Security Review Report
NM-0438 Overlay Shiva

NETHERMIND
SECURITY

(Feb 13, 2025)

Contents

1 Executive Summary

This document presents the security review performed by Nethermind Security for Overlay Shiva smart contracts. Overlay Protocol is a decentralized platform allowing users to build positions on a market or data stream without traditional counterparties (liquidity providers or market makers) taking the other side. The protocol can offer markets on ANY non-manipulable non-predictable numerical data feed.

Shiva is built on top of Overlay to manage positions across the multiple Overlay markets. Users can create and manage their positions in any Overlay market through Shiva, by doing this they get access to rewards from the Proof of Liquidity mechanism built in Berachain.

The audit comprises 701 lines of solidity code from the Shiva contracts and a Pull Request in the Overlay core contracts adding a callback to Shiva once a position gets liquidated. During the review, changes to nonce management were applied as a result of the Overlay and Nethermind teams' communication; the new implementation was also checked as part of the engagement.

The audit was performed using (a) manual analysis of the codebase, (b) automated analysis tools, and (c) creation of test cases. **Along this document, we report** two points of attention, where both of them are classified as `Informational`. The issues are summarized in Fig. 1.

This document is organized as follows. Section 2 presents the files in the scope. Section 3 summarizes the issues. Section 4 presents the system overview. Section 5 discusses the risk rating methodology. Section 6 details the issues. Section 7 discusses the documentation provided by the client for this audit. Section 8 presents the compilation, tests, and automated tests. Section 9 concludes the document.

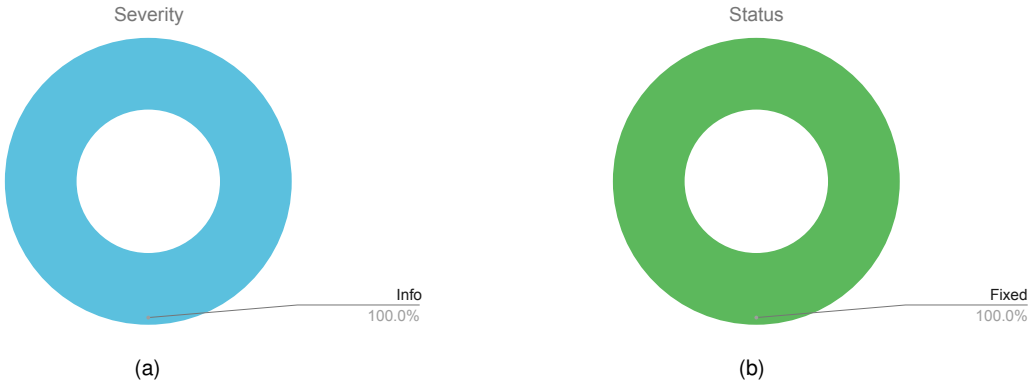


Fig. 1: Distribution of issues: Critical (0), **High** (0), **Medium** (0), **Low** (0), **Undetermined** (0), **Informational** (2), **Best Practices** (0). **Distribution of status: Fixed** (2), **Acknowledged** (0), **Mitigated** (0), **Unresolved** (0)

Summary of the Audit

Audit Type	Security Review
Initial Report	February 7, 2025
Final Report	February 13, 2025
Repository (Shiva)	v1-shiva
Commit	9abeb89927c29086da663ae0d93c385733015d75
Repository (Core)	v1-core
Pull Request	PR 207
Final Commit	3e10c6099e263170806c80a9ae9940e2d1b4f472
Documentation	README
Documentation Assessment	High
Test Suite Assessment	High

2 Audited Files

	Contract	LoC	Comments	Ratio	Blank	Total
1	src/ShivaStructs.sol	33	38	115.2%	5	76
2	src/PolStakingToken.sol	12	5	41.7%	4	21
3	src/IShiva.sol	57	106	186.0%	21	184
4	src/Shiva.sol	410	254	62.0%	90	754
5	src/utils/Utils.sol	82	43	52.4%	11	136
6	src/interfaces/aggregator/IFluxAggregator.sol	95	8	8.4%	2	105
7	src/interfaces/berachain/IRewardVaults.sol	12	8	66.7%	7	27
	Total	**701**	**462**	**65.9%**	**140**	**1303**

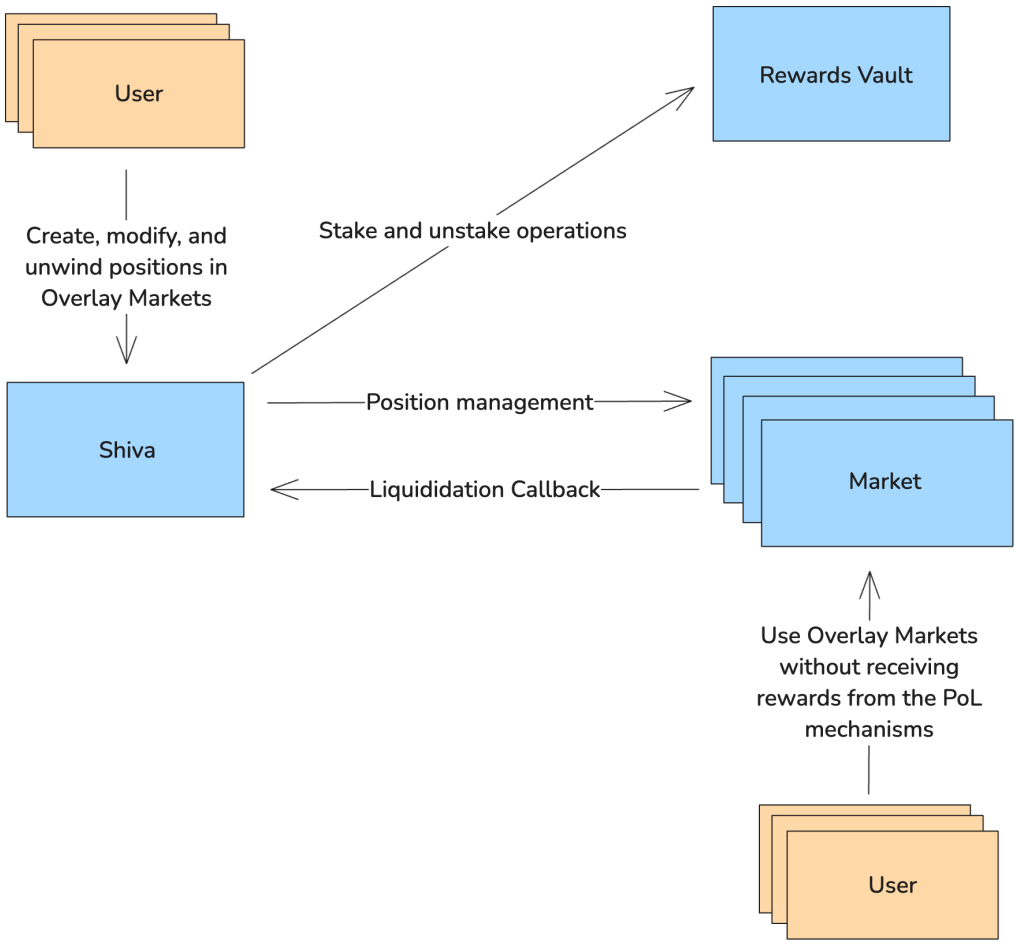
***The scope also includes the PR 207**

3 Summary of Issues

	Finding	Severity	Update
1	Donating `ovl` tokens can cause `buildSingle` operations to fail	Info	Fixed
2	`validMarkets(...)` function may return incorrect results for uncached markets	Info	Fixed

4 System Overview

The **Shiva** contract is designed to interact with the **Overlay** protocol, providing a streamlined mechanism for building, unwinding, and managing positions in **Overlay** markets. Additionally, it integrates staking functionality via the **BerachainRewardsVault**, allowing users to stake their collateral and earn rewards.



4.1 Main interactions

Users interact with **Shiva** to manage positions in **Overlay** markets through three primary actions:

- **build**: Opens a new position in an **Overlay** market. Users participating through **Shiva** can also earn rewards from the Proof of Liquidity (PoL) mechanism.
- **unwind**: Allows users to partially or fully close a position. The user's PoL rewards adjust based on their new participation level.
- **buildSingle**: Bundles an unwind and build action in the same market, effectively modifying an existing position. The user's remaining collateral from the unwind is used to open a new position in addition to any extra collateral provided.

These actions can also be executed on behalf of another user via a valid signed message, following the EIP-712.

4.2 Integration with Proof of Liquidity

Interacting with **Overlay** through **Shiva** enables users to participate in Berachain's PoL rewards system. When **Shiva** is deployed, it creates a dedicated **RewardVault** to facilitate staking for PoL rewards.

- Every position opened through **Shiva** generates a receipt token, which is automatically staked in the **RewardVault** on behalf of the user.
- Users receive PoL rewards based on their active participation in **Overlay**.
- When a position is unwound or liquidated, **Shiva** unstakes and burns the corresponding receipt tokens.

4.3 Emergency Withdraw

If an **Overlay** market is shut down for any reason, users can retrieve their positions using the `emergencyWithdraw(...)` function. Importantly:

- Even if a market is shut down, users continue earning PoL rewards until their positions are fully exited.
- Any user can execute an emergency withdrawal for another user without restriction. This could incentivize users to process withdrawals on behalf of others, increasing their own share of PoL rewards.

4.4 Markets validation

Shiva only interacts with valid markets. A market is considered valid if it has been approved by any of the authorized factories.

- The contract owner can add or remove authorized factories at any time.
- While authorized factories can be removed, any market validated at any point remains permanently valid.

5 Risk Rating Methodology

The risk rating methodology used by Nethermind Security follows the principles established by the OWASP Foundation. The severity of each finding is determined by two factors: **Likelihood** and **Impact**.

Likelihood measures how likely the finding is to be uncovered and exploited by an attacker. This factor will be one of the following values:

a) **High**: The issue is trivial to exploit and has no specific conditions that need to be met;

b) **Medium**: The issue is moderately complex and may have some conditions that need to be met;

c) **Low**: The issue is very complex and requires very specific conditions to be met.

When defining the likelihood of a finding, other factors are also considered. These can include but are not limited to motive, opportunity, exploit accessibility, ease of discovery, and ease of exploit.

Impact is a measure of the damage that may be caused if an attacker exploits the finding. This factor will be one of the following values:

a) **High**: The issue can cause significant damage, such as loss of funds or the protocol entering an unrecoverable state;

b) **Medium**: The issue can cause moderate damage, such as impacts that only affect a small group of users or only a particular part of the protocol;

c) **Low**: The issue can cause little to no damage, such as bugs that are easily recoverable or cause unexpected interactions that cause minor inconveniences.

When defining the impact of a finding, other factors are also considered. These can include but are not limited to Data/state integrity, loss of availability, financial loss, and reputation damage. After defining the likelihood and impact of an issue, the severity can be determined according to the table below.

		Severity Risk		
Impact	**High**	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Info/Best Practices	Low	Medium
	Undetermined	Undetermined	Undetermined	Undetermined
		Low	**Medium**	**High**
		Likelihood		

To address issues that do not fit a High/Medium/Low severity, Nethermind Security also uses three more finding severities: **Informational**, **Best Practices**, and **Undetermined**.

a) **Informational** findings do not pose any risk to the application, but they carry some information that the audit team intends to pass to the client formally;

b) **Best Practice** findings are used when some piece of code does not conform with smart contract development best practices;

c) **Undetermined** findings are used when we cannot predict the impact or likelihood of the issue.

6 Issues

6.1 [Info] Donating `ovl` tokens can cause `buildSingle` operations to fail

File(s): src/Shiva.sol

Description: The `buildSingle` operations allow users to `unwind` a position and `build` a new one in the same market using the withdrawn funds as part of the collateral for the new position.

```
function _buildSingleLogic(...) internal returns (...) {
    require(_params.leverage >= ONE, "Shiva:lev<min");

    // @audit - Unwind position
    _onUnwindPosition(...);

    // @audit - Compute collateral for the new positions using all the `ovl` tokens in the contract
    uint256 totalCollateral = _params.collateral + ovlToken.balanceOf(address(this));
    uint256 tradingFee = _getTradingFee(_params.ovlMarket, totalCollateral, _params.leverage);

    bool isLong =
        Utils.getPositionSide(_params.ovlMarket, _params.previousPositionId, address(this));

    // @audit - Collect remaining tokens from user
    // transfer from OVL from user to this contract
    ovlToken.transferFrom(_owner, address(this), _params.collateral + tradingFee);

    // Approve the ovlMarket contract to spend OVL
    _approveMarket(_params.ovlMarket);

    // @audit - Build new position
    positionId = _onBuildPosition(...);
}
```

After unwinding the position, all `ovl` tokens held by the contract are considered as collateral for the new position. The trading fee is calculated based on the total collateral, which includes:

- The `ovl` tokens already present in the contract, and;
- The additional `ovl` tokens specified by the user;

Users are expected to compute the total collateral of the position before executing this call so they can set the correct allowance for the function.

However, an attacker could exploit this by donating small amounts of `ovl` tokens to the contract. Since the trading fee is based on the total collateral (which includes donated tokens), this could increase the required fee beyond the user's allowance. As a result, the transaction may fail due to insufficient allowance.

This failed transactions could result in a bad user experience.

Recommendation(s): To prevent unintended fee manipulation, consider tracking the `ovl` balance before and after calling `_onUnwindPosition(...)` and using only the actual unwound amount when computing collateral.

Status: Fixed.

Update from the client: Fixed in commit 85fe9f97.

6.2 [Info] `validMarkets(...)` function may return incorrect results for uncached markets

File(s): src/Shiva.sol

Description: The `validMarkets` mapping is used by the `_checkIsValidMarket(...)` function to cache valid markets.

```
/// @notice Mapping to check if an address is a valid market
mapping(address => bool) public validMarkets;

...

function _checkIsValidMarket(address _market) internal returns (bool) {
    if (validMarkets[_market]) {
        return true;
    }

    for (uint256 i = 0; i < authorizedFactories.length; i++) {
        if (authorizedFactories[i].isMarket(_market)) {
            validMarkets[_market] = true;

            emit MarketValidated(_market);
            return true;
        }
    }

    return false;
}
```

The `validMarkets` mapping is public and a getter function for it is created automatically. While this function might appear to serve as a way to check whether a market is valid, it does not perform a full validation. Specifically, it returns false for valid markets that have not yet been cached by `_checkIsValidMarket(...)`.

Recommendation(s): Consider changing the function to be `private` or `internal` if it is not intended to be used externally.

Status: Fixed.

Update from the client: Fixed in commit d6a24f90.

7 Documentation Evaluation

Software documentation refers to the written or visual information that describes the functionality, architecture, design, and implementation of software. It provides a comprehensive overview of the software system and helps users, developers, and stakeholders understand how the software works, how to use it, and how to maintain it. Software documentation can take different forms, such as user manuals, system manuals, technical specifications, requirements documents, design documents, and code comments. Software documentation is critical in software development, enabling effective communication between developers, testers, users, and other stakeholders. It helps to ensure that everyone involved in the development process has a shared understanding of the software system and its functionality. Moreover, software documentation can improve software maintenance by providing a clear and complete understanding of the software system, making it easier for developers to maintain, modify, and update the software over time. Smart contracts can use various types of software documentation. Some of the most common types include:

- Technical whitepaper: A technical whitepaper is a comprehensive document describing the smart contract's design and technical details. It includes information about the purpose of the contract, its architecture, its components, and how they interact with each other;
- User manual: A user manual is a document that provides information about how to use the smart contract. It includes step-by-step instructions on how to perform various tasks and explains the different features and functionalities of the contract;
- Code documentation: Code documentation is a document that provides details about the code of the smart contract. It includes information about the functions, variables, and classes used in the code, as well as explanations of how they work;
- API documentation: API documentation is a document that provides information about the API (Application Programming Interface) of the smart contract. It includes details about the methods, parameters, and responses that can be used to interact with the contract;
- Testing documentation: Testing documentation is a document that provides information about how the smart contract was tested. It includes details about the test cases that were used, the results of the tests, and any issues that were identified during testing;
- Audit documentation: Audit documentation includes reports, notes, and other materials related to the security audit of the smart contract. This type of documentation is critical in ensuring that the smart contract is secure and free from vulnerabilities.

These types of documentation are essential for smart contract development and maintenance. They help ensure that the contract is properly designed, implemented, and tested, and they provide a reference for developers who need to modify or maintain the contract in the future.

Remarks about the Shiva Protocol documentation

The Shiva team provided a document containing the multiple flows available for users. The repository contains a Readme explaining the multiple features implemented in the contract. Besides this information, the contracts contain many comments, and the team was available to answer any questions raised during the engagement.

8 Complementary Checks

8.1 Compilation Output

```
> forge build
[] Compiling...
[] Compiling 10 files with Solc 0.8.10
[] Solc 0.8.10 finished in 8.21s
Compiler run successful with warnings:
Warning (8760): This declaration has the same name as another declaration.
  --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:58:20:
   |
58 |     function _debt(Position.Info memory position) internal view returns (uint256 debt_) {
   |                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
  --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:67:20:
   |
67 |     function _cost(Position.Info memory position) internal view returns (uint256 cost_) {
   |                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
  --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:76:43:
   |
76 |     function _oi(IOverlayV1Market market, Position.Info memory position)
   |                                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
  --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:99:51:
   |
99 |     function _collateral(IOverlayV1Market market, Position.Info memory position)
   |                                                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:149:9:
    |
149 |         uint256 oi = position.oiCurrent(fraction, oiTotalOnSide, oiTotalSharesOnSide);
    |         ^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:353:5:
    |
353 |     function oi(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
```

```
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:133:9:
    |
133 |         Position.Info memory position
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:

   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:190:9:
    |
190 |         uint256 oi = position.oiCurrent(fraction, oiTotalOnSide, oiTotalSharesOnSide);
    |         ^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:353:5:
    |
353 |     function oi(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:174:9:
    |
174 |         Position.Info memory position
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:234:9:
    |
234 |         uint256 oi = position.oiCurrent(fraction, oiTotalOnSide, oiTotalSharesOnSide);
    |         ^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:353:5:
    |
353 |     function oi(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:218:9:
    |
218 |         Position.Info memory position
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:257:9:
    |
257 |         Position.Info memory position
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
```

```
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:298:13:
    |
298 |             uint256 value = _valueForLiquidations(market, data, position);
    |             ^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:380:5:
    |
380 |     function value(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:293:9:
    |
293 |         bool liquidatable = _liquidatable(market, data, position);
    |         ^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:427:5:
    |
427 |     function liquidatable(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:291:9:
    |
291 |         Position.Info memory position
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:304:58:
    |
304 |     function _maintenanceMargin(IOverlayV1Market market, Position.Info memory position)
    |                                                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:334:9:
    |
334 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:346:9:
    |
346 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
```

```
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:358:9:
    |
358 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:373:9:
    |
373 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:387:9:
    |
387 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:401:9:
    |
401 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:416:9:
    |
416 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:417:9:
    |
417 |         uint256 notional = _notional(market, data, position);
    |         ^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:394:5:
    |
394 |     function notional(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:434:9:
    |
434 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
Note: The other declaration is here:

   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:450:9:
    |
450 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.

   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:464:9:
    |
464 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:481:9:
    |
481 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:484:9:
    |
484 |         uint256 value = _valueForLiquidations(market, data, position);
    |         ^^^^^^^^^^^^
Note: The other declaration is here:

   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:380:5:
    |
380 |     function value(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.

   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:485:9:
    |
485 |         uint256 maintenanceMargin = _maintenanceMargin(market, position);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:457:5:
    |
457 |     function maintenanceMargin(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:486:9:
    |
486 |         uint256 liquidationFee = _liquidationFee(market, data, position);
    |         ^^^^^^^^^^^^^^^^^^^^^^
```

```
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:443:5:
    |
443 |     function liquidationFee(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:499:9:
    |
499 |         Position.Info memory position = _getPosition(market, owner, id);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:318:5:
    |
318 |     function position(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:504:9:
    |
504 |         uint256 maintenanceMargin = _maintenanceMargin(market, position);
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:457:5:
    |
457 |     function maintenanceMargin(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:507:9:
    |
507 |         uint256 oi = _oi(market, position);
    |         ^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:353:5:
    |
353 |     function oi(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (8760): This declaration has the same name as another declaration.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:508:9:
    |
508 |         uint256 collateral = _collateral(market, position);
    |         ^^^^^^^^^^^^^^^^^^
Note: The other declaration is here:
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:368:5:
    |
368 |     function collateral(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (2519): This declaration shadows an existing declaration.
   --> test/ShivaBase.t.sol:167:9:
    |
167 |         OverlayV1Token ovlToken = new OverlayV1Token();
    |         ^^^^^^^^^^^^^^^^^^^^^^^
Note: The shadowed declaration is here:
  --> test/ShivaBase.t.sol:56:5:
   |
56 |     IOverlayV1Token ovlToken;
   |     ^^^^^^^^^^^^^^^^^^^^^^^^

Warning (9302): Return value of low-level calls not used.
  --> test/utils/MarketImpersonator.sol:20:9:
   |
20 |         address(_shiva).call(
   |         ^ (Relevant source part starts here and spans across multiple lines).

Warning (9302): Return value of low-level calls not used.
```

```
  --> test/UUPSProxy.t.sol:97:9:
   |
97 |         address(proxy).call(abi.encodeWithSignature("setMagicNumber(uint256)", 42));
   |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Warning (9302): Return value of low-level calls not used.
   --> test/UUPSProxy.t.sol:111:9:
    |
111 |         address(proxy).call(abi.encodeWithSignature("upgradeTo(address)", address(shivaV2)));
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Warning (9302): Return value of low-level calls not used.
   --> test/UUPSProxy.t.sol:117:9:
    |
117 |         address(proxy).call(abi.encodeWithSignature("upgradeTo(address)", address(shivaV2)));
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Warning (9302): Return value of low-level calls not used.
   --> test/UUPSProxy.t.sol:133:9:
    |
133 |         address(proxy).call(abi.encodeWithSignature("setMagicString(string)", "Test"));
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Warning (2072): Unused local variable.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:234:9:
    |
234 |         uint256 oi = position.oiCurrent(fraction, oiTotalOnSide, oiTotalSharesOnSide);
    |         ^^^^^^^^^^

Warning (2072): Unused local variable.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:463:9:
    |
463 |         Oracle.Data memory data = _getOracleData(feed);
    |         ^^^^^^^^^^^^^^^^^^^^^^^

Warning (2072): Unused local variable.
   --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:498:9:
    |
498 |         address feed = market.feed();
    |         ^^^^^^^^^^^^

Warning (5667): Unused function parameter. Remove or comment out the variable name to silence this warning.
  --> test/utils/MarketImpersonator.sol:25:24:
   |
25 |     function positions(bytes32 key)
   |                        ^^^^^^^^^^^

Warning (5667): Unused function parameter. Remove or comment out the variable name to silence this warning.
  --> test/utils/MarketImpersonator.sol:29:13:
   |
29 |             uint96 debtInitial_,
   |             ^^^^^^^^^^^^^^^^^^^

Warning (5667): Unused function parameter. Remove or comment out the variable name to silence this warning.
  --> test/utils/MarketImpersonator.sol:30:13:
   |
30 |             int24 midTick_,
   |             ^^^^^^^^^^^^^^

Warning (5667): Unused function parameter. Remove or comment out the variable name to silence this warning.
  --> test/utils/MarketImpersonator.sol:31:13:
   |
31 |             int24 entryTick_,
   |             ^^^^^^^^^^^^^^^^

Warning (5667): Unused function parameter. Remove or comment out the variable name to silence this warning.
  --> test/utils/MarketImpersonator.sol:32:13:
   |
32 |             bool isLong_,
   |             ^^^^^^^^^^^^
```

```
Warning (5667): Unused function parameter. Remove or comment out the variable name to silence this warning.
  --> test/utils/MarketImpersonator.sol:33:13:
   |
33 |             bool liquidated_,
   |             ^^^^^^^^^^^^^^^^

Warning (5667): Unused function parameter. Remove or comment out the variable name to silence this warning.
  --> test/utils/MarketImpersonator.sol:34:13:
   |
34 |             uint240 oiShares_,
   |             ^^^^^^^^^^^^^^^^^

Warning (2018): Function state mutability can be restricted to pure
  --> lib/v1-periphery/contracts/state/OverlayV1PriceState.sol:41:5:
   |
41 |     function _mid(Oracle.Data memory data) internal view returns (uint256 mid_) {
   |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (2018): Function state mutability can be restricted to pure
  --> lib/v1-periphery/contracts/state/OverlayV1EstimateState.sol:70:5:
   |
70 |     function _debtEstimate(Position.Info memory position) internal view returns (uint256 debt_) {
   |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (2018): Function state mutability can be restricted to pure
  --> lib/v1-periphery/contracts/state/OverlayV1EstimateState.sol:78:5:
   |
78 |     function _costEstimate(Position.Info memory position) internal view returns (uint256 cost_) {
   |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (2018): Function state mutability can be restricted to pure
  --> lib/v1-periphery/contracts/state/OverlayV1EstimateState.sol:86:5:
   |
86 |     function _oiEstimate(Position.Info memory position) internal view returns (uint256 oi_) {
   |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (2018): Function state mutability can be restricted to pure
  --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:58:5:
   |
58 |     function _debt(Position.Info memory position) internal view returns (uint256 debt_) {
   |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (2018): Function state mutability can be restricted to pure
  --> lib/v1-periphery/contracts/state/OverlayV1PositionState.sol:67:5:
   |
67 |     function _cost(Position.Info memory position) internal view returns (uint256 cost_) {
   |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (2018): Function state mutability can be restricted to view
  --> test/utils/MarketImpersonator.sol:25:5:
   |
25 |     function positions(bytes32 key)
   |     ^ (Relevant source part starts here and spans across multiple lines).
```

Shown warnings highlight possible changes in the tests and libs contracts. They do not show any relevant risks for the contracts.

8.2 Tests Output

```
> forge test
[] Compiling...
No files changed, compilation skipped

Ran 2 tests for test/UUPSProxy.t.sol:ImplementationV2Test
[PASS] testMagicNumber() (gas: 15687)
[PASS] testMagicString() (gas: 40362)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 1.12s (3.86ms CPU time)

Ran 1 test for test/UUPSProxy.t.sol:ImplementationV1Test
[PASS] testInitialized() (gas: 18903)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.12s (2.56ms CPU time)

Ran 33 tests for test/Shiva.t.sol:ShivaTest
[PASS] testFuzz_buildSingle(uint256,uint256) (runs: 256, : 1320216, ~: 1320290)
[PASS] testFuzz_partial_unwind_pol_unstake(uint256,uint256,uint256) (runs: 256, : 1369335, ~: 1369526)
[PASS] test_addAuthorizedFactory() (gas: 54148)
[PASS] test_addAuthorizedFactory_notGovernor() (gas: 23836)
[PASS] test_build() (gas: 765356)
[PASS] test_buildSingle() (gas: 1312152)
[PASS] test_buildSingle_leverageLessThanMinimum() (gas: 755766)
[PASS] test_buildSingle_noOVL() (gas: 2185978)
[PASS] test_buildSingle_noPreviousPosition() (gas: 23252)
[PASS] test_buildSingle_pausedShiva() (gas: 785290)
[PASS] test_buildSingle_priceLimitOverLimit() (gas: 1711652)
[PASS] test_build_afterUnpause() (gas: 779012)
[PASS] test_build_leverageLessThanMinimum() (gas: 248049)
[PASS] test_build_noOVL() (gas: 1142407)
[PASS] test_build_notEnoughAllowance() (gas: 410224)
[PASS] test_build_notEnoughBalance() (gas: 433996)
[PASS] test_build_pausedShiva() (gas: 247468)
[PASS] test_build_pol_stake() (gas: 757823)
[PASS] test_build_pol_stake_revert_user_withdraw() (gas: 763824)
[PASS] test_build_unauthorizedFactory() (gas: 40452)
[PASS] test_emergencyWithdraw() (gas: 729593)
[PASS] test_emergencyWithdraw_notOwner() (gas: 794202)
[PASS] test_emergencyWithdraw_pausedShiva() (gas: 820645)
[PASS] test_partial_unwind() (gas: 999446)
[PASS] test_partial_unwind_pol_unstake() (gas: 1326756)
[PASS] test_pol_withdraw_emergencyWithdraw_function() (gas: 958086)
[PASS] test_removeAuthorizedFactory() (gas: 28998)
[PASS] test_removeAuthorizedFactory_notGovernor() (gas: 23865)
[PASS] test_shivaApproveRewardVault() (gas: 764419)
[PASS] test_unwind() (gas: 856945)
[PASS] test_unwind_notOwner(bool) (runs: 256, : 881484, ~: 880520)
[PASS] test_unwind_pausedShiva() (gas: 863556)
[PASS] test_unwind_pol_unstake() (gas: 864614)
Suite result: ok. 33 passed; 0 failed; 0 skipped; finished in 5.65s (6.60s CPU time)

Ran 2 tests for test/invariants/ShivaCryticToFoundry.sol:ShivaCryticToFoundry
[PASS] invariant_shiva_dont_have_ov() (runs: 10, calls: 1000, reverts: 829)
[PASS] invariant_staking_balance_matches_notional() (runs: 10, calls: 1000, reverts: 820)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 246.26s (483.51s CPU time)

Ran 20 tests for test/ShivaOnBehalfOf.t.sol:ShivaOnBehalfOfTest
[PASS] test_buildOnBehalfOf_expiredDeadline() (gas: 265392)
[PASS] test_buildOnBehalfOf_invalidParams() (gas: 562130)
[PASS] test_buildOnBehalfOf_invalidSignature_badNonce() (gas: 271985)
[PASS] test_buildOnBehalfOf_invalidSignature_badOwner() (gas: 271995)
[PASS] test_buildOnBehalfOf_ownership() (gas: 809683)
[PASS] test_buildOnBehalfOf_pausedShiva() (gas: 264908)
[PASS] test_buildOnBehalfOf_unauthorizedFactory() (gas: 251324)
[PASS] test_buildSingleOnBehalfOf_expiredDeadline() (gas: 974956)
[PASS] test_buildSingleOnBehalfOf_invalidParams() (gas: 1375123)
[PASS] test_buildSingleOnBehalfOf_invalidSignature_badNonce() (gas: 982057)
[PASS] test_buildSingleOnBehalfOf_invalidSignature_badOwner() (gas: 982020)
[PASS] test_buildSingleOnBehalfOf_ownership() (gas: 1336644)
[PASS] test_buildSingleOnBehalfOf_pausedShiva() (gas: 1003568)
[PASS] test_emergencyWithdraw_onBehalfOf() (gas: 731148)
[PASS] test_unwindOnBehalfOf_expiredDeadline() (gas: 852344)
[PASS] test_unwindOnBehalfOf_invalidParams() (gas: 1216082)
```

```
[PASS] test_unwindOnBehalfOf_invalidSignature_badNonce() (gas: 859395)
[PASS] test_unwindOnBehalfOf_invalidSignature_badOwner() (gas: 859358)
[PASS] test_unwindOnBehalfOf_pausedShiva() (gas: 880944)
[PASS] test_unwindOnBehalfOf_withdrawal() (gas: 897307)
Suite result: ok. 20 passed; 0 failed; 0 skipped; finished in 246.26s (678.85ms CPU time)

Ran 37 tests for test/ShivaLocal.t.sol:ShivaLocalTest
[PASS] testFuzz_buildSingle(uint256,uint256) (runs: 256, : 1445629, ~: 1445728)
[PASS] testFuzz_partial_unwind_pol_unstake(uint256,uint256,uint256) (runs: 256, : 1452744, ~: 1453401)
[PASS] test_addAuthorizedFactory() (gas: 54148)
[PASS] test_addAuthorizedFactory_notGovernor() (gas: 23836)
[PASS] test_build() (gas: 837462)
[PASS] test_buildSingle() (gas: 1437456)
[PASS] test_buildSingle_leverageLessThanMinimum() (gas: 827872)
[PASS] test_buildSingle_noOVL() (gas: 2309562)
[PASS] test_buildSingle_noPreviousPosition() (gas: 23274)
[PASS] test_buildSingle_pausedShiva() (gas: 857396)
[PASS] test_buildSingle_priceLimitOverLimit() (gas: 1835495)
[PASS] test_build_afterUnpause() (gas: 851118)
[PASS] test_build_leverageLessThanMinimum() (gas: 234544)
[PASS] test_build_noOVL() (gas: 1212829)
[PASS] test_build_notEnoughAllowance() (gas: 396921)
[PASS] test_build_notEnoughBalance() (gas: 420604)
[PASS] test_build_pausedShiva() (gas: 233985)
[PASS] test_build_pol_stake() (gas: 829907)
[PASS] test_build_pol_stake_revert_user_withdraw() (gas: 835908)
[PASS] test_build_unauthorizedFactory() (gas: 40474)
[PASS] test_emergencyWithdraw() (gas: 787067)
[PASS] test_emergencyWithdraw_notOwner() (gas: 866286)
[PASS] test_emergencyWithdraw_pausedShiva() (gas: 892729)
[PASS] test_liquidate_pausedShiva() (gas: 1149398)
[PASS] test_liquidate_pol() (gas: 1126126)
[PASS] test_partial_unwind() (gas: 1121147)
[PASS] test_partial_unwind_pol_unstake() (gas: 1397180)
[PASS] test_pol_overlayMarketLiquidateCallback_called_by_impersonator_market() (gas: 1022298)
[PASS] test_pol_withdraw_emergencyWithdraw_function() (gas: 1062456)
[PASS] test_removeAuthorizedFactory() (gas: 28932)
[PASS] test_removeAuthorizedFactory_notGovernor() (gas: 23887)
[PASS] test_revert_pol_overlayMarketLiquidateCallback_called_by_non_market() (gas: 839584)
[PASS] test_shivaApproveRewardVault() (gas: 836503)
[PASS] test_unwind() (gas: 954246)
[PASS] test_unwind_notOwner(bool) (runs: 256, : 953819, ~: 953467)
[PASS] test_unwind_pausedShiva() (gas: 935663)
[PASS] test_unwind_pol_unstake() (gas: 961932)
Suite result: ok. 37 passed; 0 failed; 0 skipped; finished in 246.26s (3.93s CPU time)

Ran 6 test suites in 246.29s (746.66s CPU time): 95 tests passed, 0 failed, 0 skipped (95 total tests)
```

8.3 Automated Tools

8.3.1 AuditAgent

All the relevant issues raised by the AuditAgent have been incorporated into this report. The AuditAgent is an AI-powered smart contract auditing tool that analyses code, detects vulnerabilities, and provides actionable fixes. It accelerates the security analysis process, complementing human expertise with advanced AI models to deliver efficient and comprehensive smart contract audits. Available at https://app.auditagent.nethermind.io.

9 About Nethermind

Nethermind is a Blockchain Research and Software Engineering company. Our work touches every part of the web3 ecosystem - from layer 1 and layer 2 engineering, cryptography research, and security to application-layer protocol development. We offer strategic support to our institutional and enterprise partners across the blockchain, digital assets, and DeFi sectors, guiding them through all stages of the research and development process, from initial concepts to successful implementation.

We offer security audits of projects built on EVM-compatible chains and Starknet. We are active builders of the Starknet ecosystem, delivering a node implementation, a block explorer, a Solidity-to-Cairo transpiler, and formal verification tooling. Nethermind also provides strategic support to our institutional and enterprise partners in blockchain, digital assets, and decentralized finance (DeFi). In the next paragraphs, we introduce the company in more detail.

Blockchain Security: At Nethermind, we believe security is vital to the health and longevity of the entire Web3 ecosystem. We provide security services related to Smart Contract Audits, Formal Verification, and Real-Time Monitoring. Our Security Team comprises blockchain security experts in each field, often collaborating to produce comprehensive and robust security solutions. The team has a strong academic background, can apply state-of-the-art techniques, and is experienced in analyzing cutting-edge Solidity and Cairo smart contracts, such as ArgentX and StarkGate (the bridge connecting Ethereum and StarkNet). Most team members hold a Ph.D. degree and actively participate in the research community, accounting for 240+ articles published and 1,450+ citations in Google Scholar. The security team adopts customer-oriented and interactive processes where clients are involved in all stages of the work.

Blockchain Core Development: Our core engineering team, consisting of over 20 developers, maintains, improves, and upgrades our flagship product - the Nethermind Ethereum Execution Client. The client has been successfully operating for several years, supporting both the Ethereum Mainnet and its testnets, and now accounts for nearly a quarter of all synced Mainnet nodes. Our unwavering commitment to Ethereum's growth and stability extends to sidechains and layer 2 solutions. Notably, we were the sole execution layer client to facilitate Gnosis Chain's Merge, transitioning from Aura to Proof of Stake (PoS), and we are actively developing a full-node client to bolster Starknet's decentralization efforts. Our core team equips partners with tools for seamless node set-up, using generated docker-compose scripts tailored to their chosen execution client and preferred configurations for various network types.

DevOps and Infrastructure Management: Our infrastructure team ensures our partners' systems operate securely, reliably, and efficiently. We provide infrastructure design, deployment, monitoring, maintenance, and troubleshooting support, allowing you to focus on your core business operations. Boasting extensive expertise in Blockchain as a Service, private blockchain implementations, and node management, our infrastructure and DevOps engineers are proficient with major cloud solution providers and can host applications in-house or on clients' premises. Our global in-house SRE teams offer 24/7 monitoring and alerts for both infrastructure and application levels. We manage over 5,000 public and private validators and maintain nodes on major public blockchains such as Polygon, Gnosis, Solana, Cosmos, Near, Avalanche, Polkadot, Aptos, and StarkWare L2. Sedge is an open-source tool developed by our infrastructure experts, designed to simplify the complex process of setting up a proof-of-stake (PoS) network or chain validator. Sedge generates docker-compose scripts for the entire validator set-up based on the chosen client, making the process easier and quicker while following best practices to avoid downtime and being slashed.

Cryptography Research: At Nethermind, our Cryptography Research team is dedicated to continuous internal research while fostering close collaboration with external partners. The team has expertise across a wide range of domains, including cryptography protocols, consensus design, decentralized identity, verifiable credentials, Sybil resistance, oracles, and credentials, distributed validator technology (DVT), and Zero-knowledge proofs. This diverse skill set, combined with strong collaboration between our engineering teams, enables us to deliver cutting-edge solutions to our partners and clients.

Smart Contract Development & DeFi Research: Our smart contract development and DeFi research team comprises 40+ world-class engineers who collaborate closely with partners to identify needs and work on value-adding projects. The team specializes in Solidity and Cairo development, architecture design, and DeFi solutions, including DEXs, AMMs, structured products, derivatives, and money market protocols, as well as ERC20, 721, and 1155 token design. Our research and data analytics focuses on three key areas: technical due diligence, market research, and DeFi research. Utilizing a data-driven approach, we offer in-depth insights and outlooks on various industry themes.

Our suite of L2 tooling: Warp is Starknet's approach to EVM compatibility. It allows developers to take their Solidity smart contracts and transpile them to Cairo, Starknet's smart contract language. In the short time since its inception, the project has accomplished many achievements, including successfully transpiling Uniswap v3 onto Starknet using Warp.

- **Voyager** is a user-friendly Starknet block explorer that offers comprehensive insights into the Starknet network. With its intuitive interface and powerful features, Voyager allows users to easily search for and examine transactions, addresses, and contract details. As an essential tool for navigating the Starknet ecosystem, Voyager is the go-to solution for users seeking in-depth information and analysis;
- **Horus** is an open-source formal verification tool for StarkNet smart contracts. It simplifies the process of formally verifying Starknet smart contracts, allowing developers to express various assertions about the behavior of their code using a simple assertion language;
- **Juno** is a full-node client implementation for Starknet, drawing on the expertise gained from developing the Nethermind Client. Written in Golang and open-sourced from the outset, Juno verifies the validity of the data received from Starknet by comparing it to proofs retrieved from Ethereum, thus maintaining the integrity and security of the entire ecosystem.

Learn more about us at nethermind.io.

General Advisory to Clients

As auditors, we recommend that any changes or updates made to the audited codebase undergo a re-audit or security review to address potential vulnerabilities or risks introduced by the modifications. By conducting a re-audit or security review of the modified codebase, you can significantly enhance the overall security of your system and reduce the likelihood of exploitation. However, we do not possess the authority or right to impose obligations or restrictions on our clients regarding codebase updates, modifications, or subsequent audits. Accordingly, the decision to seek a re-audit or security review lies solely with you.

Disclaimer

This report is based on the scope of materials and documentation provided by you to Nethermind in order that Nethermind could conduct the security review outlined in **1. Executive Summary** and **2. Audited Files**. The results set out in this report may not be complete nor inclusive of all vulnerabilities. Nethermind has provided the review and this report on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. This report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, Nethermind disclaims any liability in connection with this report, its content, and any related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Nethermind does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and Nethermind will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.