Morpho - `adapter-wsteth-eth`

Security Review

Cantina Managed review by:

Emanuele Ricci, Lead Security Researcher

Jonah1005, Lead Security Researcher

March 11, 2024

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Morpho is a lending pool optimizer. It improves the capital efficiency of positions on existing lending pools by seamlessly matching users peer-to-peer.

Morpho's rates stay between the supply rate and the borrow rate of the pool, reducing the interests paid by the borrowers while increasing the interests earned by the suppliers. It means that you are getting boosted peer-to-peer rates or, in the worst case scenario, the APY of the pool. Morpho also preserves the same experience, liquidity and parameters (collateral factors, oracles, ...) as the underlying pool.

From Feb 19th to Feb 23rd the Cantina team conducted a review of morpho-blue-oracles on commit hash d21e9403. The team identified a total of **5** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 0
- Gas Optimizations: 1
- Informational: 4

3 Findings

3.1 Gas Optimization

3.1.1 Consider using `stETH.getPooledEthByShares` directly instead of passing through `wstETH`

Severity: Gas Optimization

Context: WstEthEthExchangeRateChainlinkAdapter.sol#L29

Description: The current implementation of `WstEthEthExchangeRateChainlinkAdapter.latestRoundData` queries the `wstETH` contract to get the `wstETH/stETH` conversion rate. The underlying logic of such function inside `wstETH` directly calls `stETH.getSharesByPooledEth` as seen below:

```
/**
 * @notice Get amount of wstETH for a one stETH
 * @return Amount of wstETH for a 1 stETH
 */
function tokensPerStEth() external view returns (uint256) {
    return stETH.getSharesByPooledEth(1 ether);
}
```

This means that the current implementation of `WstEthEthExchangeRateChainlinkAdapter.latestRoundData` is performing an additional call that could be skipped by interacting directly with the `stETH` contract.

Recommendation: Given that `wstETH` is a non-upgradable contract, Morpho should consider replacing the current `WstEthEthExchangeRateChainlinkAdapter.latestRoundData` logic to directly call `stETH` and save gas:

```
  /// @dev Silently overflows if `stEthPerToken` is greater than `type(int256).max`.
  function latestRoundData() external view returns (uint80, int256, uint256, uint256, uint80) {
      // It is assumed that `stEthPerToken` returns a price with 18 decimals precision.
-     return (0, int256(WST_ETH.stEthPerToken()), 0, 0, 0);
+     return (0, int256(ST_ETH.getSharesByPooledEth(1 ether)), 0, 0, 0);
  }
```

Morpho: Addressed in PR 81.

Cantina Managed: The recommendations have been implemented in PR 81. The `latestRoundData` function now calls directly `ST_ETH.getPooledEthByShares`. Note that the `ST_ETH` will be declared as `constant` (instead of `immutable`) in PR 83.

3.2 Informational

3.2.1 `WstEthEthExchangeRateChainlinkAdapter` can only be deployed and used on Ethereum Mainnet

Severity: Informational

Context: WstEthEthExchangeRateChainlinkAdapter.sol

Description: Lido's documentation about `wstETH` on L2s explicitly says that

> Unlike on the Ethereum mainnet, wstETH on L2s is a plain ERC-20 token and cannot be unwrapped to unlock stETH on the corresponding L2 network.

This means that `WstEthEthExchangeRateChainlinkAdapter` will only work when deployed on the Ethereum Mainnet.

Recommendation: Morpho should be aware of this limitation and explicitly document this behavior. This oracle adapter can't be deployed a used on other chains and provides a `wstETH/stETH` conversion rate using the current logic.

Morpho: Addressed in PR 83.

Cantina Managed: The recommendations have been implemented in PR 83:

- The NatSpec documentation correctly suggests that this contract should only be deployed and used on Ethereum mainnet.
- The `ST_ETH` is now a constant variable.

3.2.2 `WST_ETH` should be declared `constant` and hardcoded at compilation time

Severity: Informational

Context: WstEthEthExchangeRateChainlinkAdapter.sol#L18

Description: The current implementation of `WstEthEthExchangeRateChainlinkAdapter` declares the `WST_ETH` variable as `immutable`. Such variable is initialized during the `constructor` with the `address wstEth` input parameter. The `WST_ETH` contract references is later used by `latestRoundData` to retrieve the conversion rate between `wstETH/stETH` to fulfill the role of this oracle contract.

This implementation pattern makes sense when contracts could have different addresses based on which chain they are deployed to, but it won't be the case for the Lido `wstETH` or `stETH` contracts. As explained in their documentation:

> Unlike on the Ethereum mainnet, wstETH on L2s is a plain ERC-20 token and cannot be unwrapped to unlock stETH on the corresponding L2 network.

Due to this, `WST_ETH` can be declared `constant` and hardcoded at compilation time.

Recommendation: Morpho should consider declaring `WST_ETH` as `constant` and hardcoding its initial value at compilation time.

Morpho: Addressed in PR 83.

Cantina Managed: The recommendations have been implemented in PR 83:

- Contract now uses `stETH` instead of `wstETH` to retrieve the exchange rate.
- `stETH` is declared as `constant` and not `immutable`.

3.2.3 Improve the NatSpec documentation about `latestRoundData` return values

Severity: Informational

Context: WstEthEthExchangeRateChainlinkAdapter.sol#L26-L30

Description: The `latestRoundData()` function in the `WstEthEthExchangeRateChainlinkAdapter` returns the following values to be compliant to the Chainlink interface signature as explained in their documentation:

- `uint80 roundId`
- `int256 answer`
- `uint256 startedAt`
- `uint256 updatedAt`
- `uint80 answeredInRound`

The current implementation of `WstEthEthExchangeRateChainlinkAdapter.latestRoundData` only returns a non-empty values for the `answer` returned variable, leaving all the others "empty" (using the Solidity default compiler value).

While this could be seen as a valid behavior for a custom Chainlink-like oracle, such behavior should be documented and explained.

Recommendation: Morpho should improve the natspec documentation about values returned by `latestRoundData`, documenting that only the `answer` return parameter will have a valid value, while the other parameters will be "empty".

An additional change that should be considered is to clarify that this is a "Chainlink-compliant" contract that does not fully follow the normal behavior of a Chainlink oracle that would always return non-empty values for all the returned parameters.

Morpho: Addressed in PR 83.

Cantina Managed: PR 83 correctly documents and clarifies the value (always `0`) returned by the named parameters `uint80 roundId`, `int256 answer`, `uint256 startedAt`, `uint256 updatedAt`, `uint80 answeredInRound`.

3.2.4 Consider documenting that `WstEthEthExchangeRateChainlinkAdapter` returns the conversion rate of `wstETH/stETH` and not `wstETH/ETH`

Severity: Informational

Context: WstEthEthExchangeRateChainlinkAdapter.sol#L12

Description: When `WstEthEthExchangeRateChainlinkAdapter.latestRoundData()` is called, the current implementation returns `int256(WST_ETH.stEthPerToken())` as the `answer` of a Chainlink-like response.

The Lido `wstETH` implementation of `stEthPerToken` returns the amount of `stETH` for a `one wstETH` (`1e18`):

```
function stEthPerToken() external view returns (uint256) {
    return stETH.getPooledEthByShares(1 ether);
}
```

This means that the Morpho's oracle is not returning the "price" for `wstETH/ETH` but the "price" (conversion rate would be the correct term) between `wstETH` and `stETH`.

The amount of `ETH` corresponding to `1 stETH` unit, received when the user finishes the withdrawal multi-step operation will be only known when the operation is **finalized** on the Lido withdraw queue.

Recommendation: Morpho has different options here

1. Rename the contract, docs and everything to reflect the fact that this is indeed a `wstETH/stETH` oracle.
2. Clarify and document that while this is a `wstETH/stETH` it will be used to "price" a `wstETH/ETH` conversion, implying that there's a hard assumption that `stETH:ETH` has a conversion ratio of `1:1`.

Morpho: Addressed in PR 83.

Cantina Managed: The recommendations have been implemented in PR 83. The contract name has been renamed from `WstEthEthExchangeRateChainlinkAdapter` to `WstEthStEthExchangeRateChainlinkAdapter` and all the NatSpec documentation and references have been correctly updated to reflect the fact that this is indeed an `wstETH/stETH` rate feed and not `wstETH/ETH` rate feed.