M Portal Lite Security Audit

: M0 Protocol - M Portal Lite

May 25, 2025

Revision 1.0

ChainLight@Theori

Table of Contents

Executive Summary

From April 28, 2025, ChainLight conducted a security audit of M0 Protocol's M Portal Lite smart contracts for three days. The engagement focused on uncovering critical vulnerabilities and assessing their potential impact.

Summary of Findings

- **Critical:** 1
- **Low:** 1
- **Informational:** 1

Audit Overview

Scope

Name	M Portal Lite Security Audit
Target / Version	• Git Repository (`m0-foundation/m-portal-lite`): commit `099ea8ee2b75c789866dd095a6362273e44263a4`
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	1	• `MPORTALLITE-001`
High	0	• N/A
Medium	0	• N/A
Low	1	• `MPORTALLITE-003`
Informational	1	• `MPORTALLITE-002`
Note	0	• N/A

Findings

Summary

#	ID	Title	Severity	Status
1	`MPORTALLITE-001`	Missing Sender Validation in `HyperlaneBridge.handle()` Leading to Fund Theft via Arbitrary Message Forgery	Critical	Patched
2	`MPORTALLITE-002`	Rounding Down in `HubPortal._mintOrUnlock()` Weakening `totalBridgedPrincipal` Safeguard	Informational	Won't Fix
3	`MPORTALLITE-003`	Integer Truncation in `HubPortal._mintOrUnlock()` Leading to Potential `bridgedPrincipal` Check Bypass	Low	Won't Fix

#1 `MPORTALLITE-001` Missing Sender Validation in `HyperlaneBridge.handle()` Leading to Fund Theft via Arbitrary Message Forgery

ID	Summary	Severity
`MPORTALLITE-001`	The `HyperlaneBridge.handle()` function lacks sender validation. This can lead to minting unbacked tokens on SpokePortals or theft of all locked tokens from the `HubPortal`.	**Critical**

Description

The `handle()` function in the `HyperlaneBridge` contract forwards incoming messages from other chains to portal contracts. It is intended to be only accept messages from bridge contracts on the other chains. However, it does not check if the `sender_` parameter, indicating the sender on the source chain, is a legitimate, known bridge contract. This allows an attacker to send the message from any address, delivering messages to portal contracts while impersonating a bridge contract on the other chains.

Impact

Critical

This vulnerability allows for the complete forgery of inter-chain messages, leading to severe consequences:

- **Theft of Funds from `HubPortal`:** An attacker can craft messages instructing the `HubPortal` to release all locked tokens, effectively draining the bridge.
- **Minting of Unbacked Tokens on SpokePortals:** An attacker can instruct a `SpokePortal` to mint an excessive amount of tokens that are not collateralized by assets in the `HubPortal`.

Recommendation

The `HyperlaneBridge.handle()` function must verify that the `sender_` corresponds to the valid bridge contract address for the given `sourceChainId_`. (e.g., `require(sender_ == _getPeer(sourceChainId_));`)

Remediation

Patched

It has been patched as recommended.

#2 MPORTALLITE-002 Rounding Down in HubPortal._mintOrUnlock() Weakening totalBridgedPrincipal Safeguard

ID	Summary	Severity
MPORTALLITE-002	The HubPortal._mintOrUnlock() function calculates principalAmount by rounding down, which could subtly weaken the totalBridgedPrincipal safeguard.	Informational

Description

The _mintOrUnlock() function in the HubPortal contract calculates the principalAmount using IndexingMath.getPrincipalAmountRoundedDown(uint240(amount_), _currentIndex()). A safeguard, the totalBridgedPrincipal check, limits the amount bridged out from any SpokePortal to the amount bridged in. Due to the rounding down in getPrincipalAmountRoundedDown, a sufficiently large _currentIndex() can cause the calculated principalAmount to be marginally less than its true proportional value. Consequently, multiple small unlock operations might cumulatively permit a total unlocked principal exceeding the intended strict limit of the totalBridgedPrincipal check.

Impact

Informational

Meaningful exploitation requires specific conditions: an attacker who can craft messages from a single chain by either compromising the bridge or a chain, a sufficiently large _currentIndex(), and transaction costs lower than the attacker's per-transaction gain (itself limited by _currentIndex()).

Recommendation

To maintain the intended strictness of the totalBridgedPrincipal check, consider modifying the principalAmount calculation in _mintOrUnlock() to round up. This approach would

ensure the principal debited is never less than its proportional value, thereby preserving the integrity of the `totalBridgedPrincipal` limit.

Remediation

Won't Fix

The client prefers to keep the round-down approach, as changing it to round up might lead to the failure of legitimate bridging in some cases.

#3 MPORTALLITE-003 Integer Truncation in HubPortal._mintOrUnlock() Leading to Potential bridgedPrincipal Check Bypass

ID	Summary	Severity
MPORTALLITE-003	An integer truncation exists in `HubPortal._mintOrUnlock()` where casting an input `amount_` to `uint240`. This could potentially allow the `bridgedPrincipal` check to be bypassed.	Low

Description

The `HubPortal._mintOrUnlock()` function directly casts the input `amount_` to `uint240` when calculating `principalAmount` using `IndexingMath.getPrincipalAmountRoundedDown(uint240(amount_), _currentIndex())`. If the value of `amount_` exceeds the maximum representable by a `uint240` integer (`2^240 - 1`), it will be truncated. This truncation results in a `principalAmount` smaller than intended. Consequently, this reduced `principalAmount` could incorrectly satisfy the `if (principalAmount > totalBridgedPrincipal)` condition, even if the original, non-truncated `amount_` represented a principal value significantly exceeding `totalBridgedPrincipal`.

Impact

Low

An attacker, capable of crafting messages from a single chain (e.g., by compromising the bridge or a chain), could potentially bypass the `bridgedPrincipal` check. Successful exploitation, leading to the theft of locked tokens, is contingent on the `HubPortal` contract holding an `mToken` balance greater than `2^240 - 1`. If the contract's actual balance is less than the non-truncated `amount_`, the `IERC20(mToken).transfer(recipient_, amount_);` call would fail due to insufficient funds.

Recommendation

It is recommended to use `SafeCast.toUint240()` when converting `amount_` to `uint240` to prevent potential truncation.

Remediation

Won't Fix

The client considers the conditions necessary for exploitation—specifically, the `HubPortal` contract's `mToken` balance exceeding `2^240 - 1`—to be unrealistic. This assessment is based on the nature of M as an overcollateralized stablecoin and the relatively small total circulation of USD compared to such a figure.

This assessment is considered reasonable under current conditions. However, this issue warrants reconsideration if features that might temporarily allow unbacked minting, such as flash mints, are introduced to M.

Revision History

Version	Date	Description
1.0	May 25, 2025	Initial version

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.