shieldify

Ion Protocol

SECURITY REVIEW

Date: 18 April 2024

CONTENTS

1. About Shieldify

Positioned as the first hybrid Web3 Security company, Shieldify shakes things up with a unique subscription-based auditing model that entitles the customer to unlimited audits within its duration, as well as top-notch service quality thanks to a disruptive 6-layered security approach working with the best talents in the space.

Learn more about us at shieldify.org.

2. Disclaimer

This security review does not guarantee bulletproof protection against a hack or exploit. Smart contracts are a novel technological feat with many known and unknown risks. The protocol, which this report is intended for, indemnifies Shieldify Security against any responsibility for any misbehavior, bugs, or exploits affecting the audited code during any part of the project's life cycle. It is also pivotal to acknowledge that modifications made to the audited code, including fixes for the issues described in this report, may introduce new problems and necessitate additional auditing.

3. About Ion Protocol

Ion Protocol is a decentralized money market purpose-built for all types of staked and re-staked assets. Ion protocol unlocks capital efficiency for yield-bearing staking collaterals using reactive interest rates, collateral-specific utilization, and price-agnostic liquidations. Borrowers can recursively borrow against their staked assets to increase their staking yields, while lenders can gain exposure to the boosted staking yield generated by borrower collateral.

Take a deep dive into ion's documentation here.

4. Risk Classification

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

4.1 Impact

- **High** – results in a significant risk for the protocol's overall well-being. Affects all or most users
- **Medium** – results in a non-critical risk for the protocol affects all or only a subset of users, but is still unacceptable
- **Low** – losses will be limited but bearable – and covers vectors similar to griefing attacks that can be easily repaired

4.2 Likelihood

- **High** – almost certain to happen and highly lucrative for execution by malicious actors
- **Medium** – still relatively likely, although only conditionally possible
- **Low** – requires a unique set of circumstances and poses non-lucrative cost-of-execution to rewards ratio for the actor

5. Security Review Summary

The bug bounty competition was conducted in the span of two weeks (February 28th until March 12th) and a total of 30 smart contract security researchers participated. This SPP has been the most recent layer in Ion's security methodology. The codebase had been audited multiple times beforehand by other reputable security companies like OpenZeppelin (twice), Hats Finance & Formal Verification Competition with Certora.

Ion security overview:

1. OpenZeppelin Audit December 2023 - Full Audit
2. Hats Finance January 2024 (Regular Audit and Formal Verification Competition with Certora)
3. OpenZeppelin Audit February 2024 - weETH Integration
4. Shieldify Private Pool(SPP)

Overall, the code is written to very high standards and implements very solid security practices. Shieldify's Private Pool competition contributed by identifying a medium-severity issue regarding an unsafe downcast truncation while interacting with the Uniswap Oracle Library.

Shieldify's team extends their gratitude to Ion's team for their exceptional work communication and responsiveness.

5.1 Protocol Summary

Project Name	**Ion protocol**
Repository	ion-protocol
Type of Project	DeFi, Lending platform for staked and re-staked assets
Audit Timeline	14 days
Review Commit Hash	701925d9a601fdc841ce1981b208286c49fb8f1b
Fixes Review Commit Hash	5bcbd9b12cc8187a7b269c03ea5697d3834ffd20

5.2 Scope

The following smart contracts were in the scope of the security review:

File	nSLOC
src/IonPool.sol	693
src/InterestRate.sol	243
src/flash/handlers/base/UniswapFlashloanBalancerSwapHandler.sol	196
src/reward/RewardModule.sol	192
src/Liquidation.sol	177
src/libraries/uniswap/TickMath.sol	168
src/flash/handlers/base/UniswapFlashswapHandler.sol	145

src/flash/handlers/base/BalancerFlashloanDirectMintHandler.sol	131
src/flash/handlers/base/IonHandlerBase.sol	114
src/flash/handlers/WstEthHandler.sol	103
src/YieldOracle.sol	97
src/oracles/reserve/ReserveOracle.sol	76
src/libraries/math/WadRayMath.sol	72
src/Whitelist.sol	68
src/periphery/IonZapper.sol	57
src/interfaces/ProviderInterfaces.sol	44
src/join/GemJoin.sol	40
src/libraries/StaderLibrary.sol	40
src/flash/handlers/EthXHandler.sol	35
src/oracles/spot/EthXSpotOracle.sol	35
src/admin/TransparentUpgradeableProxy.sol	34
src/oracles/spot/SwEthSpotOracle.sol	34
src/flash/handlers/SwEthHandler.sol	33
src/oracles/spot/SpotOracle.sol	28
src/libraries/uniswap/UniswapOracleLibrary.sol	26
src/oracles/spot/WstEthSpotOracle.sol	26
src/oracles/reserve/EthXReserveOracle.sol	22
src/oracles/reserve/SwEthReserveOracle.sol	21
src/oracles/reserve/WstEthReserveOracle.sol	21
src/admin/ProxyAdmin.sol	20
src/libraries/LidoLibrary.sol	20
src/periphery/IonRegistry.sol	18
src/libraries/SwellLibrary.sol	16
src/oracles/reserve/ReserveFeed.sol	13
src/interfaces/IChainlink.sol	7
src/interfaces/IWETH9.sol	6
src/interfaces/IReserveFeed.sol	5
src/interfaces/IYieldOracle.sol	4
Total	**3080**

6. Findings Summary

The following number of issues have been identified, sorted by their severity:

- **Critical** and **High** issues: **0**
- **Medium** issues: **1**
- **Low** issues: **0**

ID	Title	Severity	Status
[M-01]	Unsafe Downcast Truncation in `UniswapOracleLibrary` Leading to Invalid Price Data	Medium	Resolved

7. Findings

[M-01] Unsafe Downcast Truncation in `UniswapOracleLibrary` Leading to Invalid Price Data

Severity

Medium Risk

Reported By

Mario Poneder

Description

Unsafe downcast from **int56** to **int24** of variable **tickCumulativesDelta** in **UniswapOracleLibrary**. The variable **tickCumulativesDelta** is downcasted **before** being divided by **secondsAgo**.

```
arithmeticMeanTick = int24(tickCumulativesDelta) / int24(int32(secondsAgo
    ));   // @audit cast to 'int24' happens *before* division
```

Impact

In case of **tickCumulativesDelta** exceeding **type(int24).min/max**, the value is silently (no revert) truncated and the caller of the **UniswapOracleLibrary** (e.g. **SwEthSpotOracle**) proceeds with wrong price data.

Proof of Concept (PoC)

For reference, the original implementation from Uniswap:

```
arithmeticMeanTick = int24(tickCumulativesDelta / secondsAgo);  // @audit
    cast to 'int24' happens *after* division
```

Furthermore, according to Uniswap's `TickMath` library:

```
// @dev The minimum tick that may be passed to #getSqrtRatioAtTick
    computed from log base 1.0001 of 2**-128
  int24 internal constant MIN_TICK = -887272;
// @dev The maximum tick that may be passed to #getSqrtRatioAtTick
    computed from log base 1.0001 of 2**128
  int24 internal constant MAX_TICK = -MIN_TICK;
```

Consequently, `type(int24).max / MAX_TICK = 9`. Therefore, in the worst case **int24(tickCumulativesDelta)** could already be truncated after 10 cumulated ticks. Although this specific scenario is unrealistic, it provides us with an estimate of how easily the current implementation can lead to invalid price data.

Location of Affected Code

File: src/libraries/uniswap/UniswapOracleLibrary.sol#L37

```
function consult(
  address pool,
  uint32 secondsAgo
)
  internal
  view
  returns (int24 arithmeticMeanTick, uint128 harmonicMeanLiquidity)
{
// code

// NOTE: Changed to match versions
// arithmeticMeanTick = int24(tickCumulativesDelta / secondsAgo);
  arithmeticMeanTick = int24(tickCumulativesDelta) / int24(int32(
     secondsAgo));

// code
}
```

Recommendation

Downcast **after** division:

```
arithmeticMeanTick = int24(tickCumulativesDelta / int56(int32(secondsAgo)
   ));
```

Team Response

Acknowledged, the downcast truncation is unsafe as it can overflow causing a negative impact on borrow and lender's usage of the protocol. The swETH market in this codebase is no longer in the roadmap to be deployed. So thankfully this vulnerability will not be introduced in production. Regardless we're glad that this issue was discovered

shieldify

Thank you!