Metamask Delegation Framework Audit Report

Prepared by Cyfrin

Version 2.0

Lead Auditors

0kage

May 7, 2025

Contents

1 About Cyfrin

Cyfrin is a Web3 security company dedicated to bringing industry-leading protection and education to our partners and their projects. Our goal is to create a safe, reliable, and transparent environment for everyone in Web3 and DeFi. Learn more about us at cyfrin.io.

2 Disclaimer

The Cyfrin team makes every effort to find as many vulnerabilities in the code as possible in the given time but holds no responsibility for the findings in this document. A security audit by the team does not endorse the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the solidity implementation of the contracts.

3 Risk Classification

	Impact: High	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4 Protocol Summary

5 Audit Scope

The following contracts were included in the scope of this audit:

Enforcers

- LogicalOrWrapperEnforcer.sol
- ERC20BalanceChangeEnforcer.sol
- ERC721BalanceChangeEnforcer.sol
- ERC1155BalanceChangeEnforcer.sol
- NativeBalanceChangeEnforcer.sol

6 Executive Summary

Over the course of 3 days, the Cyfrin team conducted an audit on the Metamask Delegation Framework smart contracts provided by Metamask. In this period, a total of 2 issues were found.

Summary

Project Name	Metamask Delegation Framework
Repository	delegation-framework
Commit	42a2cb4c1d07...
Audit Timeline	Apr 28th - Apr 30th
Methods	Manual Review, Stateful Fuzzing

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	0
Informational	2
Gas Optimizations	0
Total Issues	2

Summary of Findings

[I-1] Execution mode restriction in `LogicalOrWrapperEnforcer` can be removed	Resolved
[I-2] Delegate controlled privilege escalation risk in `LogicalOrWrapperEnforcer`	Resolved

7 Findings

7.1 Informational

7.1.1 Execution mode restriction in `LogicalOrWrapperEnforcer` can be removed

Description: The `LogicalOrWrapperEnforcer` contract currently includes the `onlyDefaultExecutionMode` modifier on all of its hook functions. This creates an unnecessary restriction since the individual enforcers being wrapped already apply their own execution mode restrictions. This design decision could limit the wrapper's flexibility and prevent it from working with caveats that might support non-default execution modes.

Recommended Mitigation: Consider removing the `onlyDefaultExecutionMode` modifier from all hook functions in the `LogicalOrWrapperEnforcer` and let the individual wrapped caveat enforcers handle their own execution mode restrictions.

Not only is this gas efficient, this change would also allow the `LogicalOrWrapperEnforcer` to be more flexible and forward-compatible with future caveats, while still maintaining the appropriate execution mode restrictions through the wrapped enforcer contracts themselves.

Metamask: Fixed in commit d38d53d.

Cyfrin: Resolved.

7.1.2 Delegate controlled privilege escalation risk in `LogicalOrWrapperEnforcer`

Description: When using the `LogicalOrWrapperEnforcer` enforcer, delegators may define multiple caveat groups with different security properties, expecting that all groups provide adequate security boundaries.

However, since delegates control which group is evaluated during execution, they can select the least restrictive group to bypass stricter security requirements defined in other groups.

For example, if a delegator defines two groups:

- Group 0: Requires minimum balance change of 100 tokens
- Group 1: Requires minimum balance change of 50 tokens

A delegate can choose Group 1 at execution time, allowing them to transfer only 50 tokens when the delegator may have expected the 100 token minimum to apply.

While this behavior is by design, it creates a scenario where delegates can bypass intended security restrictions by selecting the least restrictive caveat group. Since this enforcer's behavior gives such control to the delegate, it effectively allows them to elevate their privileges to the least restrictive option available across all defined groups.

Recommended Mitigation: Consider adding a security notice similar to the one added in all balance change enforcers.

```
/**
 * @dev Security Notice: This enforcer allows delegates to choose which caveat group to use at
 ↪  execution time
 * via the groupIndex parameter. If multiple caveat groups are defined with varying levels of
 ↪  restrictions,
 * delegates can select the least restrictive group, bypassing stricter requirements in other groups.
 *
 * To maintain proper security:
 * 1. Ensure each caveat group represents a complete and equally secure permission set
 * 2. Never assume delegates will select the most restrictive group
 * 3. Design caveat groups with the understanding that delegates will choose the path of least
 ↪  resistance
 *
 * Use this enforcer at your own risk and ensure it aligns with your intended security model.
 */
```

Metamask: Fixed in commit d38d53d.

Cyfrin: Resolved.