PERIMETER

Origin Protocol WOETH

Offensive Fuzzing Report

Apr. 03, 2025

Prepared By:

Rappie | Lead Fuzzing Specialist
rappie@perimetersec.io

Table of Contents

About Perimeter

Perimeter's mission is to deliver the highest quality fuzzing services to protocols by uniting the world's foremost fuzzing specialists. We possess extensive expertise in fuzzing a diverse range of protocols, from smaller, niche protocols to some of the largest and most complex in DeFi.

In order to deliver on our mission, we have developed the most advanced scaffolding and libraries, enabling us to create highly sophisticated fuzzing suites tailored to meet the unique challenges of each protocol.

Learn more about us at perimetersec.io

Risk Classification

The severity of security issues identified during the security review is classified according to the table below.

- Critical findings are highly likely to be exploited with severe impact on the protocol and require immediate attention.
- High findings are very likely to occur, easy to exploit, or difficult but highly incentivized, and should be resolved as quickly as possible.
- Medium findings are possible in certain circumstances or when incentivized, with a moderate likelihood of occurring, and should be addressed.
- Low findings involve rare circumstances to exploit or offer little to no incentives, though addressing them is still recommended.
- Informational issues represent improvements that do not impact the project's overall security but are worth considering.

Severity Level	High Impact	Medium Impact	Low Impact
High Likelihood	Critical	High	Medium
Medium Likelihood	High	Medium	Low
Low Likelihood	Medium	Low	Low

Services Provided

Perimeter has successfully delivered a comprehensive suite of services that include:

- **Fuzzing Suite Development:** Design and implement a stateful fuzzing suite using Echidna and Medusa. This suite will be tailor-made for the protocol and contracts in scope. The completed fuzzing suite can later be integrated into the testing suite to serve long-term security needs.
- **Findings Reporting:** Provide thorough documentation and reporting of all findings identified throughout the engagement.
- **Invariant Testing Assurance:** Guarantee that each invariant implemented will be tested no fewer than 50,000,000 instances, ensuring thorough validation and reliability.
- **Proof-of-Concept Development:** Develop a corresponding Proof-of-Concept (PoC) for each finding and assertion/property counterexample identified, to demonstrate potential vulnerabilities and their implications.
- **Comprehensive Final Report:** Create a detailed final report that will include all findings, along with their corresponding PoCs. This report will also detail the invariants tested, their run status, and the number of runs, providing a comprehensive overview of the engagement's outcomes.

Files in Scope

The engagement will be focused on the files listed below, acquired from commit [a53a8ceb2acf5a6bf39d971e4163a26a2ff84e3d.]

```
src/token/WOETH.sol
```

Files Out of Scope

Files outside the scope were not directly considered in achieving the target. However, since many of these files are utilized by those within the scope, a significant portion was indirectly covered.

Methodology

The primary goal of this engagement was to implement an offensive fuzzing suite targeting the most vulnerable components of the WOETH codebase. Rather than attempting exhaustive coverage, the objective was to uncover critical vulnerabilities. Additionally, key invariants were defined to validate the correct behavior of WOETH under many different scenarios.

Testing specifically addressed two central questions:

- Assess whether any part of the WOETH implementation could result in the loss of user funds.
- Verify that the yield distribution mechanism operates according to its scheduled intervals.

We replicated WOETH's testing suite deployment, which involves minting a fixed amount of WOETH to a burner address, enhancing the resolution of the exchange rate, and mitigating risks such as ERC-4626 inflation attacks. For this engagement, we assumed this process would be followed during deployment to the mainnet. Adhering to this setup is essential, as deviations could expose the protocol to known vulnerabilities and unreported issues that we considered out of scope.

Due to inherent rounding errors in the WOETH code, certain invariants required setting predefined tolerance levels. Due to time constraints, these tolerances were manually established within acceptable boundaries rather than optimized through Echidna. Subsequent fuzzing tests confirmed these thresholds, verifying that rounding errors consistently remained within expected limits.

Issues Found

At Perimeter, our objective is to thoroughly investigate and uncover critical vulnerabilities through fuzzing and reveal issues often overlooked in manual reviews. Any lower-severity findings are incidental to this core focus.

Severity	Count	Fixed	Partially Resolved	Acknowledged
Critical	0	0	0	0
High	0	0	0	0
Medium	0	0	0	0
Low	0	0	0	0
Informational	0	0	0	0
Total	0	0	0	0

Invariants

We created many tests to verify the correctness of **31** invariants described in the table below. During the execution phase, these invariants were assessed for a total of **4,503,000,000+** calls.

The table below lists all invariants that are part of this engagement.

Invariant	Description	Tested	Passed	# Runs
TASSETS-01	Deposits do not affect total assets except for the OETH transferred by the user	✅	✅	4.5B+
TASSETS-02	Minting does not affect total assets except for the OETH transferred by the user	✅	✅	4.5B+
TASSETS-03	Withdrawing does not affect total assets except for the OETH transferred by the user	✅	✅	4.5B+
TASSETS-04	Redeeming does not affect total assets except for the OETH transferred by the user	✅	✅	4.5B+
TASSETS-05	Transfering does not affect total assets	✅	✅	4.5B+
TASSETS-06	Changing OETH supply does not affect total assets	✅	✅	4.5B+
TASSETS-07	Donating does not change total assets	✅	✅	4.5B+
TASSETS-08	Scheduling yield does not change total assets	✅	✅	4.5B+
SOLV-01	All users can fully redeem their wOETH at any time	✅	✅	4.5B+
SOLV-02	Sum of all users' redeem previews equals total assets (within tolerance)	✅	✅	4.5B+
SOLV-03	Redeem preview strictly increases over time	✅	✅	4.5B+
YIELD-01	All currently undistributed yield is fully distributed at cycle end (within tolerance)	✅	✅	4.5B+
YIELD-02	Exchange rate remains constant within a single block (within tolerance)	✅	✅	4.5B+
YIELD-03	Redeem preview never decreases without interactions with WOETH (within tolerance)	✅	✅	4.5B+
YIELD-04	Redeemable yield within a cycle never exceeds total yield assets distributed during that cycle (within tolerance)	✅	✅	4.5B+

Invariant	Description	Tested	Passed	# Runs
YIELD-05	Redeemable yield never exceeds yield assets unlocked proportionally to elapsed time in the current cycle (within tolerance)	✅	✅	4.5B+
YIELD-06	Current yield assets never exceed locked assets during the previous cycle	✅	✅	4.5B+
PREVIEW-01	Deposit preview equals actual outcome	✅	✅	4.5B+
PREVIEW-02	Mint preview equals actual outcome	✅	✅	4.5B+
PREVIEW-03	Withdraw preview equals actual outcome	✅	✅	4.5B+
PREVIEW-04	Redeem preview equals actual outcome	✅	✅	4.5B+
REVERT-01	Deposit does not unexpectedly revert	✅	✅	4.5B+
REVERT-02	Mint does not unexpectedly revert	✅	✅	4.5B+
REVERT-03	Withdraw does not unexpectedly revert	✅	✅	4.5B+
REVERT-04	Redeem does not unexpectedly revert	✅	✅	4.5B+
REVERT-05	Transfer does not unexpectedly revert	✅	✅	4.5B+
REVERT-06	Changing OETH supply does not unexpectedly revert	✅	✅	4.5B+
REVERT-07	Scheduling yield does not unexpectedly revert	✅	✅	4.5B+
REVERT-08	Function 'totalAssets' never reverts	✅	✅	4.5B+
GLOBAL-01	Total assets never exceed WOETH contract's OETH balance	✅	✅	4.5B+
GLOBAL-02	Tracked assets never exceed WOETH contract's OETH balance	✅	✅	4.5B+

Tolerances

Rounding errors that cause multiple invariants to break were identified during our investigation. These issues exist in the current codebase but remain within acceptable limits.

To address this, tolerances were defined for each affected invariant to ensure they fall within acceptable thresholds. The table below outlines these invariants along with their respective tolerances.

Invariant	Description	Tolerances
SOLV-02	Sum of all users' redeem previews equals total assets (within tolerance)	1 per WOETH holder in the redeem preview sum
YIELD-01	All currently undistributed yield is fully distributed at cycle end (within tolerance)	1 per WOETH holder in the redeem preview sum
YIELD-02	Exchange rate remains constant within a single block (within tolerance)	Previous exchange rate scaled down by 1e14
YIELD-03	Redeem preview never decreases without interactions with WOETH (within tolerance)	1 per WOETH holder in the redeem preview sum
YIELD-04	Redeemable yield within a cycle never exceeds total yield assets distributed during that cycle (within tolerance)	Current redeem preview sum scaled down by 1e14
YIELD-05	Redeemable yield never exceeds yield assets unlocked proportionally to elapsed time in the current cycle (within tolerance)	Current redeem preview sum scaled down by 1e14

Disclaimer

All activities conducted by Perimeter in connection with this project were carried out in accordance with the terms outlined in a Statement of Work and an agreed-upon project plan, as set forth in a proposal document delivered prior to the commencement of the project.

Security assessment projects are subject to time limitations, and as such, the findings presented in this report should not be interpreted as an exhaustive or comprehensive identification of all security issues, vulnerabilities, or defects within the target codebase. Perimeter makes no representations or warranties that the target codebase is free from defects.

Furthermore, this report is not intended to be, and should not be construed as, investment advice or a recommendation to participate in any financial transactions. The content herein does not constitute endorsements or recommendations for any financial decisions, securities, or investment strategies.