THREE SIGMA

Hyperliquid Forwarder Bridge

Security Review



Disclaimer

Security Review

Hyperliquid Forwarder Bridge



Disclaimer

The ensuing audit offers no assertions or assurances about the code's security. It cannot be deemed an adequate judgment of the contract's correctness on its own. The authors of this audit present it solely as an informational exercise, reporting the thorough research involved in the secure development of the intended contracts, and make no material claims or guarantees regarding the contract's post-deployment operation. The authors of this report disclaim all liability for all kinds of potential consequences of the contract's deployment or use. Due to the possibility of human error occurring during the code's manual review process, we advise the client team to commission several independent audits in addition to a public bug bounty program.

Table of Contents

Security Review

Hyperliquid Forwarder Bridge



Table of Contents

Disclaimer **3**
Summary **7**
Scope **9**
Methodology **9**
Project Dashboard **13**
Risk Section **16**
Findings **18**
3S-Hyperliquid Forwarder-N01 18

Summary

Security Review

Hyperliquid Forwarder Bridge



Summary

Three Sigma audited HyperliquidForwarder in a 0.4 person week engagement. The audit was conducted on 10/06/2025.

Protocol Description

The HyperliquidForwarder contract is designed to facilitate ERC20 token transfers from the HyperEVM network to the HyperCore network using a two-step process. It allows for the bridging of tokens such as USDT0 and USDE through verified intermediaries. The contract includes security measures to prevent unsupported tokens and mismatched bridge addresses.

Scope

Security Review

Hyperliquid Forwarder Bridge



Scope

Filepath	nSLOC
src/HyperliquidForwarder.sol	59
Total	**59**

Methodology

Security Review

Hyperliquid Forwarder Bridge



To begin, we reasoned meticulously about the contract's business logic, checking security-critical features to ensure that there were no gaps in the business logic and/or inconsistencies between the aforementioned logic and the implementation. Second, we thoroughly examined the code for known security flaws and attack vectors. Finally, we discussed the most catastrophic situations with the team and reasoned backwards to ensure they are not reachable in any unintentional form.

Taxonomy

In this audit, we classify findings based on Immunefi's Vulnerability Severity Classification System (v2.3) as a guideline. The final classification considers both the potential impact of an issue, as defined in the referenced system, and its likelihood of being exploited. The following table summarizes the general expected classification according to impact and likelihood; however, each issue will be evaluated on a case-by-case basis and may not strictly follow it.

Impact / Likelihood	LOW	MEDIUM	HIGH
NONE	None		
LOW	Low		
MEDIUM	Low	Medium	Medium
HIGH	Medium	High	High
CRITICAL	High	Critical	Critical

Project Dashboard

Security Review

Hyperliquid Forwarder Bridge



Project Dashboard

Application Summary

Name	HyperliquidForwarder
Repository	https://github.com/SwellNetwork/hyperliquid-forwarder
Commit	56e4679
Language	Solidity
Platform	Hyperliquid

Engagement Summary

Timeline	10/06/2025
№ of Auditors	2
Review Time	0.4 person weeks

Vulnerability Summary

Issue Classification	Found	Addressed	Acknowledged
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	0	0	0
None	1	1	0

Category Breakdown

Suggestion	1
Documentation	0
Bug	0
Optimization	0
Good Code Practices	0

Risk Section

Security Review

Hyperliquid Forwarder Bridge



Risk Section

No risk was identified.

Findings

Security Review

Hyperliquid Forwarder Bridge



Findings

3S-Hyperliquid Forwarder-N01

The function **addTokenIDToBridgeMapping** can be declared as external

Id	3S-Hyperliquid Forwarder-N01
Classification	None
Category	Suggestion
Status	Addressed in #b5ab1a7.

Description

The **HyperliquidForwarder::addTokenIDToBridgeMapping** function is designed to add or update the bridge mapping for a token based on its ID. This function is currently declared with **public** visibility, which allows it to be called both internally and externally. However, the function is intended to be called only externally, as it does not require internal invocation. The use of **public** visibility instead of **external** results in unnecessary gas consumption due to the additional overhead of copying arguments from calldata to memory.

Recommendation

Change the visibility of the **addTokenIDToBridgeMapping** function from **public** to **external** to optimize gas usage.