic is not implemented, making them redundant code. Such unused events are still compiled into the bytecode during deployment, consuming extra storage and increasing the contract's deployment gas cost. Additionally, unused events can complicate code maintenance and create ambiguities during audits.
Recommendation	It is recommended that the project team clarify the business logic and remove the related redundant code.
Status	**Acknowledged.** The project team has acknowledged the issue.

[Turboflow-SVM-01] Missing Permission Checks

Severity Level	Medium
Lines	update_token_metadata.rs#L55-95
Type	Business Security
Description	When conditional compilation #[cfg(not(feature = "enable-mint"))] is enabled, the `update_token_metadata` function allows updating the metadata account of SVM_TOKEN. However, this interface lacks permission checks, and the update_authority of the metadata is the program's PDA account mint_authority, enabling anyone to call this interface and modify the metadata account data. Similarly, the create_token_metadata function also lacks permission validation, allowing anyone to create metadata accounts. As a result, arbitrarily set metadata accounts could mislead users and disrupt off-chain data tracking.

```
pub fn update_token_metadata(
    ctx: Context<UpdateTokenMetadata>,
    params: &UpdateTokenMetadataParmas,
) -> Result<()> {
    let cpi_accounts = UpdateMetadataAccountsV2 {
        metadata: ctx.accounts.metadata.clone(),
        update_authority: ctx.accounts.mint_authority.clone(),
    };
    let authority_seeds: &[&[&[u8]]] = &[&[
        TRANSFER_AUTHORITY.as_bytes(),
        &[ctx.accounts.bridge_account.transfer_authority_bump],
    ]];
    let cpi_ctx =
CpiContext::new(ctx.accounts.token_metadata_program.clone(),
cpi_accounts)
        .with_signer(authority_seeds);
    let creators = vec![Creator {
        address: ctx.accounts.mint_authority.key(),
        verified: true,
        share: 100,
    }];
    let name = params.name.clone();
    let symbol = params.symbol.clone();
    let uri = params.uri.clone();
```

```
    let seller_fee_basis_points = params.seller_fee_basis_points;
    let data = DataV2 {
        name,
        symbol,
        uri,
        seller_fee_basis_points,
        creators: Some(creators),
        collection: None,
        uses: None,
    };
    update_metadata_accounts_v2(cpi_ctx, None, Some(data), None,
Some(false))?;
    Ok(())
}
```

Recommendation	It is recommended that the project team add permission checks to the `update_token_metadata` and `create_token_metadata` interfaces, restricting calls to whitelisted addresses only.
Status	**Fixed.**The project only allows accounts with TOKEN_ADMIN_ROLE to call this function.

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 Critical 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
EVM Contract	Coding Conventions	Deprecated Items
		Redundant Code
		require/assert Usage
		Default Values
	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
	Business Security	Business Logics
		Business Implementations
		Manipulable Token Price
		Centralized Asset Control
		Arbitrage Attack
		Access Control
SVM Contract	Coding Conventions	SPL Token Standards
		Visibility Specifiers
		Lamport Check
		Account Check
		Signer Check
		Program Id Check
		Deprecated Items

	General Vulnerability	Integer Overflow/Underflow
		Reentrancy
		Pseudo-random Number Generator (PRNG)
		Transaction-Ordering Dependence
		DoS (Denial of Service)
		Function Call Permissions
		Returned Value Security
		Replay Attack
		Overriding Variables
		Third-party Protocol Interface Consistency
	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 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/