CODE SECURITY ASSESSMENT

NATIVE ORG

Overview

Project Summary

- Name: Native org - v3 periphery warp
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/Native-org/v3-periphery/tree/feat/wrap
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Native org - v3 periphery warp
Version	v2
Type	Solidity
Dates	Apr 01 2025
Logs	Apr 01 2025; Apr 01 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	1
Total	2

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):
- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Centralization risk	Low	Centralization	Acknowledged
2	Missing two-step transfer ownership pattern	Informational	Business Logic	Acknowledged

2.2 Notable Findings

Please replace angle brackets in the template.

1. Centralization risk	
Severity: Low	Category: Centralization
Target: - contracts/base/PeripheryMutableStateI.sol	

Description

The `PeripheryMutableState` contract has privileged accounts, the owner can perform privileged operations such as setUnderlying, setAllowance, and others.

If the owner's private key is compromised, an attacker could transfer ownership to another address and arbitrarily modify the `underlying` token associated with `lpToken`, as well as adjust the contract's allowance for the `NLP` contract.

If the privileged accounts are plain EOA accounts, this can be worrisome and pose a risk to the other users.

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

Status

This issue has been acknowledged by the team.

SALUS

2.3 Informational Findings

2. Missing two-step transfer ownership pattern	
Severity: Informational	Category: Business logic
Target: - contracts/base/PeripheryMutableStatel.sol	

Description

The `PeripheryMutableState` contract uses a custom function `setOwner()` which is a simple mechanism to transfer the ownership not supporting a two-step transfer ownership pattern. This simpler mechanism can be useful for quick tests, but projects with production concerns are likely to outgrow it. Transferring ownership is a critical operation and this could lead to transferring it to an inaccessible wallet or renouncing the ownership, e.g. mistakenly.

Recommendation

It is recommended to implement a two-step transfer of ownership mechanism where the ownership is transferred and later claimed by a new owner to confirm the whole process and prevent lockout.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covers the files in [Native-org/v3-periphery](#) The repository is built on Uniswap V3, and this audit specific ally examines the modifications made in this repository to Uniswap V3 Periphery at [0682387](#) commit.

SALUS