Hypernova Ethereum Contracts
Smart Contract Assessment Report

Date: July 18, 2025
Version 1.0 [Final]

Contents

Executive Summary

Supra Labs team engaged with *RektProof Security* to perform an assessment of *Hypernova Ethereum Contracts*. This assessment was conducted between *Jun 24th* and *Jul 14th, 2025*.

Assessment overview

This report outlines the results of the security audit conducted on the Hypernova Ethereum Contracts.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource, and expertise-bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs, and on-chain monitoring are strongly recommended.

Contact Information

Name	Title	Contact Information
Supra Labs		
Arun Doraiswamy	VP of Engineering (Blockchain)	a.doraiswamy@supra.com
RektProof Security		
Naveen Kumar J	Smart Contract Auditor	naina.navs@gmail.com
Kalai Mohan	Smart Contract Auditor	kalaimohan328@gmail.com
Raj Kumar	Smart Contract Auditor	0xrajkumar1337@gmail.com

Finding Severity Classification

The following table defines a severity matrix that determines the severity of the bug by evaluating two main factors: the impact of the issue and its likelihood. This approach ensures an accurate assignment of severity.

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Scope

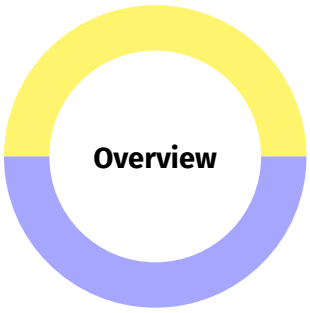
The source code was delivered to us in a git repository at `https://github.com/Entropy-Foundation/supra-interoperability-solutions`. This audit was performed against commit `f37d1bae4b6abac6ea044780cd8193e3b6766d08`. Reported issues were fixed in the commit `c2db8e15c771f4bcf263c50e0890cb6c84369f78`.

A brief description of the programs is as follows:

Name	Description
Hypernova	Ethereum Bridge Component.

Findings

Severity	Count
CRITICAL	0
HIGH	0
MEDIUM	1
LOW	0
INFO	1



0.1 Moderate Risk

0.1.1 TokenVault Locked Amount Data Type Issue

Severity: Moderate

Description:

The *lockedTokens* mapping uses *uint64* to store the total locked amount for each address. While this seems large, it may not be sufficient for tokens with high decimal places or high total supply.

Impact:

This causes the contract to fail when users try to lock large amounts of tokens. Once the locked amount approaches the *uint64* limit, users will be unable to lock additional tokens, causing transaction failures.

Recommendation:

Change the data type from *uint64* to *uint256* in the lockedTokens mapping. Also change relevant function parameters involved.

Team Comment: Known Issue

Status:

Resolved

0.2 Informational

0.2.1 TokenVault isValidLimit Function - Boundary Condition Issue

Description: The *isValidLimit* function has a logical error in how it checks valid amounts. It currently uses *(amount > min && amount < max)* which means users cannot use the exact minimum and maximum values. The function should use *(amount >= min && amount <= max)* to allow the boundary values.

```
function isValidLimit(uint64 amount, uint64 min, uint64 max) internal pure returns (bool) {
    return (amount > min && amount < max);
}
```

Impact:

This issue causes user transactions to fail when they try to use the minimum or maximum values. Users will waste gas fees and get confused when their valid amounts are rejected.

Recommendation:

Change the function to use >= and <= instead of > and <:

```
function isValidLimit(uint64 amount, uint64 min, uint64 max) internal pure returns (bool) {
    return (amount >= min && amount <= max);
}
```

Status:

Resolved