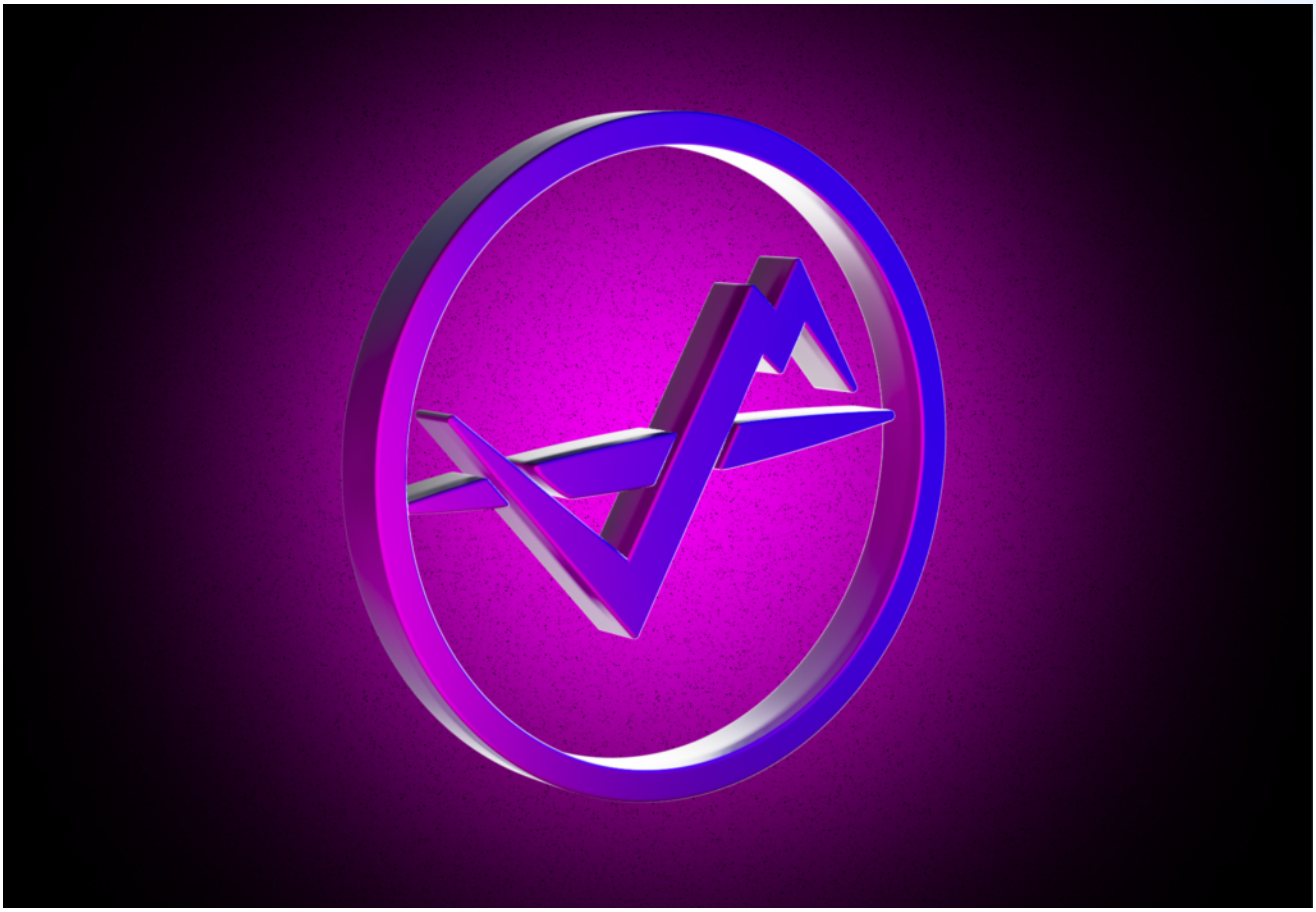
< Back to articles

# IPOR: Protocol Core audit summary

Aleksandra Yudina // AUDITS, ETHEREUM, SOLIDITY, WAKE DECEMBER 18, 2023



IPOR protocol (Inter Protocol Over-block Rate) consists of two parts: the IPOR index and the automated market maker.

The index functions as an interest rate benchmark, aggregating the lending and borrowing interest rates from multiple DeFi lending platforms to provide an objective view of the current market status. The logic lies off-chain. To access the index, a user must call the IPOR Oracle contract.

The second part is the IPOR automated market maker. Unlike the standard AMM, the IPOR AMM allows users to open/close swaps and speculate on interest rates. Swaps can be opened in two different directions:

- To pay a fixed interest rate and receive a floating interest rate
- To pay a floating interest rate and receive a fixed interest rate

The vital role is a liquidity provider. Liquidity providers provide liquidity to the AMM and receive a share of the swap fees. Liquidity in the pool is automatically rebalanced between the protocol treasury and the asset management, which deposits assets into Aave or Compound to earn interest.

**Revision 1.0 & 1.1**

IPOR engaged Ackee Blockchain to perform a security review of the IPOR protocol with a total time donation of 59 engineering days in a period between June 1 and July 28, 2023. Revisions 1.0 and 1.1 were performed during one review.

**Revision 1.2**

IPOR provided an updated codebase with fixes for several reported issues on the 22nd of August, 2023. However, not all the issues were fixed. Together with fixes, new contracts for staked ETH pool were introduced.

**Revision 1.3 & 1.4**

IPOR provided an updated codebase with fixes and a few code changes on the date 18th of September, 2023, the review was done with a time donation of 3 MD. For Revision 1.4 IPOR provided an updated codebase with fixes on the commit `3d99b22`. No new changes were introduced except for updating the values of the spread slope.

**Revision 2.0**

IPOR reached out to Ackee Blockchain to perform an incremental security review of an updated version of the codebase  with a total time donation of 7 engineering days in a period between December 4 and December 11, 2023.

**Revision 2.1**

IPOR provided an updated codebase with fixes for the reported issues. The codebase also contains changes to the stETH demand spread model and additional code refactoring.

# Methodology

**Revision 1.0 & 1.1**

We began our review using static analysis tools Wake. We then took a deep dive into the logic of the contracts and performed a manual code review. In parallel with the manual review, we created comprehensive fuzz tests for the most critical parts of the system, such as opening/closing swaps and providing liquidity. For testing and fuzzing, we have involved Wake testing framework. The test simulates the real-world behavior with a complete deployment of the protocol and with a focus on unexpected call sequences and edge case values.

During the fuzz testing, we found several issues that were reported to the client and immediately fixed. The cooperation with the team and their ability to quickly react was crucial to fuzz testing, where a correctly working protocol is necessary as it allows us to find more issues and edge cases. Fuzz testing at its final stage ran on newer commits (with bug fixes) than the initial one. This commit will be mentioned in Revision 1.1.

During the review, we paid particular attention to:

- ensuring the arithmetic of the system is correct

- detecting possible financial attacks such as flash loans
- ensuring the protocol's behavior stays consistent in edge case scenarios
- checking access control mechanisms
- detecting possible reentrancies in the code
- checking proper storage handling
- comparing the code logic to the documentation
- looking for common issues such as data validation.

**Revision 1.2**

In the first part of the review, we reviewed all the fixes and ensured they corresponded with our recommendations.

In the second part of the review, we manually reviewed the new codebase containing four new contracts and changes in several older contracts. The new codebase allows providing liquidity with Ether, Wrapped Ether, and Staked Ether.

**Revision 1.3**

We began the review with a focus on fixes of previously discovered issues. Then, we focused on the new changes in the codebase. After the manual review, we updated the fuzz tests to be relevant to the new codebase.

**Revision 2.0**

Our review began with updating the fuzz test to be relevant to the new refactored codebase.

After the original fuzz test was updated, we started writing a new fuzz test for the latest part of the protocol managing the Staked ETH. With the new fuzz test, we discovered the two most severe issues reported in this revision.

In parallel with the fuzz testing, we performed a manual review, ran Wake static analysis, and discussed all detections.

During the review we were focusing on the following code changes:

- new risk indicators approach with the parameters being signed and passed directly to the contracts
- new stETH feature — opening and closing swaps
- a different model of spread logic for stETH
- admin-only setter for time-weighted notional data
- the no-closing period after opening a swap
- code refactoring and new architecture compatibility.

# Scope

Revision 1.0: The audit was initially performed on the commit `680c80f`.

Revision 1.1: The review then continued on the last provided commit `87c4d345`

Revision 1.2: The review was performed on the commit `1847f3e6`

Revision 1.3: The review was performed on the commit `553e1c7`

Revision 1.4: The review was performed on an updated codebase with fixes on the commit `3d99b22`

Revision 2.0: The audit was performed on the commit `2c633063`

Revision 2.1: The audit was performed on an updated codebase with fixes for the reported issues on the commit `125b3f3`

# Findings

Here we present our findings.

## Critical severity

C1: Profit & loss accounted twice when unwinding

## High severity

H1: Unwinding formula

H2: Broken reentrancy lock

H3: Unwinding fee accounted twice in liquidityPool balance

## Medium severity

M1: `INTEREST_FROM_STRATEGY_BELOW_ZERO` reverts

M2: Inaccurate hypothetical interest formula

M3: Pool contribution is not updated when liquidity is redeemed

M4: Incorrect event data

M5: Unwinding fee normalization

M6: `IPOR_508` reverts during deposit

M7: Liquidation deposits accounted into LP balance

## Low severity

L1: Value in incorrect decimals

L2: Liquidation deposit accounted twice in rebalancing logic

L3: Aave incorrect APY formula

L4: Close swap and redeem transaction reverts

L5: No data validation while setting `redeemFeeRateEth`

L6: Close swap insufficient balance revert

L7: `IporProtocolRouter` return & revert data dropped

## Warning severity

W1: Usage of `solc` optimizer

W2: `SoapIndicatorRebalanceLogic` underflow

W3: Insufficient data validation in the constructor

W4: Missing array length check in the initialize function

W5: `_calculateRedeemedCollateralRatio` underflow

W6: Constant block production relied on

W7: Github secrets leak

W8: Infinite approval

W9: Missing swap direction validation

W10: Setting array max index in constructor

W11: IporProtocolRouter memory constraints violation

## Informational severity

I1: Unreachable code

I2: Use `type(uint256).max` instead of integer literal

I3: Duplicated code

I4: Redundant require

I5: Using magic numbers

I6: Use `forceApprove` instead of `safeApprove`

I7: User can lose funds if the protocol is used incorrectly

I8: Mixing `_msgSender()` and `msg.sender` across the codebase

I9: Redundant logging of `block.timestamp`

I10: Unused code

# CONCLUSION

Our review resulted in **39 findings across Revision 1.0 – 2.0**, ranging from *Info* to *Critical* severity.

As of Revision 2.0, the new codebase is more readable than the previous ones. However, there are still some parts that could be improved. The NatSpec documentation is part of interface contracts, and it does not cover all the functions and used libraries (RiskIndicatorsValidatorLib, for example). The code contains unused functions, and tests are not written based on the expected behavior but instead based on the known outputs of the code being tested.

**We recommend IPOR to:**

- avoid writing tests based on the outputs of the code being tested
- use static analysis tools (such as Wake) to keep the codebase clean

- update the documentation to reflect the current state of the protocol (concerning the new stETH part, especially).

**Ackee Blockchain's full *IPOR* audit report with a more detailed description of all findings and recommendations can be found here.**

We were delighted to audit **IPOR** and look forward to working with them again.

**Final note**

In the given time donation and after all reported issues were fixed, the auditing team doesn't see any issue that would lead to a loss of funds or any other catastrophic consequences. The confidence of the auditing team is based on a manual review and a fuzz testing model.

The IPOR team sticks with good practices. The code quality is high, the code contains NatSpec documentation, and the general documentation is comprehensive. IPOR team also provided many diagrams and mathematical equations, which made the audit process more effective.

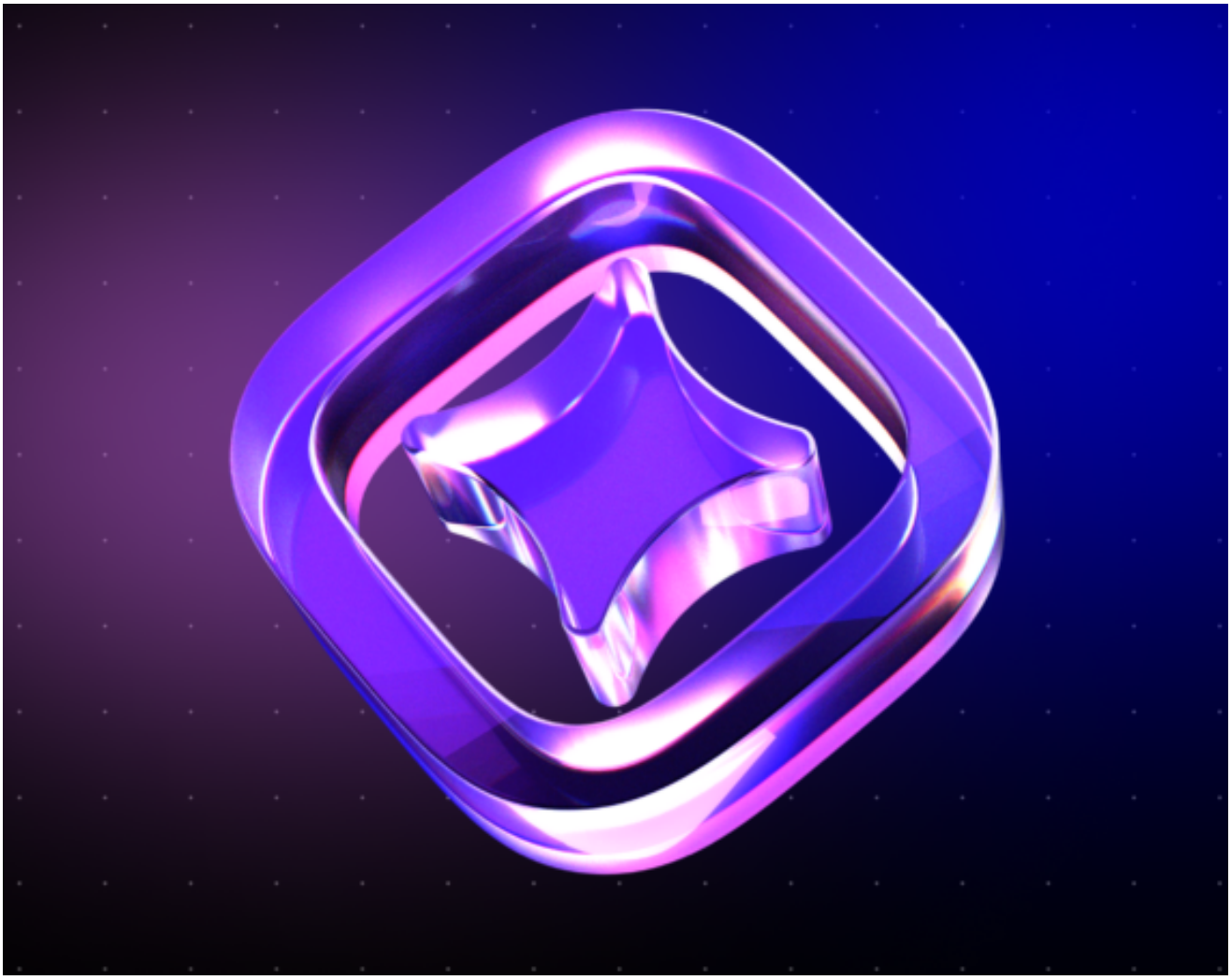
We cannot rule out the chance of DoS of the protocol caused by some edge case conditions. However, it is not directly related to security but rather to the mathematical and architectural complexity of the protocol.

The next step to enhance confidence in the protocol is to extend the fuzz test and model all the parts of the protocol that are not included in the current fuzz test (liquidity mining, governance).

Audit DeFi Ethereum IPOR Solidity Wake

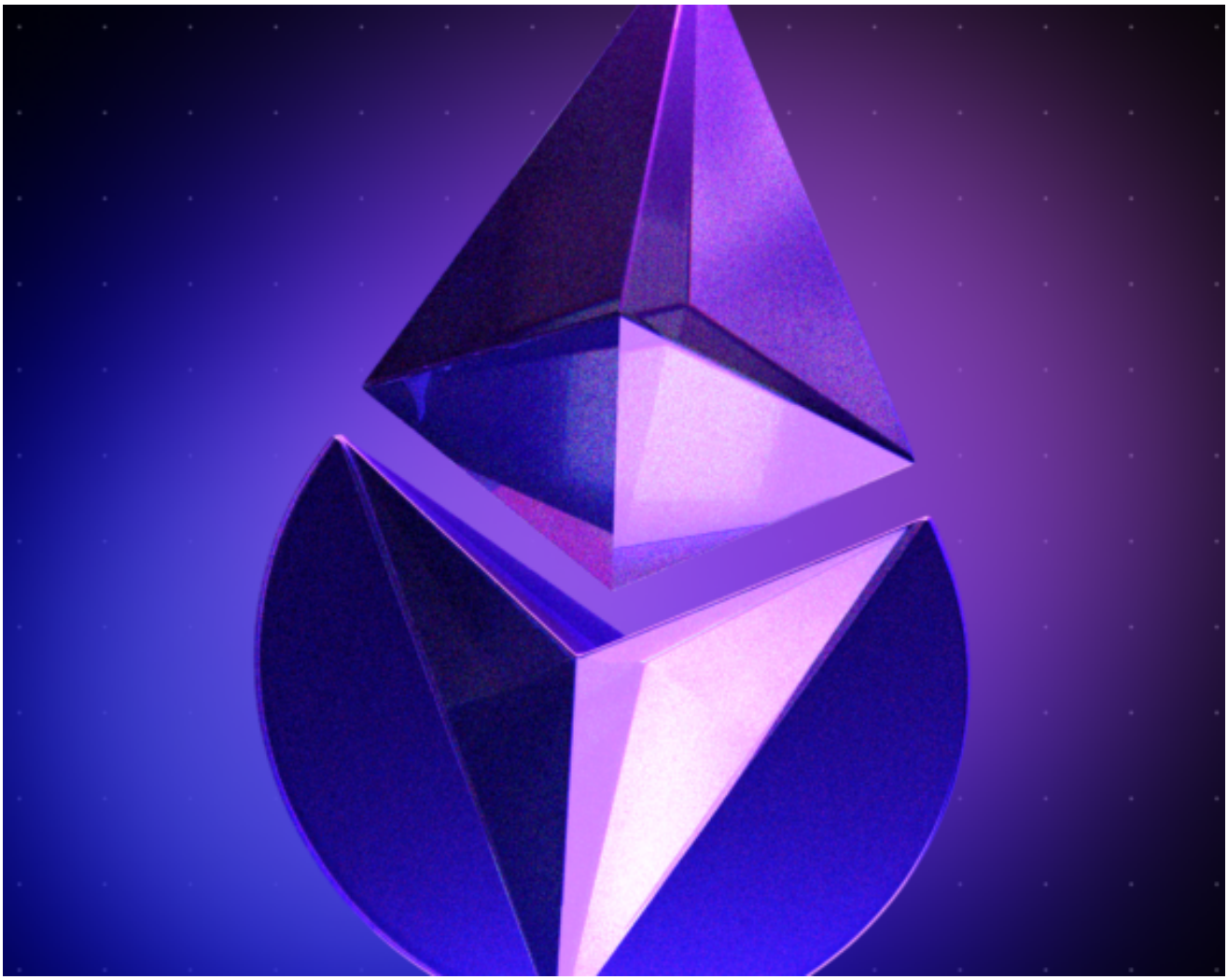
**Aleksandra Yudina**

# You May Also Like



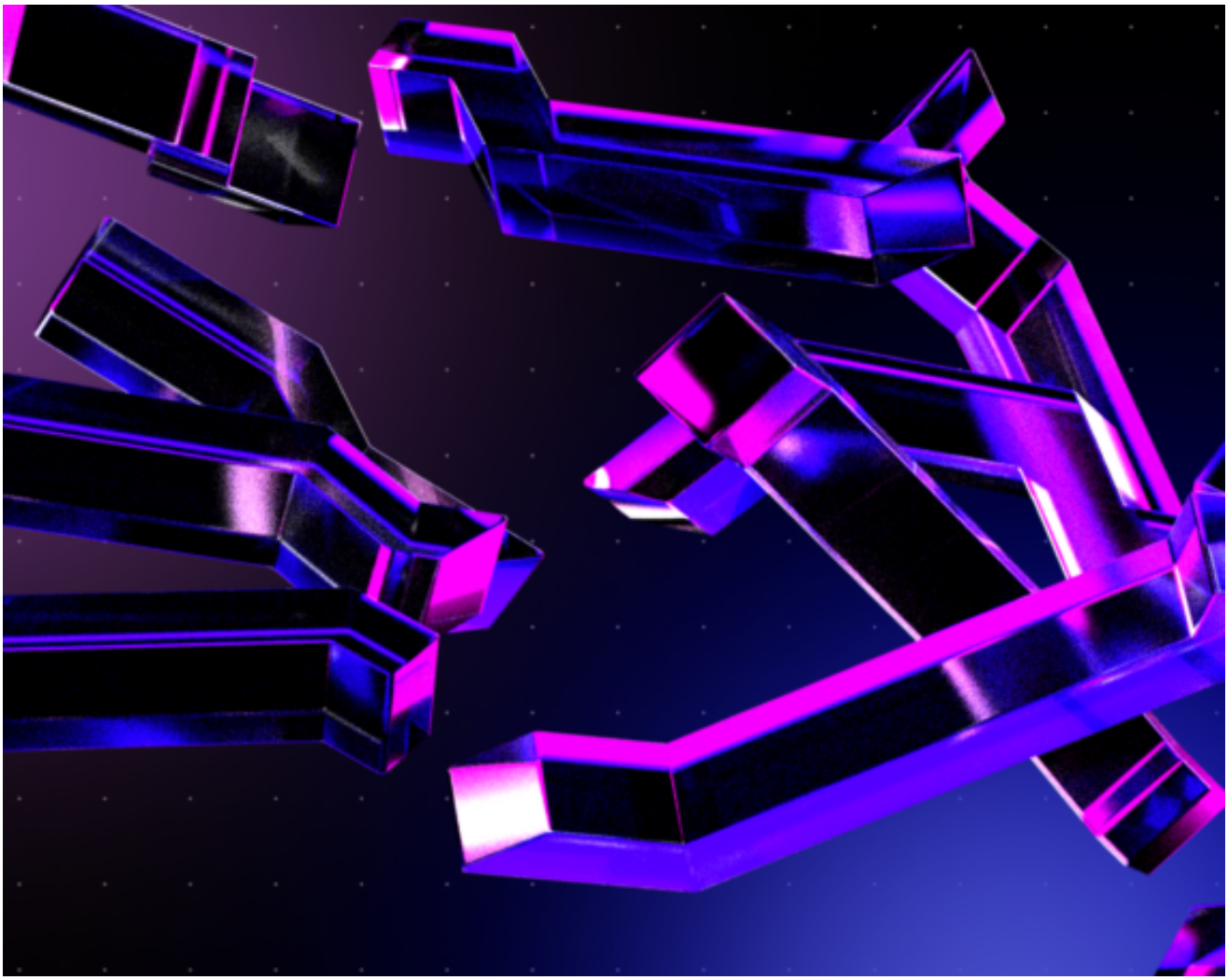
// AUDITS, ETHEREUM, WAKE AUGUST 2, 2024

## Rhinestone Module Registry Audit Summary



// AUDITS ETHEREUM WAKE JULY 29, 2024

# Lido stETH on Optimism Audit Summary



// EDUCATION, ETHEREUM, HACKS, TUTORIAL, WAKE JULY 11, 2024

# Cross Contract Reentrancy Attack