BEOSIN
Web3 Security & Compliance

Bifrost (veth-3.0 && slpx-v2)

Smart Contract Security Audit

No. 202509251035

Sep 25th, 2025

SECURING BLOCKCHAIN ECOSYSTEM

WWW.BEOSIN.COM

Contents

Summary of Audit Results

After auditing, 1 low-risk and 2 Info items were identified in the Bifrost (veth-3.0 && slpx-v2). 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:

Business overview

The Solidity audit in this review covers the following contracts under the vETH-3.0 module: HyperbridgeHandler, LayerZeroHandler, SnowbridgeHandler, SlpForward, SlpProxy, SlpSsvManager, SlpVault, SlpWithdrawalVault, and WhitelistBytes32.
For slpx v2, the audit includes the following contracts: BridgeVault, Oracle, VTokenBase, VToken, and VETH. All other contracts are outside the scope of this audit.

vETH-3.0 Module Overview

Bridges Contracts： All contracts under the bridges folder use a whitelist for access control. Their primary role is to facilitate cross-chain transfers across three different bridging protocols.
SlpForward： This contract distributes assets to different bridges. It supports cross-chain transfers of either the stored WETH within the contract or the amount sent by the user in the current transaction.
SlpProxy： Acts as a central management contract with the following responsibilities,Handling staking (`bond/unbond`) with the ETH → ETH2.0 DepositContract,Registering and removing validators on the SSV Network. Interacting with cross-chain bridges (Snowbridge, Hyperbridge, LayerZero, etc.) to return ETH to Bifrost or L2. Managing fund vaults, which are categorized into bond, bridge, and withdrawal vaults.
SlpSsvManager: Provides the interface for registering and removing validators in the `_ssvNetwork`. It also handles staking and withdrawal operations, and includes liquidation functionality for clusters in order to retrieve `_ssvToken`.
SlpVault: Implements access-controlled storage and retrieval of the platform token.
ValidatorManager: A batch management tool for Ethereum validators. It is designed around EIP-7002 and EIP-7251, providing functionality for batch consolidation, switching, and exiting of validators. The contract also includes built-in permission control and error handling mechanisms.
WhitelistBytes32: Implements whitelist management logic to enforce access restrictions.

slpx v2 Module Overview

VToken Suite: This module consists of three contracts in sequential inheritance: VTokenBase, VToken, and VETH. VTokenBase implements the core logic, modifying the ERC4626 standard for asset-to-share conversion. The conversion ratio is provided by the Oracle rather than reflecting the actual on-chain balance. Each conversion is subject to a fee.

Functions `asyncMint` and `asyncRedeem` send the updated token amounts from the current cycle across chains to Bifrost. Functions `withdrawComplete` and `withdrawCompleteTo` allow users to claim withdrawals based on queued and pending amounts, determining the redeemable and requested quantities.

Oracle: The contract owner can adjust the conversion fee ratio.The contract supports a pause mechanism: when paused, conversions are disabled.Price updates are applied only after the contract receives an `onAccept` callback.

BridgeVault: Serves as storage for assets exchanged via VToken. Assets are withdrawn from the vault during redemption operations.

1 Overview

1.1 Project Overview

Project Name	Bifrost (veth-3.0 && slpx-v2)
Project Language	Solidity
Platform	Ethereum, Base
Code Base	https://github.com/bifrost-io/vETH-3.0-contract/tree/audit-1.0.0 https://github.com/bifrost-io/slpx-contracts-v2/releases/tag/audit-1.0.0
Commit Hash	vETH-3.0: 4cfd6de9fe2a02730284c5f404ef5648654c29ae a61c2cbc027cbdd695f274bd7739bfc10f6ce40d 45f215b4936d95f43ec2715b2983ed6595db8c93 Slpx v2: 3f88e1d77c45da5953b032e2a14cf314d3b3733f 9a546046806e3b15526401a7e93013e32d303cfe

1.2 Audit Overview

Audit work duration: Sep 5, 2025 – Sep 19, 2025

Update time: Sep 25, 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 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
Bifrost-01	The asyncMint function sets the redeem parameter to true	Low	**Fixed**
Bifrost-02	Compilation issues with the test script	Info	**Fixed**
Bifrost-03	Missing function in the called contract	Info	**Fixed**

Finding Details:

[Bifrost-01] The asyncMint Function Sets the redeem Parameter to true

Severity Level	Low
Lines	ValidatorManager.sol #L 82
Type	Business Security

Description	The `batchConsolidation` function validates whether the provided msg.value meets the required cost; however, any excess amount is not refunded and remains locked in the contract. This results in users' overpaid funds being stranded and irretrievable, leading to potential value loss.
	`function batchConsolidation(bytes[] calldata sourcePubkeys, bytes calldata targetPubkey)`
	`external`
	`payable`
	`onlyAuthority(msg.sender)`
	`{`
	`uint256 batchSize = sourcePubkeys.length;`

Recommendation	It is recommended to enforce that the amount of `msg.value` provided in the `batchConsolidation` function must exactly match the required fee, thereby preventing excess funds. Alternatively, consider implementing a withdrawal function that allows the admin or users to reclaim any excess funds, ensuring they are not permanently locked in the contract.

Status	**Fixed.**
	`uint256 exitFee = getExitFee();`
	`require(msg.value == batchSize * exitFee, InsufficientFeePerValidator());`

[Bifrost-02] Compilation issues with the test script

Severity Level	Info
Lines	foundry.toml #L 10
Type	Business Security

Description	In the remappings section of foundry.toml, if the mapping for @polytope-labs/ismp-solidity does not end with a /, it will cause compilation errors.
	`remappings = [`
	`"@openzeppelin/contracts/=lib/openzeppelin-contracts/contracts",`
	`"@openzeppelin/contracts-upgradeable/=lib/openzeppelin-contracts-upgradeable/contracts/",`
	`"@polytope-labs/ismp-solidity/=node_modules/@polytope-labs/ismp-solidity",`

Recommendation	It is recommended to append a / to the @polytope-labs/ismp-solidity mapping to ensure successful compilation.

Status	**Fixed.**
	`"@polytope-labs/ismp-solidity/=node_modules/@polytope-labs/ismp-solidity/",`
	`"@polytope-labs/solidity-merkle-trees/=node_modules/@polytope-labs/solidity-merkle-trees/"`
	`]`

[Bifrost-03] Missing Function in the Called Contract

Severity Level	Info
Lines	LayerZeroHandler.t.sol #L 141
Type	Business Security
Description	In the tests for the LayerZeroHandler contract, `vm.mockCall` was used on the stargateRouter to simulate the `swapETH` function. However, the actual deployed address (0x2836045A50744FB50D3d04a9C8D18aD7B5012102) does not contain an implementation of `swapETH`. Before going live, the project must ensure that the stargateRouter contract includes the proper implementation of swapETH.
Recommendation	It is recommended that the project ensures the stargateRouter contract includes the correct implementation of `swapETH`.
Status	**Fixed.**

```
        vm.mockCall(
            address(stargateRouter),
            abi.encodeWithSignature("swapETH(uint16,address,bytes,uint256,uint256)"),
            abi.encode()
        );

        uint256 initialBalance = SENDER.balance;
        vm.prank(SENDER);
        uint256 result = handler.sendToken{value: AMOUNT + lzFee}(address(0), TO_ADDRESS, AMOUNT, data);
```

```
   address public constant SEPOLIA_STARGATE_ROUTER = 0x676Fa8D37B948236aAcE03A0b34fc0Bc37FABA8D;
```

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 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/