Security Review Report
NM-0362-BERACHAIN-BEX

NETHERMIND
SECURITY

(Dec 06, 2024)

Contents

1 Executive Summary **2**

2 Audited Files **3**
2.1 balancer-v2-vault-v2 . . . 3
2.2 balancer-v2-weighted-pool-v5 . . . 3
2.3 balancer-v2-monorepo . . . 3
2.4 balancer-deployments . . . 3

3 Summary of Issues **3**

4 System Overview **4**
4.1 Vault v2 . . . 4
4.2 Weighted Pool v5 . . . 4
4.3 Protocol Fees Withdrawer v2 . . . 4
4.4 Protocol Fee Percentages Provider v2 . . . 4
4.5 Pool Creation Helper . . . 4

5 Risk Rating Methodology **5**

6 Issues **6**
6.1 [Info] The maximum pause window in a weighted pool can never be reached . . . 6

7 Documentation Evaluation **7**

8 Test Suite Evaluation **8**
8.1 Compilation Output . . . 8
8.2 Tests Output . . . 10
8.2.1 Slither . . . 14
8.2.2 AuditAgent . . . 14

9 About Nethermind **15**

1 Executive Summary

This document presents the security review performed by Nethermind Security for Berachain BEX smart contracts. Berachain BEX is a native exchange on the Berachain blockchain, developed as a fork of the Balancer v2 contracts with specific modifications. It includes the core components of Balancer v2, notably the `Vault` contract, and supports two types of pools: Composable Stable Pools and Weighted Pools.

This security review focuses exclusively on the smart contracts and tasks listed in Section 2 (*Audited Files*).

The audit was performed using (a) manual analysis of the codebase and (b) creation of test cases. **Along this document, we report** one point of attention, 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.

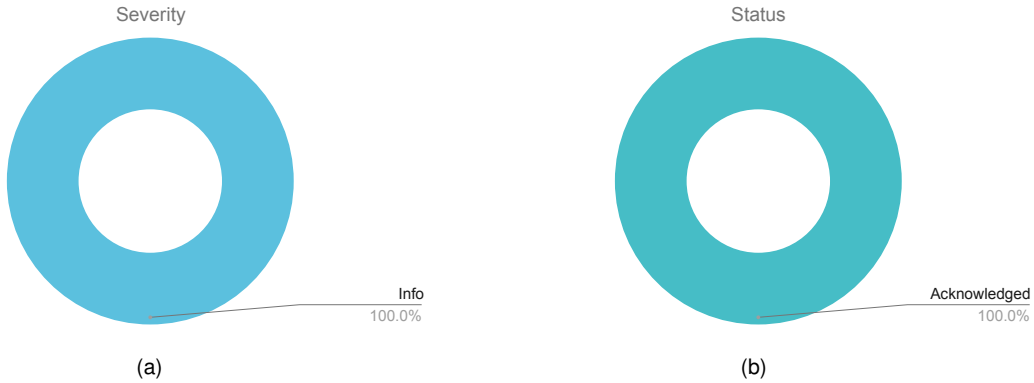

Fig. 1: Distribution of issues: Critical (0), **High** (0), **Medium** (0), **Low** (0), **Undetermined** (0), **Informational** (1), **Best Practices** (0). **Distribution of status: Fixed** (0), **Acknowledged** (1), **Mitigated** (0), **Unresolved** (0)

Summary of the Audit

Audit Type	Security Review
Initial Report	Dec 06, 2024
Response from Client	Regular responses during audit engagement
Final Report	Dec 06, 2024
Repository	balancer-v2-vault-v2, balancer-v2-weighted-pool-v5, balancer-v2-monorepo, balancer-deployments
Commit (Audit)	**balancer-v2-vault-v2:** fec2011e2e83375dbe3f779da230da0247906b19 **balancer-v2-weighted-pool-v5** 70f425650ca8bf51e02373173e1120e4e7052602 **balancer-v2-monorepo** 41378bfae9e06f18567b3d3adf322942fbde1ef0 **balancer-deployments** c243be8b109387c2c7f20bc685e90f0f0519bdce
Commit (Final)	**balancer-v2-vault-v2:** fec2011e2e83375dbe3f779da230da0247906b19 **balancer-v2-weighted-pool-v5** 70f425650ca8bf51e02373173e1120e4e7052602 **balancer-v2-monorepo** 41378bfae9e06f18567b3d3adf322942fbde1ef0 **balancer-deployments** c243be8b109387c2c7f20bc685e90f0f0519bdce
Documentation Assessment	High
Test Suite Assessment	Medium

2 Audited Files

	Contract	LoC	Comments	Ratio	Blank	Total
1	v2-solidity-utils/contracts/helpers/TemporarilyPausable.sol	56	53	94.6%	20	129
2	contracts/vault/ProtocolFeesCollector.sol	70	27	38.6%	23	120
3	pkg/standalone-utils/contracts/ProtocolFeesWithdrawer.sol	109	66	60.6%	29	204
4	pkg/standalone-utils/contracts/ProtocolFeePercentagesProvider.sol	119	23	19.3%	30	172
5	pkg/standalone-utils/contracts/PoolCreationHelper.sol	129	96	74.4%	25	250
6	v2-pool-utils/contracts/factories/FactoryWidePauseWindow.sol	20	32	160.0%	10	62
	Total	**503**	**297**	**59.0%**	**137**	**937**

This audit covered the following tasks, organized by repositories:

2.1 balancer-v2-vault-v2

- Verification that commit 34d308247c4b848987d150d89c514e7ef8d3cd37 in the `balancer-v2-vault-v2` repository implements the same vault code as Balancer mainnet deployment, defined in commit d81a658c95e0ed3c3724d90dafb40869c9550523 of the `balancer-deployments` repository.
- Review of the changes implemented in commit fec2011e2e83375dbe3f779da230da0247906b19.

2.2 balancer-v2-weighted-pool-v5

- Verification that commit 5e15464e08a69f48511f1426638a0739299a5c5d of `balancerv2-weighted-pool-v5` repository implements the same Weighted Pool and Factory code as Balancer mainnet deployment, defined in commit d81a658c95e0ed3c3724d90dafb40869c9550523 of the `balancer-deployments` repository.
- Review of the changes implemented in commit 70f425650ca8bf51e02373173e1120e4e7052602.

2.3 balancer-v2-monorepo

- Review of the changes implemented in commit 41378bfae9e06f18567b3d3adf322942fbde1ef0 for the two standalone contracts: `ProtocolFeePercentagesProvider` and `ProtocolFeesWithdrawer`, as well as the new contract `PoolCreationHelper`. The changes are compared to the mainnet deployments defined in commit d81a658c95e0ed3c3724d90dafb40869c9550523.

2.4 balancer-deployments

- Review of the deployment tasks in commit c243be8b109387c2c7f20bc685e90f0f0519bdce, ensuring that the bytecode for modified contracts reflects the latest implementation, while the bytecode for unchanged contracts remains the same.

3 Summary of Issues

	Finding	Severity	Update
1	The maximum pause window in a weighted pool can never be reached	Info	Acknowledged

4 System Overview

Berachain BEX is a fork of Blanacer v2, which will be deployed on Berachain chains. It incorporates some modifications to the core Balancer components. The main components of Balancer that are relevant to BEX include:

- The `Vault` contract, which is the core of balancer protocol, facilitates interactions with different liquidity pools.
- The `WeightedPoolFactory` and `ComposableStablePool`, the two pool types supported by Berachain BEX.

The following subsections provide an overview of the modifications introduced by the Berachain team to the forked Balancer v2 code.

4.1 Vault v2

The `Vault` contract is the core contract of the protocol, responsible for holding and managing the tokens in each Balancer pool. It is the gateway for executing most operations, such as swaps, joins, and exits.

The Berachain team introduced the following changes:

- **Extended Pause Window and Buffer Period**: The vault's pause window and buffer period durations have been increased.

```
    uint256 private constant _MAX_PAUSE_WINDOW_DURATION =  5 * 365 days;
    uint256 private constant _MAX_BUFFER_PERIOD_DURATION = 365 days;
```

- **Increased Flashloan Fee**: The flashloan fee is set to 100%

```
    uint256 private constant _MAX_PROTOCOL_FLASH_LOAN_FEE_PERCENTAGE = 1e18; // 100%
```

4.2 Weighted Pool v5

The `WeightedPool` is one of the two supported pool types in Berachain BEX. The Berachain team has updated the pause window and buffer period durations at both the pool and factory levels. The Factory contract defines the constant `_INITIAL_PAUSE_WINDOW_DURATION`, which represents the maximum allowable pause window for pools created by the factory. When a new pool is deployed, its pause window duration is set based on the remaining time from the `_INITIAL_PAUSE_WINDOW_DURATION`, starting from the factory's deployment time.

```
    uint256 private constant _INITIAL_PAUSE_WINDOW_DURATION = 4 * 365 days;
    uint256 private constant _BUFFER_PERIOD_DURATION = 180 days;
```

Each weighted pool contract inherits from the `TemporarilyPausable` contract, which sets a maximum pause window limit of 5 years:

```
    uint256 private constant _MAX_PAUSE_WINDOW_DURATION = 5 * 365 days;
    uint256 private constant _MAX_BUFFER_PERIOD_DURATION = 365 days;
```

4.3 Protocol Fees Withdrawer v2

The `ProtocolFeeWithdrawer` is a standalone contract implementing a denylist of tokens that cannot be withdrawn from the Protocol fee Collector. The Berachain team updated this contract to include a fee distribution mechanism. This mechanism splits the collected fees between the `polFeeCollector` and `feeReceiver` contracts, based on a predefined percentage `polFeeCollectorPercentage`. The distribution and withdrawal process is handled within the `distributeAndWithdrawCollectedFees(...)` function.

```
    function distributeAndWithdrawCollectedFees(IERC20[] calldata tokens) external override authenticate
```

4.4 Protocol Fee Percentages Provider v2

The `ProtocolFeePercentagesProvider` contract offers functions to retrieve and update different fees within the protocol by interacting with the `protocolFeeCollector` contract. The Berachain team updated the `_MAX_PROTOCOL_FLASH_LOAN_FEE_PERCENTAGE` constant within this contract to allow setting the flash loan fee percentage to 100%.

```
    uint256 private constant _MAX_PROTOCOL_FLASH_LOAN_FEE_PERCENTAGE = 1e18; // 100%
```

4.5 Pool Creation Helper

The `PoolCreationHelper` contract is a new standalone contract introduced by the Berachain team. It simplifies pool creation by enabling users to create and join a pool in a single transaction. The contract exposes the following two key functions:

- **createAndJoinWeightedPool(...)**: Allows the creation of a weighted pool and joining it in one transaction.
- **createAndJoinStablePool(...)**: Allows creating a stable pool and joining it in one transaction.

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] The maximum pause window in a weighted pool can never be reached

File(s): TemporarilyPausable.sol

Description: The FactoryWidePauseWindow contract defines the constant _INITIAL_PAUSE_WINDOW_DURATION, which represents the maximum allowable pause window for pools deployed by the factory. Currently, this is set to 4 years.

When a new pool is created, its pause window duration is determined by the remaining time from the _INITIAL_PAUSE_WINDOW_DURATION, starting from the factory's deployment time.

```
function getPauseConfiguration() public view returns (uint256 pauseWindowDuration, uint256 bufferPeriodDuration) {
    uint256 currentTime = block.timestamp;
    if (currentTime < _poolsPauseWindowEndTime) {
        pauseWindowDuration = _poolsPauseWindowEndTime - currentTime;
        bufferPeriodDuration = _BUFFER_PERIOD_DURATION;
    } else {
        pauseWindowDuration = 0;
        bufferPeriodDuration = 0;
    }
}
```

Each weighted pool contract inherits from the TemporarilyPausable contract, which sets an additional maximum pause window limit of 5 years:

```
uint256 private constant _MAX_PAUSE_WINDOW_DURATION = 5 * 365 days;
```

However, due to the way the pool's pause window is calculated (based on the factory's deployment time), the pause window can never exceed the initial 4-year period. This effectively makes the 5-year maximum pause window redundant and unused.

Recommendation(s): Consider updating the constant to ensure it aligns with how the pool pause window is calculated.

Status: Acknowledged

Update from the client: This is similar to composable stable V6 pools.

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 Berachain BEX documentation

The `Berachain` team provided a comprehensive Notion document detailing the key modifications made to the forked Balancer v2 code. Additionally, they proactively addressed all concerns and questions raised by the Nethermind Security team during their regular meetings.

8 Test Suite Evaluation

8.1 Compilation Output

balancer-v2-vault-v2

```
> npx hardhat compile
Generating typings for: 45 artifacts in dir: typechain-types for target: ethers-v6
Successfully generated 112 typings!
Compiled 45 Solidity files successfully (evm target: istanbul).
```

balancer-v2-weighted-pool-v5

```
> npx hardhat compile
Generating typings for: 56 artifacts in dir: typechain-types for target: ethers-v6
Successfully generated 122 typings!
Compiled 55 Solidity files successfully (evm target: istanbul).
```

balancer-v2-monorepo

```
> yarn build
[@balancer-labs/balancer-js]: Process started
[@balancer-labs/v2-governance-scripts]: Process started
[@balancer-labs/v2-interfaces]: Process started
[@balancer-labs/v2-liquidity-mining]: Process started
[@balancer-labs/v2-pool-linear]: Process started
[@balancer-labs/v2-pool-stable]: Process started
[@balancer-labs/v2-interfaces]: Compiled 98 Solidity files successfully
[@balancer-labs/v2-interfaces]: Process exited (exit code 0), completed in 8s 303ms
[@balancer-labs/v2-pool-utils]: Process started
[@balancer-labs/balancer-js]:
[@balancer-labs/balancer-js]: src/index.ts → dist/index.umd.js, dist/index.js, dist/index.esm.js...
[@balancer-labs/balancer-js]: (!) Missing global variable names
[@balancer-labs/balancer-js]: https://rollupjs.org/configuration-options/#output-globals
[@balancer-labs/balancer-js]: Use "output.globals" to specify browser global variable names corresponding to external
↪  modules:
[@balancer-labs/balancer-js]: @ethersproject/abi (guessing "abi")
[@balancer-labs/balancer-js]: @ethersproject/constants (guessing "constants")
[@balancer-labs/balancer-js]: @ethersproject/bignumber (guessing "bignumber")
[@balancer-labs/balancer-js]: @ethersproject/bytes (guessing "bytes")
[@balancer-labs/balancer-js]: @ethersproject/abstract-signer (guessing "abstractSigner")
[@balancer-labs/balancer-js]: @ethersproject/address (guessing "address")
[@balancer-labs/balancer-js]: created dist/index.umd.js, dist/index.js, dist/index.esm.js in 2.4s
[@balancer-labs/balancer-js]:
[@balancer-labs/balancer-js]: src/index.ts → dist/index.d.ts...
[@balancer-labs/balancer-js]: created dist/index.d.ts in 2.2s
[@balancer-labs/balancer-js]: Process exited (exit code 0), completed in 9s 687ms
[@balancer-labs/v2-pool-weighted]: Process started
[@balancer-labs/v2-governance-scripts]: Compiled 38 Solidity files successfully
[@balancer-labs/v2-governance-scripts]: Process exited (exit code 0), completed in 11s 15ms
[@balancer-labs/v2-solidity-utils]: Process started
[@balancer-labs/v2-pool-linear]: Compiled 47 Solidity files successfully
[@balancer-labs/v2-pool-linear]: Process exited (exit code 0), completed in 11s 481ms
[@balancer-labs/v2-standalone-utils]: Process started
[@balancer-labs/v2-pool-stable]: Compiled 73 Solidity files successfully
[@balancer-labs/v2-pool-stable]: contracts/ComposableStablePool.sol:53:1: Warning: Contract code size exceeds 24576
↪  bytes (a limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider enabling the
↪  optimizer (with a low "runs" value!), turning off revert strings, or using libraries.
[@balancer-labs/v2-pool-stable]: contract ComposableStablePool is
[@balancer-labs/v2-pool-stable]: ^ (Relevant source part starts here and spans across multiple lines).
[@balancer-labs/v2-pool-stable]:
[@balancer-labs/v2-pool-stable]: Process exited (exit code 0), completed in 20s 992ms
```

```
[@balancer-labs/v2-vault]: Process started
[@balancer-labs/v2-solidity-utils]: Compiled 77 Solidity files successfully
[@balancer-labs/v2-solidity-utils]: Process exited (exit code 0), completed in 11s 778ms
[@balancer-labs/v2-pool-utils]: Compiled 83 Solidity files successfully
[@balancer-labs/v2-pool-utils]: Process exited (exit code 0), completed in 14s 872ms
[@balancer-labs/v2-vault]: Compiled 69 Solidity files successfully
[@balancer-labs/v2-vault]: Process exited (exit code 0), completed in 12s 413ms
[@balancer-labs/v2-pool-weighted]: Compiled 101 Solidity files successfully
[@balancer-labs/v2-pool-weighted]: contracts/WeightedPool.sol:24:1: Warning: Contract code size exceeds 24576 bytes (a
↪  limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider enabling the
↪  optimizer (with a low "runs" value!), turning off revert strings, or using libraries.
[@balancer-labs/v2-pool-weighted]: contract WeightedPool is BaseWeightedPool, WeightedPoolProtocolFees {
[@balancer-labs/v2-pool-weighted]: ^ (Relevant source part starts here and spans across multiple lines).
[@balancer-labs/v2-pool-weighted]:
[@balancer-labs/v2-pool-weighted]: Process exited (exit code 0), completed in 30s 175ms
[@balancer-labs/v2-standalone-utils]: Compiled 168 Solidity files successfully
[@balancer-labs/v2-standalone-utils]: @balancer-labs/v2-pool-stable/contracts/ComposableStablePool.sol:53:1: Warning:
↪  Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable
↪  on mainnet. Consider enabling the optimizer (with a low "runs" value!), turning off revert strings, or using
↪  libraries.
[@balancer-labs/v2-standalone-utils]: contract ComposableStablePool is
[@balancer-labs/v2-standalone-utils]: ^ (Relevant source part starts here and spans across multiple lines).
[@balancer-labs/v2-standalone-utils]:
[@balancer-labs/v2-standalone-utils]: @balancer-labs/v2-pool-weighted/contracts/WeightedPool.sol:24:1: Warning:
↪  Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable
↪  on mainnet. Consider enabling the optimizer (with a low "runs" value!), turning off revert strings, or using
↪  libraries.
[@balancer-labs/v2-standalone-utils]: contract WeightedPool is BaseWeightedPool, WeightedPoolProtocolFees {
[@balancer-labs/v2-standalone-utils]: ^ (Relevant source part starts here and spans across multiple lines).
[@balancer-labs/v2-standalone-utils]:
[@balancer-labs/v2-standalone-utils]: Process exited (exit code 0), completed in 37s 847ms
[@balancer-labs/v2-liquidity-mining]: Compiled 130 Solidity files successfully
[@balancer-labs/v2-liquidity-mining]: Vyper compilation finished successfully
[@balancer-labs/v2-liquidity-mining]: Process exited (exit code 0), completed in 3m 16s
Done in 3m 16s
```

8.2 Tests Output

```
> yarn test
  PoolCreationHelper
    check set state
       should set the state correctly
    createAndJoinWeightedPool
       creates and joins a weighted pool
       create and join weighted pool with same token more than once
       create and join weighted `WBERA` pool with BERA (314ms)
       wbera pool creation fails if not enough BERA is sent (263ms)
       should not consume BERA if wbera pool is joined with wbera (300ms)
       should not consume wbera if wbera pool is joined with BERA (333ms)
    createAndJoinStablePool
       creates and joins a composable stable pool (347ms)
       create and join WBERA pool with BERA (401ms)
       wbera pool creation fails if not enough BERA is sent (257ms)
       should not consume BERA if wbera pool is joined with wbera (300ms)
       should not consume wbera if wbera pool is joined with BERA (308ms)

  PoolRecoveryHelper
    constructor
       supports no initial factories
       stores initial factories
    factory list
      add
         reverts if the caller does not have permission
         new factories can be added
         duplicate factories are rejected
      remove
         reverts if the caller does not have permission
         existing factories can be removed
         non-existent factories are rejected
    enable recovery mode
       reverts if the pool is not from a known factory
       reverts if none of the pool's rate providers reverts
       enables recovery mode on the pool if any of the rate providers revert

  ProtocolFeePercentagesProvider
    construction
       reverts if the maximum yield value is too high
       reverts if the maximum aum value is too high
       emits ProtocolFeeTypeRegistered events for custom types
       emits ProtocolFeePercentageChanged events for custom types
    with provider
      fee type configuration
        native fee types
          fee type Swap
             returns the fee type as valid
             returns the fee type name
             returns the fee type maximum value
          fee type FlashLoan
             returns the fee type as valid
             returns the fee type name
            1) returns the fee type maximum value
        custom fee types
          fee type Yield
             returns the fee type as valid
             returns the fee type name
             returns the fee type maximum value
             sets an initial value
          fee type AUM
             returns the fee type as valid
             returns the fee type name
             returns the fee type maximum value
             sets an initial value
        invalid fee type
           isValidFeeType returns false
           get name reverts
           get maximum reverts
```

```
is valid fee percentage
        native fee types
          fee type Swap
             returns true if the fee is below the maximum
             returns true if the fee equals the maximum
             returns false if the fee is above the maximum
          fee type FlashLoan
             returns true if the fee is below the maximum
             returns true if the fee equals the maximum
            2) returns false if the fee is above the maximum
        custom fee types
          fee type Yield
             returns true if the fee is below the maximum
             returns true if the fee equals the maximum
             returns false if the fee is above the maximum
          fee type AUM
             returns true if the fee is below the maximum
             returns true if the fee equals the maximum
             returns false if the fee is above the maximum
        invalid fee type
           reverts
      set fee type value
        native fee types
          fee type Swap
            when the caller is authorized
              when the provider is authorized
                when the value is below the maximum
                   sets the value
                   emits a ProtocolFeePercentageChanged event
                when the value is equal to the maximum
                   sets the value
                   emits a ProtocolFeePercentageChanged event
                when the value is above the maximum
                   reverts
              when the provider is not authorized
                 reverts
            when the caller is not authorized
               reverts
          fee type FlashLoan
            when the caller is authorized
              when the provider is authorized
                when the value is below the maximum
                   sets the value
                   emits a ProtocolFeePercentageChanged event
                when the value is equal to the maximum
                   sets the value
                   emits a ProtocolFeePercentageChanged event
                when the value is above the maximum
                  3) reverts
              when the provider is not authorized
                 reverts
            when the caller is not authorized
               reverts
        custom fee types
          fee type Yield
            when the caller is authorized
              when the value is below the maximum
                 sets the value
                 emits a ProtocolFeePercentageChanged event
              when the value is equal to the maximum
                 sets the value
                 emits a ProtocolFeePercentageChanged event
              when the value is above the maximum
                 reverts
            when the caller is not authorized
               reverts
          fee type AUM
            when the caller is authorized
              when the value is below the maximum
                 sets the value
                 emits a ProtocolFeePercentageChanged event
              when the value is equal to the maximum
                 sets the value
                 emits a ProtocolFeePercentageChanged event
```

```
            when the value is above the maximum
               reverts
          when the caller is not authorized
             reverts
      invalid fee type
         reverts
    native fee type out of band change
      swap fee
         the provider tracks value changes
      flash loan fee
         the provider tracks value changes
    register fee type
      when the caller is authorized
        when the fee type is already in use
           reverts
        when the maximum value is 0%
           reverts
        when the maximum value is above 100%
           reverts
        when the initial value is above the maximum value
           reverts
        when the new fee type data is valid
           returns registered data
           marks the fee type as valid
           emits a ProtocolFeeTypeRegistered event
           emits a ProtocolFeePercentageChanged event
           reverts on register attempt
           can change value after registration
      when the caller is not authorized
         reverts

ProtocolFeeSplitter
  constructor
    4) "before each" hook: deploy ProtocolFeeSplitter, ProtocolFeesWithdrawer & grant permissions at step "Running
    ↪  shared before each or reverting" for "sets the protocolFeesWithdrawer"

ProtocolFeesWithdrawer
  constructor
     lists the initially denylisted tokens
     reports the initial denylisted tokens as ineligible for withdrawal
  denylistToken
     adds the token to the denylist
     emits an event
     reverts if already denylisted
  allowlistToken
     removes the token from the denylist
     emits an event
     reverts if not denylisted
  withdrawCollectedFees
    when caller is authorized
      when attempting to claim allowlisted tokens
         withdraws the expected amount of tokens
      when attempting to claim denylisted tokens
         reverts
      when attempting to claim a mix of allowlisted and denylisted tokens
         reverts
      when tokens are later added from the denylist
         reverts
      when tokens are removed from the denylist
         allows withdrawing these tokens
    when caller is not authorized
       reverts
  distributeAndWithdrawCollectedFees
    when caller is authorized
       withdraws allowlisted tokens to POL fee collector
       withdraw allowlist tokens with `polFeeCollectorPercentage` as 50%
       reverts when attempting to withdraw denylisted tokens
      when attempting to claim a mix of allowlisted and denylisted tokens
         reverts
      when tokens are later added from the denylist
         reverts
      when tokens are removed from the denylist
         allows withdrawing these tokens
```

```
    when caller is not authorized
      reverts
  set PolFeeCollector
    caller is allowed to set the polFeeCollector
    reverts if zero address
    reverts if sender not allowed
  set feeReceiver
    caller is allowed to set the feeReceiver
    reverts if zero address
    reverts if sender not allowed
  set polFeeCollectorPercentage
    caller is allowed to set the polFeeCollectorPercentage
    reverts if percentage is greater than 100%
    reverts if sender not allowed

1455 passing (2m)
4 failing

1) ProtocolFeePercentagesProvider
     with provider
       fee type configuration
         native fee types
           fee type FlashLoan
             returns the fee type maximum value:

    AssertionError: Expected "10000000000000000" to be equal 10000000000000000
    + expected - actual
     {
    -  "_hex": "0x2386f26fc10000"
    +  "_hex": "0x0de0b6b3a7640000"
       "_isBigNumber": true
     }
    at Context.<anonymous> (test/ProtocolFeePercentagesProvider.test.ts:118:76)
    at processTicksAndRejections (node:internal/process/task_queues:95:5)
    at runNextTicks (node:internal/process/task_queues:64:3)
    at listOnTimeout (node:internal/timers:538:9)
    at processTimers (node:internal/timers:512:7)

2) ProtocolFeePercentagesProvider
     with provider
       is valid fee percentage
         native fee types
           fee type FlashLoan
             returns false if the fee is above the maximum:

    AssertionError: expected true to equal false
    + expected - actual
    -true
    +false
    at Context.<anonymous> (test/ProtocolFeePercentagesProvider.test.ts:171:89)
    at processTicksAndRejections (node:internal/process/task_queues:95:5)
    at runNextTicks (node:internal/process/task_queues:64:3)
    at listOnTimeout (node:internal/timers:538:9)
    at processTimers (node:internal/timers:512:7)

3) ProtocolFeePercentagesProvider
     with provider
       set fee type value
         native fee types
           fee type FlashLoan
             when the caller is authorized
               when the provider is authorized
                 when the value is above the maximum
                   reverts:
   AssertionError: Expected transaction to be reverted with Invalid fee percentage, but other exception was thrown:
    ↪  Error: VM Exception while processing transaction: reverted with reason string 'BAL#601'
    ↪  (FLASH_LOAN_FEE_PERCENTAGE_TOO_HIGH)

4) ProtocolFeeSplitter
     "before each" hook: deploy ProtocolFeeSplitter, ProtocolFeesWithdrawer & grant permissions at step "Running
      ↪  shared before each or reverting" for "sets the protocolFeesWithdrawer":
   Error: missing argument:  in Contract constructor (count=2, expectedCount=4, code=MISSING_ARGUMENT,
    ↪  version=contracts/5.7.0)
```

8.2.1 Slither

All the relevant issues raised by Slither have been incorporated into the issues described in this report.

8.2.2 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.