Hyperlane Q2 2025 Security Audit

: Hyperlane MCWR, ICA, CCTP, OP Audit

June 26, 2025

Revision 1.0

ChainLight@Theori

Table of Contents

Executive Summary

From **16 May 2025**, ChainLight (Theori) conducted a security audit of Hyperlane's `audit-q2-2025` branch, which introduced smart-contract changes. The engagement focused on uncovering critical vulnerabilities and assessing their potential impact.

Summary of Findings

- **Critical:** 2
- **Informational:** 5

Audit Overview

Scope

Name	Hyperlane Q2 2025 Security Audit
Target / Version	• Git Repository (`hyperlane-xyz/hyperlane-monorepo`): `audit-q2-2025` branch (commit `10fca83f7c30b895de891df1083c3f6b45ddb1e9`, commit after patches `23cbbb325a4004a701c6f979d52cf0329c3de5dd`), PR `#6276` (commit `117fe8d20c764fbec99331e8c8cce6b5dafcdd48`)
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	2	• `HYPERLANE-2506-004` • `HYPERLANE-2506-005`
High	0	• N/A
Medium	0	• N/A
Low	0	• N/A
Informational	5	• `HYPERLANE-2506-001` • `HYPERLANE-2506-002` • `HYPERLANE-2506-003` • `HYPERLANE-2506-006` • `HYPERLANE-2506-007`
Note	0	• N/A

Findings

Summary

#	ID	Title	Severity	Status
1	`HYPERLANE-2506-001`	Potential Inaccessibility of Existing Interchain Accounts (ICAs) Due to Address Derivation Changes in `InterchainAccountRouter`	Informational	Won't Fix
2	`HYPERLANE-2506-002`	Interchain Gas Payment (IGP) Cost Estimation's Edge Cases in `InterchainAccountRouter` Quoting Functions	Informational	Won't Fix
3	`HYPERLANE-2506-003`	`msg.value` Handling in `callRemoteCommitReveal` May Lead to Reveal Phase Failure	Informational	Won't Fix
4	`HYPERLANE-2506-004`	`TokenBridge` Susceptible to Ownership Takeover Due to Parent `initialize` Function Left Callable	Critical	Patched
5	`HYPERLANE-2506-005`	ISM `verify` Implementations Lack Unique Binding Between Metadata and Message	Critical	Patched
6	`HYPERLANE-2506-006`	Considerations for MEV Risk in the Commit-Reveal Execution Path	Informational	Won't Fix
7	`HYPERLANE-2506-007`	Minor Suggestions	Informational	Patched

#1 `HYPERLANE-2506-001` Potential Inaccessibility of Existing Interchain Accounts (ICAs) Due to Address Derivation Changes in `InterchainAccountRouter`

ID	Summary	Severity
`HYPERLANE-2506-001`	Recent changes to the `InterchainAccountRouter` may alter the derivation of the deterministic address of Interchain Accounts. This could render ICAs deployed with previous router versions inaccessible when interacting through a new router version.	Informational

Description

The `InterchainAccountRouter` contract uses the `CREATE2` opcode to deterministically deploy `OwnableMulticall` proxy contracts, which serve as Interchain Accounts (ICAs). The address of an ICA is derived from a salt and the bytecode hash of the proxy.

Recent modifications to the `InterchainAccountRouter` may affect this derivation logic in two ways:

1. **Salt Calculation:** The internal `_getSalt` function now incorporates a `_userSalt` parameter into its `keccak256(abi.encodePacked(...))` calculation. Suppose previous versions of the router calculated the salt without this `_userSalt` (or with a different packing of elements). In that case, the salt generated by the new version will differ, even if `_userSalt` is defaulted to `InterchainAccountMessage.EMPTY_SALT` (`bytes32(0)`), because the arguments to `abi.encodePacked` have changed.
2. **Bytecode Hash:** The `bytecodeHash` used for deploying ICA proxies is `_proxyBytecodeHash(implementation)`, which is `keccak256(MinimalProxy.bytecode(implementation))`. The `implementation` is the `OwnableMulticall` master copy deployed by the `InterchainAccountRouter`'s constructor using `_implementationBytecode(address(this))`. If the `OwnableMulticall` contract's `creationCode` or the `MinimalProxy.bytecode()` logic

has changed between router versions, the `implementation` address and subsequently the `bytecodeHash` for ICA proxies would also change.

Either of these changes would result in newly derived/deployed ICA addresses being different from those derived by previous versions of the `InterchainAccountRouter`, even if the same logical inputs (origin domain, owner, remote router address, remote ISM address, and a default/empty user salt) are used.

Impact

Informational (The severity has been lowered from High to Informational due to the clarification.)

Users attempting to interact with or predict addresses of ICAs created by an older router version via a newly deployed router version may lose access, and any assets or states managed by those pre-existing ICAs could become inaccessible through the new router if it cannot derive the old addresses.

Recommendation

To ensure backward compatibility, it is recommended to implement a set of 'legacy' functions within the new `InterchainAccountRouter`. These functions should replicate the exact address derivation logic of the previous router version, using the original salt calculation method and the bytecode hash of the prior `OwnableMulticall` implementation.

(If the new `InterchainAccountRouter` is deployed to a separate address rather than replacing the implementation of the existing proxy, existing users will still be able to access their previous ICAs through the original `InterchainAccountRouter`.)

Remediation

Won't Fix

The client has clarified that the new `InterchainAccountRouter` will be deployed at a new address, rather than upgrading the existing one. The old router deployment will be deprecated but will remain functional for legacy users.

#2 `HYPERLANE-2506-002` Interchain Gas Payment (IGP) Cost Estimation's Edge Cases in `InterchainAccountRouter` Quoting Functions

ID	Summary	Severity
`HYPERLANE-2506-002`	The IGP quoting functions in `InterchainAccountRouter` may produce inaccurate gas estimates by not accounting for custom hooks and by using a simplified message body for calculation. This can lead to transaction failures due to underfunding or cause users to overpay for interchain gas.	Informational

Description

The `quoteGasPayment` and `quoteGasForCommitReveal` functions in the `InterchainAccountRouter` may provide inaccurate gas cost estimations due to two primary factors:

1. **Assumption of Default Hook:** The quoting functions consistently use the router's own configured hook (`address(hook)`) for estimation (`_Router_quoteDispatch`). However, the actual dispatch functions (`callRemoteWithOverrides` , `callRemoteCommitReveal`) permit users to specify a custom `IPostDispatchHook` . If this custom hook has a different gas consumption profile, the estimate will be inaccurate.
2. **Simplified Message Body:** The functions use an empty `bytes` array (`new bytes(0)` or `bytes("")`) as a placeholder for the `_messageBody` during quoting. Actual `InterchainAccountMessage` bodies contain structured data (such as owner, ISM, salt, calls, or commitment hashes), and if the underlying `Mailbox.quoteDispatch` or the hook's gas calculation depends on the message body's size or content, using an empty body will lead to an underestimation of the true IGP cost.

Impact

Informational (The severity has been lowered from Medium to Informational due to the clarification.)

Unreliable IGP quotes can lead to a poor user experience. Underestimating the cost may result in transaction failures on the destination chain, while overestimating can cause users to pay more than necessary for interchain gas payments.

Recommendation

To improve accuracy, the IGP quoting functions should be enhanced.

1. The `quoteGasPayment` and `quoteGasForCommitReveal` functions should be modified to accept an optional `IPostDispatchHook _customHook` parameter, using it for estimation if provided.
2. The `quoteGasForCommitReveal` function should be further refined to construct realistic mock message bodies (based on expected content and size parameters) for both the commit and reveal phases to generate a more precise total IGP quote.

Remediation

Won't Fix

The client has stated that the quoting functions are provided as a convenience for applications using the default configuration. It is expected that users leveraging custom hooks with dynamic gas behavior will need to implement a custom integration on the client side to ensure accurate gas estimation.

#3 `HYPERLANE-2506-003` `msg.value` Handling in `callRemoteCommitReveal` May Lead to Reveal Phase Failure

ID	Summary	Severity
`HYPERLANE-2506-003`	The `callRemoteCommitReveal` function's "commit-first, use-refund-for-reveal" funding strategy is fragile. Its success depends on the commit hook correctly refunding any surplus `msg.value` and on the user providing an accurate initial payment.	Informational

Description

The `InterchainAccountRouter.callRemoteCommitReveal` function funds its two message dispatches by sending `msg.value` for both messages with the initial commit message. The `StandardHookMetadata` is configured to refund any IGP overpayment to the `InterchainAccountRouter` contract itself. This refund is then intended to fund the subsequent reveal message.

This mechanism relies on the following two conditions:

1. **Hook Behavior:** The process relies on the `IPostDispatchHook` used for the commit message to refund any surplus `msg.value`. If the hook consumes a substantial portion of it or forwards the refund to a different address, the router will have insufficient funds for the reveal dispatch (e.g., `CCIPHook`).
2. **`msg.value` Accuracy:** The user must provide an initial `msg.value` that precisely covers the total IGP costs for both the commit and reveal dispatches. This includes the specific gas costs of the hooks used in each phase and potential variations in destination chain gas prices, which might not be perfectly captured by quoting functions (especially if they suffer from issues like those in `HYPERLANE-2506-002`). If the commit phase consumes more gas than anticipated (due to hook costs or higher gas prices), the refund to the router will be smaller than needed for the reveal.

Impact

Informational (The severity has been lowered from Medium to Informational due to the clarification.)

If the router's balance is insufficient after the commit dispatch, the reveal message will be dispatched with insufficient IGP. This will cause the reveal to fail on the destination chain, leading to the failure of the entire commit-reveal sequence.

Recommendation

It is recommended that users calculate the total required `msg.value` off-chain by leveraging an improved `quoteGasForCommitReveal()` function. As suggested in `HYPERLANE-2506-002`, this function should be capable of individually estimating and summing the IGP for both the commit and reveal messages, accounting for the specific hooks and realistic message sizes for both phases.

(The on-chain part does not need to be updated, but it remains heavily reliant on this accurate upfront payment and the correct refund behavior of the commit-phase hook.)

Remediation

Won't Fix

As clarified in the response for `HYPERLANE-2506-002`, considering only the default configuration, the `quoteGasForCommitReveal` endpoint accurately reflects the cost of dispatching both messages. Also, if a user provides insufficient `gasLimit`, it is possible that the reveal message will not be delivered, but this behavior is considered consistent with existing `callRemote` semantics.

#4 `HYPERLANE-2506-004` `TokenBridge` Susceptible to Ownership Takeover Due to Parent `initialize` Function Left Callable

ID	Summary	Severity
`HYPERLANE-2506-004`	The `TokenBridge` contract does not properly disable the `initialize` function of its parent, `HypERC20Collateral`. This allows an attacker to reinitialize the contract and transfer ownership of the `TokenBridge` contract to themselves.	**Critical**

Description

The `TokenBridge` contract defines its own `initialize()` function but does so by overloading, not overriding, the function of the same name in its parent contract, `HypERC20Collateral`. The parent's `initialize()` function includes the `reinitializer(2)` modifier, which means it can be called even after other initialization functions have been executed.

Because the parent's initializer is not disabled, an attacker can call `HypERC20Collateral.initialize()` on an already-deployed `TokenBridge` instance unless `HypERC20Collateral.initialize()` is called along with `TokenBridge.initialize()` during deployment. This would re-trigger the ownership transfer logic within the parent contract, allowing the attacker to set themselves as the new owner.

Impact

Critical

An attacker who gains ownership can freely change the contract's configuration, which may lead to the theft of funds.

Recommendation

The `TokenBridge` contract must explicitly override its parent's `initialize()` function. The overriding function should either be empty or contain a `revert()` statement and must not include the `reinitializer()` modifier. This will effectively disable the parent initializer and prevent it from being called on the `TokenBridge` contract.

Remediation

Patched

It has been patched by using `initializer` modifier in `HypERC20Collateral.initialize()` instead of `reinitializer` modifier.

#5 `HYPERLANE-2506-005` ISM `verify` Implementations Lack Unique Binding Between Metadata and Message

ID	Summary	Severity
`HYPERLANE-2506-005`	Several Interchain Security Module (ISM) contracts contain `verify` functions that do not cryptographically check if the provided metadata is uniquely bound to the specific message being processed. This flaw could allow an attacker to forge messages by replaying valid metadata with a malicious message.	**Critical**

Description

The `Mailbox.process(bytes calldata _metadata, bytes calldata _message)` function calls `ism.verify(_metadata, _message)` to authenticate incoming messages. However, an investigation of several ISM implementations (`TokenBridgeCctp`, `CommitmentReadIsm`, `OPL2ToL1CcipReadIsm`, `OpL1NativeTokenBridge`) reveals that their respective `verify()` functions do not ensure that the `_metadata` is bound to the `_message`.

This creates a vulnerability where, as long as valid metadata is provided, any arbitrary message can pass the check. An attacker could capture valid metadata from a legitimate transaction and reuse it with a manipulated message directed at a recipient that uses one of the vulnerable ISMs.

Impact

Critical

An attacker can supply previously approved metadata with a forged message to trigger an arbitrary `recipient.handle()` call. Depending on the recipient's behavior, this could lead to a range of exploits, including theft of funds or other unexpected state changes.

Recommendation

The `verify()` functions in the affected ISM contracts must be updated to validate that the metadata is dependent on the message. This can be achieved by ensuring that a unique identifier from the message, such as `message.id()`, is cryptographically included in and verified against the metadata, similar to `ArbL2ToL1ISM._verifyWithOutboxCall()`. To prevent variants, it is recommended to define a standard interface for this verification that all ISMs inheriting from a common parent must implement.

Remediation

Patched

All affected ISMs have been patched (except the `CommitmentReadIsm`, which turned out to have the required validations).

#6 `HYPERLANE-2506-006` Considerations for MEV Risk in the Commit-Reveal Execution Path

ID	Summary	Severity
`HYPERLANE-2506-006`	The commit-reveal scheme, intended to mitigate Miner Extractable Value (MEV), can be bypassed under certain conditions.	Informational

Description

Two MEV-related risks have been identified in the commit-reveal execution path:

1. **Front-running `revealAndExecute()`:** The `revealAndExecute()` function on the `OwnableMulticall` contract is `public`, meaning anyone can invoke it directly. An attacker could monitor the chain for a legitimate `REVEAL` message and front-run it by calling `revealAndExecute()` first. While this might execute the user's intended action, the legitimate message will subsequently fail upon processing, leaving it in an unresolved state in the Hyperlane explorer and causing confusion for the user. Note that this also allows a bypass of `CommitmentReadISM.verify()`, although that does not lead to an exploitable condition as of now.

2. **Mempool Exposure:** When a relayer submits a `REVEAL` message via `Mailbox.process()`, the transaction may be visible in the public mempool. This allows MEV searchers to detect the transaction and its payload, enabling them to perform attacks like sandwiching if the transaction involves a token swap or other profitable action.

Impact

Informational (The severity has been lowered from High to Informational due to the clarification.)

Front-running attacks can disrupt the standard message-passing flow, causing legitimate transactions to fail and creating a confusing user experience. Sandwich attacks can lead to direct financial loss for users by providing them with a worse execution price on swaps and other operations.

Recommendation

To better mitigate these MEV risks, the following measures are recommended:

1. Restrict access to the `revealAndExecute()` function so that it can only be invoked by its owner (the `ICARouter`). The logic should be adjusted so `ICARouter.handle()` is responsible for the call.
2. Only designated relayers should be allowed to execute messages, and they should utilize private transaction infrastructure, such as Flashbots, to submit `REVEAL` transactions without publicly broadcasting them.

Remediation

Won't Fix

The client clarified that Case 1 is unlikely since the off-chain calldata service authenticates a relayer. For Case 2, the design assumes the availability of private transaction infrastructure, and applications should layer additional MEV mitigation if this is not the case.

#7 `HYPERLANE-2506-007` Minor Suggestions

ID	Summary	Severity
`HYPERLANE-2506-007`	The description includes multiple suggestions for preventing incorrect settings caused by operational mistakes, mitigating potential issues, and improving code maturity and readability.	Informational

Description

- In `InterchainAccountRouter`'s `_dispatchMessageWithValue()`, for the `_router != InterchainAccountMessage.EMPTY_SALT` check, it is suggested to use `bytes32(0)` directly to clarify the meaning of `EMPTY_SALT`, or to define and use an explicit constant like `EMPTY_ROUTER_BYTES32` in `Router` or a common library.
- With the introduction of the `InterchainAccountMessage`'s `MessageType` enum (`CALLS`, `COMMITMENT`, `REVEAL`) and changes to the message encoding method, the existing message format description in the comments is no longer accurate. Also, update the comments regarding the message structure of the `InterchainAccountMessageReveal` library (`[1 byte: messageType.Reveal][32 bytes: ism][32 bytes: commitment]`).
- In the constructor of `OPL2ToL1CcipReadIsm`, check if the `_opPortal` address is a contract.
- An event should be added to `OwnableMulticall`'s `setCommitment()`.
- In the constructor of `OwnableMulticall`, check if the `_owner` argument is a zero address.
- In `AbstractCcipReadIsm`'s `setUrls()`, check if the length of the argument `_urls` is greater than 0.
- Consider renaming functions in `InterchainAccountMessageReveal` (e.g., `ism()` to `ismReveal()` and `commitment()` to `commitmentReveal()`) to avoid confusion with similarly named functions in `InterchainAccountMessage` for improved code clarity.

Impact

Informational

Recommendation

Consider applying the suggestions in the description above.

Remediation

Patched

It has been patched as recommended.

Revision History

Version	Date	Description
1.0	June 26, 2025	Initial version

ChainLight