ChainLight

Hyperlane CCIP Warp Route Security Audit

: Hyperlane PR #5392, #5399, #5394, #5405

Feb 20, 2025

Revision 1.1

ChainLight@Theori

Table of Contents

Hyperlane CCIP Warp Route Security Audit 1
Table of Contents 2
Executive Summary 3
Audit Overview 4
- Scope 4
- Code Revision 4
- Severity Categories 5
- Status Categories 6
- Finding Breakdown by Severity 7

Findings 8
- Summary 8
- #1 HL-250217-001 Message Replay May Lead to Temporary Freeze of Funds (PR #5399) 9
- #2 HL-250217-002 supportsMetadata() Should Be Overridden/Implemented in DefaultHook and AmountRoutingHook (PR #5394 & #5405) 10
- #3 HL-250217-003 Refund May Fail Due to Usage of transfer() (PR #5399) 12
- #4 HL-250217-004 Minor Suggestions 13
- Revision History 14

Executive Summary

Beginning on February 9, 2024, ChainLight of Theori performed a security audit on a set of pull requests related to the CCIP integration of Hyperlane. Our primary concerns involved following issues and potential impacts:

- Theft of funds
- Permanent freeze of funds
- Denial of service

As a result, we identified issues as listed below.

- Total: 4
- High: 1 (Message replay leading to temporary freeze of funds)
- Low: 1 (Functionality issue with smart contract wallets as refund address)
- Informational: 2

Audit Overview

Scope

Name	Hyperlane CCIP Warp Route Security Audit
Target / Version	• Git Repository (hyperlane-xyz/hyperlane-monorepo): PR 5392, 5399, 5394, 5405
Application Type	Smart contracts
Lang. / Platforms	Smart contracts [Solidity]

Code Revision

N/A

Severity Categories

Severity	Description
Critical	The attack cost is low (not requiring much time or effort to succeed in the actual attack), and the vulnerability causes a high-impact issue. (e.g., Effect on service availability, Attacker taking financial gain)
High	An attacker can succeed in an attack which clearly causes problems in the service's operation. Even when the attack cost is high, the severity of the issue is considered "high" if the impact of the attack is remarkably high.
Medium	An attacker may perform an unintended action in the service, and the action may impact service operation. However, there are some restrictions for the actual attack to succeed.
Low	An attacker can perform an unintended action in the service, but the action does not cause significant impact or the success rate of the attack is remarkably low.
Informational	Any informational findings that do not directly impact the user or the protocol.
Note	Neutral information about the target that is not directly related to the project's safety and security.

Status Categories

Status	Description
Reported	ChainLight reported the issue to the client.
WIP	The client is working on the patch.
Patched	The client fully resolved the issue by patching the root cause.
Mitigated	The client resolved the issue by reducing the risk to an acceptable level by introducing mitigations.
Acknowledged	The client acknowledged the potential risk, but they will resolve it later.
Won't Fix	The client acknowledged the potential risk, but they decided to accept the risk.

Finding Breakdown by Severity

Category	Count	Findings
Critical	0	• N/A
High	1	• `HL-250217-001`
Medium	0	• N/A
Low	1	• `HL-250217-003`
Informational	2	• `HL-250217-002` • `HL-250217-004`
Note	0	• N/A

Findings

Summary

#	ID	Title	Severity	Status
1	`HL-250217-001`	Message Replay May Lead to Temporary Freeze of Funds (PR #5399)	High	Acknowledged
2	`HL-250217-002`	`supportsMetadata()` Should Be Overridden/Implemented in `DefaultHook` and `AmountRoutingHook` (PR #5394 & #5405)	Informational	Won't Fix
3	`HL-250217-003`	Refund May Fail Due to Usage of `transfer()` (PR #5399)	Low	Patched
4	`HL-250217-004`	Minor Suggestions	Informational	Patched

#1 `HL-250217-001` Message Replay May Lead to Temporary Freeze of Funds (PR #5399)

ID	Summary	Severity
`HL-250217-001`	Attackers can resend a previously dispatched message, potentially causing the transferred funds in that message to be temporarily frozen.	High

Description

An attacker can invoke `postDispatch()` with the most recent `message.id`, leading to the legitimate message being recognized as already used. In non-strict-order flows that allow transferring `msg.value`, a replayed message processed first may invalidate the legitimate message and freeze the associated funds until manual recovery. In hooks that do not support `msg.value` transfers (e.g., `CCIPHook`), there is no effect.

Impact

High

Funds of the affected message can be temporarily frozen if the replayed message is processed before the legitimate one.

Recommendation

Include a check in `AbstractMessageIdAuthHook._postDispatch()` to revert if a `messageId` has already been dispatched, similar to `validateMessageOnce` in `RateLimitedHook`. Alternatively, restrict `postDispatch()` to be callable only by the mailbox.

Remediation

Acknowledged

A fix is planned.

#2 HL-250217-002 supportsMetadata() Should Be Overridden/Implemented in DefaultHook and AmountRoutingHook (PR #5394 & #5405)

ID	Summary	Severity
HL-250217-002	DefaultHook and AmountRoutingHook inherit supportsMetadata(), potentially causing inconsistent behavior with child hooks.	Informational

Description

Since DefaultHook and AmountRoutingHook inherit supportsMetadata() from AbstractPostDispatchHook, their supportsMetadata() may report incorrect results if their child hooks have metadata encoding incompatible with AbstractPostDispatchHook.

Impact

Informational

Affected contracts may report incorrect metadata support if their child hooks have incompatible metadata encoding.

Recommendation

Override/Implement supportsMetadata() in both contracts:

DefaultHook.sol:

```
function supportsMetadata(bytes calldata metadata, bytes calldata message)
    public
    override
    returns (bool)
{
    return _hook().supportsMetadata(metadata, message);
}
```

and in `AmountRoutingHook.sol`:

```
function supportsMetadata(bytes calldata metadata, bytes calldata message)
    public
    returns (bool)
{
    return IPostDispatchHook(_partition(message))
        .supportsMetadata(metadata, message);
}
```

Remediation

Won't Fix

Enforcing compatible metadata encoding is deferred to child hooks. A hook that does not decode the metadata is expected to simply return `true` to avoid unnecessary call tree cost.

#3 `HL-250217-003` Refund May Fail Due to Usage of `transfer()` (PR #5399)

ID	Summary	Severity
`HL-250217-003`	A refactoring switched from `sendValue()` to `transfer()`, imposing a strict 2,300 gas limit that may cause refunds to contract addresses to fail.	Low

Description

Previously, `sendValue()` allowed forwarding additional gas so contract-based recipients with more complex fallback functions could handle refunds. Switching to `transfer()` enforces a low gas stipend, leading to reverts if the recipient contract requires more gas (e.g., multisig or AA wallets).

Impact

Low

- Refunds to contract addresses may fail due to insufficient gas.
- Only externally owned accounts (EOAs) reliably succeed with `transfer()`.

Recommendation

Revert to a `call`-based approach such as `sendValue()` to ensure enough gas for contract-based recipients.

Remediation

Patched

The issue has been resolved as recommended.

#4 HL-250217-004 Minor Suggestions

ID	Summary	Severity
HL-250217-004	The description includes multiple suggestions for preventing incorrect settings caused by operational mistakes, mitigating potential issues, and improving code maturity and readability.	Informational

Description

1. If `CCIPIsm.preVerifyMessage()` is called directly from the router instead of through `_ccipReceive()`, the validations of `ccipOrigin` and `sender` performed in `_ccipReceive()` might be bypassed. Currently, the router can only call the receiver's `ccipReceive()`, so this is not an immediate issue. However, it is recommended to include all message validations in `_isAuthorized()` rather than `_ccipReceive()`.
2. In `CCIPIsm._ccipReceive()`, it is recommended to use `0` instead of `msg.value` when calling `preVerifyMessage()`.
3. In `CCIPHook._buildCCIPMessage()`, if `extraArgs` is empty (`""`), `allowOutOfOrderExecution` defaults to `false`, enforcing strict message ordering. If out-of-order execution is acceptable, consider setting `allowOutOfOrderExecution` to `true`.

Impact

Informational

Recommendation

Consider applying the suggestions in the description above.

Remediation

Patched

- For Item 1, the team continues to rely on `CCIPReceiver` to enforce `msg.sender` on the `ccipReceive` call and `AbstractMessageIdAuthorizedIsm` to enforce `msg.sender` on the `preVerify` call. Also, there is no access to the CCIP message data in `isAuthorized()`.
- Items 2 and 3 have been resolved.

Revision History

Version	Date	Description
1.0	Feb 17, 2025	Initial version
1.1	Feb 20, 2025	Corrected impact of HL-250217-001 issue

Theori, Inc. (“We”) is acting solely for the client and is not responsible to any other party. Deliverables are valid for and should be used solely in connection with the purpose for which they were prepared as set out in our engagement agreement. You should not refer to or use our name or advice for any other purpose. The information (where appropriate) has not been verified. No representation or warranty is given as to accuracy, completeness or correctness of information in the Deliverables, any document, or any other information made available. Deliverables are for the internal use of the client and may not be used or relied upon by any person or entity other than the client. Deliverables are confidential and are not to be provided, without our authorization (preferably written), to entities or representatives of entities (including employees) that are not the client, including affiliates or representatives of affiliates of the client.