Consultancy Report
NM-0243 Investigation on Funds Locked in Mode's Bridge

NETHERMIND
SECURITY

(May 14, 2024)

Contents

1 Scenario Description

ETHER.FI has been ETHEREUM native since the inception of the protocol and recently deployed a cross-chain solution to enable native minting of weETH on BASE, LINEA, BLAST, MODE, and more to follow. Funds got stuck when syncing ETH from the MODE's LAYER 2 to the ETHEREUM LAYER 1. This issue happened due to a misconfiguration of the contracts. ETHER.FI attempted to use the existing **CrossDomainMessenger** contract between MODE and ETHEREUM. The architecture of this bridge is shown in Fig. 1.

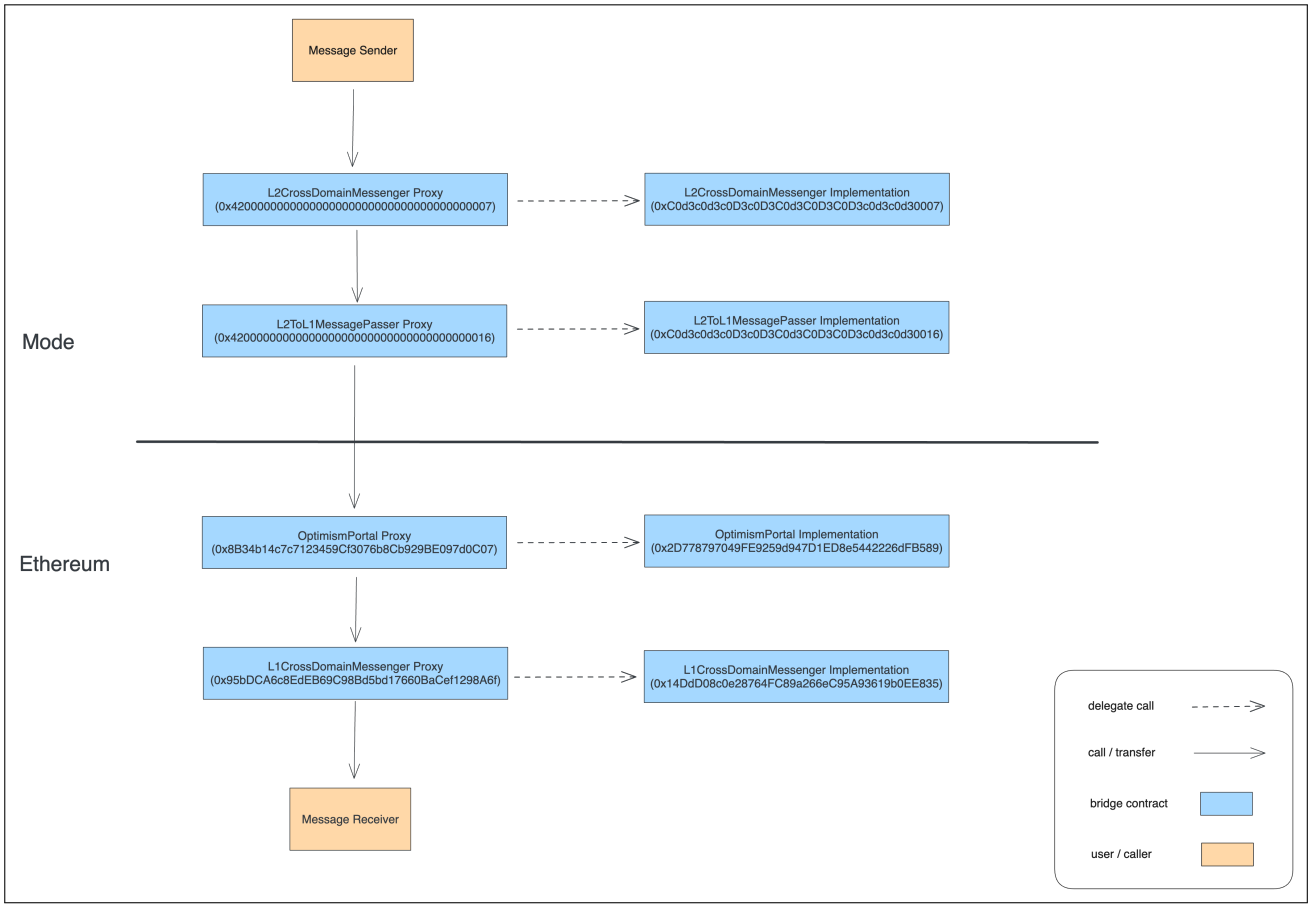

Fig 1: CrossDomainMessenger bridge architecture

The **CrossDomainMessenger** contract on ETHEREUM only accepts funds from the other side of the bridge (the **CrossDomainMessenger** contract deployed on MODE). The ETHER.FI team used the implementation contract of the **CrossDomainMessenger** on MODE, effectively sending the message from the wrong contract in MODE LAYER 2. This error causes the message to be invalid to the **CrossDomainMessenger** contract in LAYER 1. As a result, the message cannot be delivered from the **OptimismPortal** contract on Ethereum to the **CrossDomainMessenger** contract.

The following list shows the expected flow to sync funds from MODE to ETHEREUM:

(1) The sync process is starts on ETHER.FI contracts in the MODE's NETWORK;

(2) The ETHER.FI contract calls the function `sendMessage(...)` in the **L2CrossDomainMessenger** to send a message to the ETHER.FI contract on ETHEREUM using the **CrossDomainMessenger** mechanism;

> **Here is where we find the issue under discussion. ETH was sent to the implementation contract, not the proxy one.**

(3) The **L2CrossDomainMessenger** contract calls the function `initiateWithdrawal(...)` on the **L2ToL1MessagePasser** contract in MODE to record the message;

(4) The withdrawal is proven in the designated **OptimismPortal** in ETHEREUM;

(5) The user calls the function `finalizeWithdrawal(...)` in the contract **OptimismPortal** to execute the withdrawal initiated by the contract **L2CrossDomainMessenger**;

(6) The contract **`OptimismPortal`** calls the function `relayMessage(...)` in the contract **`L1CrossDomainMessenger`**;

> **The transaction is not passing this stage because the message is being sent from an address not recognized as a valid sender.**

(7) The contract **`L1CrossDomainMessenger`** delivers the message to `ETHER.FI` in `ETHEREUM`.

2 Relevant Addresses to this Investigation

This section summarizes the most relevant addresses considered in this investigation.

L2CrossDomainMessenger proxy contract in `MODE`	0x4200000000000000000000000000000000000007
L2CrossDomainMessenger implementation contract in `MODE`	0xC0d3c0d3c0D3c0D3C0d3C0D3C0D3c0d3c0d30007
L2ToL1MessagePasser proxy contract in `MODE`	0x4200000000000000000000000000000000000016
L2ToL1MessagePasser implementation contract in `MODE`	0xC0D3C0d3C0d3c0d3C0d3C0D3c0D3c0d3c0D30016
L1CrossDomainMessenger proxy contract in `ETHEREUM`	0x95bDCA6c8EdEB69C98Bd5bd17660BaCef1298A6f
L1CrossDomainMessenger implementation contract in `ETHEREUM`	0x14DdD08c0e28764FC89a266eC95A93619b0EE835
OptimismPortal proxy contract in `ETHEREUM`	0x8B34b14c7c7123459Cf3076b8Cb929BE097d0C07
OptimismPortal implementation contract in `ETHEREUM`	0x2D778797049FE9259d947D1ED8e5442226dFB589

3 Conditions that contributed to funds being locked

This investigation identified two situations contributing to the funds becoming locked.

(1) The `MODE` network documentation listed the wrong address for the contract **`L2CrossDomainMessenger`**;

(2) The contracts used for messaging between `MODE` and `ETHEREUM` networks were not updated with the latest updates in the `OPTIMISM` stack.

> **The links [1] and [2] present issues related to this investigation, where funds could be locked. The issue [1] presents that real values are used in the implementation contract. The issue [2] presents a similar scenario to the one from this investigation: sending to portal implementation leads to frozen funds.**

4 Timeline of Actions

- **Apr 19 2024 00:44:17 AM (-05:00 UTC)** - `ETHER.FI` starts the deployment and configuration of their Cross Chain/LayerZero enabled contracts to `MODE`;
- **Apr 16 2024 02:25:53 AM (-05:00 UTC)** - OFT Deployed;
- **Apr 16 2024 02:25:59 AM (-05:00 UTC)** - OFT initialize call. A call to the function `initialize(address l2ExchangeRateProvider, address rateLimiter, address tokenOut, uint32 dstEid, address messenger, address receiver, address delegate)` was executed, providing the wrong value `0xc0d3c0d3c0d3c0d3c0d3c0d3c0d3c0d3c0d30007` in the `messenger` parameter;
- **Apr 19 to May 8** - Users are depositing `ETH` on `MODE`, minting `weETH`;
- **Apr 25 2024 15:44:19 PM (-09:00 UTC)** - Sync operation started on Mode.
- **Apr 29 2024 04:08:51 AM (-05:00 UTC)** - Sync operation started on Mode.
- **Apr 29 2024 20:38:21 PM (-05:00 UTC)** - Sync operation started on Mode.
- **May 01 2024 21:11:53 PM (-05:00 UTC)** - Sync operation started on Mode.
- **May 06 2024 18:03:01 PM (-05:00 UTC)** - Sync operation started on Mode.
- **May 1 to May 9** - Withdrawal proof generation. The OP stack SDK successfully generated proofs for withdrawals on the `BLAST L2`, but the same script did not work for `MODE`.

5 Solution Proposed by the Ether.Fi Team

The ETHER.FI team proposed a temporary change to the contract **L1CrossDomainMessenger** that would allow them to receive the stuck funds. The change consists in setting the **L2CrossDomainMessenger** contract implementation in MODE as a valid message sender for the L1 part of the bridge.

```
diff --git a/packages/contracts-bedrock/src/L1/L1CrossDomainMess
index 4aaf2bd31..e36fc3cad 100644
--- a/packages/contracts-bedrock/src/L1/L1CrossDomainMessenger.s
+++ b/packages/contracts-bedrock/src/L1/L1CrossDomainMessenger.s
@@ -81,7 +81,7 @@ contract L1CrossDomainMessenger is CrossDomain

     /// @inheritdoc CrossDomainMessenger
     function _isOtherMessenger() internal view override returns
-        return msg.sender == address(portal) && portal.l2Sender() == address(otherMessenger);
+        return msg.sender == address(portal) && (portal.l2Sender() == address(otherMessenger) || portal.l2Sender() ==
 ↪  address(L2_CROSS_DOMAIN_MESSENGER_IMPLEMENTATION_ADDRESS));
     }

     /// @inheritdoc CrossDomainMessenger
```

6 Evaluation of the Solution Proposed by the Ether.Fi Team

As previously mentioned, the **CrossDomainMessenger** system is expected to have one endpoint on each chain. These endpoints are referred to as **L1CrossDomainMessenger** and **L2CrossDomainMessenger** contracts. The change proposed by the ETHER.FI team implies adding one more endpoint to this system in the MODE network. However, because the newly added source would be the current implementation contract of the **L2CrossDomainMessenger**, which is immutable, it is known that it would behave in the expected way.

During the review of this change, the Nethermind Security team did not identify any immediate risks if this modification is implemented TEMPORARILY to recover the funds. The L1CrossDomainMessenger can be upgraded to the proposed implementation, the funds recovered, and the contract upgraded again to the current implementation.

As shown in Section 3, the implementation contract:

(a) should not accept funds (reverting any transactions sending funds to it);

(b) if the protocol designers decide to accept funds in the implementation contract, the funds should be forwarded to L1.

However, given the intrinsic complexities of bridging mechanisms and the limited time dedicated to this review, we recommend obtaining a review from the auditing firm that has previously audited the bridge.

7 Disclaimer & Limitations of This Consultancy

The consultancy occurred between May 10th, 2024 (Friday) and May 12th, 2024 (Sunday). The time constraint limited its depth and thoroughness. This report has been prepared by Nethermind Security ("We" "us" "our") at the request of ETHER.FI ("client") for the purpose of providing an analysis of the issue (as detailed in the report) and evaluating the solution proposed by the ETHER.FI team.

The findings and opinions expressed herein are based on the information made available to us by the client and other relevant sources, as well as our professional expertise and judgment. While every effort has been made to ensure the accuracy and reliability of the information contained in this report, we cannot guarantee its completeness or suitability for any particular purpose.

Nethermind Security has conducted this analysis independently and without bias. Our recommendations are based solely on our professional assessment of the information available to us and are intended to assist the client in making informed decisions. However, it is important to note that our opinions may differ from those of other experts or stakeholders involved in this matter.

This report does not indicate the endorsement of any particular project or team nor guarantee its security. No third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, Nethermind Security disclaims any liability in connection with this report, its content, and any related services and products and anyone's use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Nethermind Security shall not be liable for any loss, damage, or expense incurred by the client or any third party arising from the use of this report or reliance on its contents. The client or any other part acknowledges that the decision to implement any recommendations contained herein is ultimately their own responsibility.

As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. The report, its contents, access, and/or usage thereof, including any associated services or materials,

are provided for informational purposes only. They should not be construed or relied upon as financial, investment, tax, legal, regulatory, or any other form of professional advice. Readers are advised to seek appropriate independent advice tailored to their specific circumstances before making any financial, investment, or legal decisions.

8 About Nethermind

Nethermind is a Blockchain Research and Software Engineering company. Our work touches every part of the web3 ecosystem - from layer 1 and layer 2 engineering, cryptography research, and security to application-layer protocol development. We offer strategic support to our institutional and enterprise partners across the blockchain, digital assets, and DeFi sectors, guiding them through all stages of the research and development process, from initial concepts to successful implementation. We offer security audits of projects built on EVM-compatible chains and Starknet. We are active builders of the Starknet ecosystem, delivering a node implementation, a block explorer, a Solidity-to-Cairo transpiler, and formal verification tooling. Nethermind also provides strategic support to our institutional and enterprise partners in blockchain, digital assets, and decentralized finance (DeFi). In the next paragraphs, we introduce the company in more detail.

Blockchain Security: At Nethermind, we believe security is vital to the health and longevity of the entire Web3 ecosystem. We provide security services related to Smart Contract Audits, Formal Verification, and Real-Time Monitoring. Our Security Team comprises blockchain security experts in each field, often collaborating to produce comprehensive and robust security solutions. The team has a strong academic background, can apply state-of-the-art techniques, and is experienced in analyzing cutting-edge Solidity and Cairo smart contracts, such as ArgentX and StarkGate (the bridge connecting Ethereum and StarkNet). Most team members hold a Ph.D. degree and actively participate in the research community, accounting for 240+ articles published and 1,450+ citations in Google Scholar. The security team adopts customer-oriented and interactive processes where clients are involved in all stages of the work.

Blockchain Core Development: Our core engineering team, consisting of over 20 developers, maintains, improves, and upgrades our flagship product - the Nethermind Ethereum Execution Client. The client has been successfully operating for several years, supporting both the Ethereum Mainnet and its testnets, and now accounts for nearly a quarter of all synced Mainnet nodes. Our unwavering commitment to Ethereum's growth and stability extends to sidechains and layer 2 solutions. Notably, we were the sole execution layer client to facilitate Gnosis Chain's Merge, transitioning from Aura to Proof of Stake (PoS), and we are actively developing a full-node client to bolster Starknet's decentralization efforts. Our core team equips partners with tools for seamless node set-up, using generated docker-compose scripts tailored to their chosen execution client and preferred configurations for various network types.

DevOps and Infrastructure Management: Our infrastructure team ensures our partners' systems operate securely, reliably, and efficiently. We provide infrastructure design, deployment, monitoring, maintenance, and troubleshooting support, allowing you to focus on your core business operations. Boasting extensive expertise in Blockchain as a Service, private blockchain implementations, and node management, our infrastructure and DevOps engineers are proficient with major cloud solution providers and can host applications in-house or on clients' premises. Our global in-house SRE teams offer 24/7 monitoring and alerts for both infrastructure and application levels. We manage over 5,000 public and private validators and maintain nodes on major public blockchains such as Polygon, Gnosis, Solana, Cosmos, Near, Avalanche, Polkadot, Aptos, and StarkWare L2. Sedge is an open-source tool developed by our infrastructure experts, designed to simplify the complex process of setting up a proof-of-stake (PoS) network or chain validator. Sedge generates docker-compose scripts for the entire validator set-up based on the chosen client, making the process easier and quicker while following best practices to avoid downtime and being slashed.

Cryptography Research: At Nethermind, our Cryptography Research team is dedicated to continuous internal research while fostering close collaboration with external partners. The team has expertise across a wide range of domains, including cryptography protocols, consensus design, decentralized identity, verifiable credentials, Sybil resistance, oracles, and credentials, distributed validator technology (DVT), and Zero-knowledge proofs. This diverse skill set, combined with strong collaboration between our engineering teams, enables us to deliver cutting-edge solutions to our partners and clients.

Smart Contract Development & DeFi Research: Our smart contract development and DeFi research team comprises 40+ world-class engineers who collaborate closely with partners to identify needs and work on value-adding projects. The team specializes in Solidity and Cairo development, architecture design, and DeFi solutions, including DEXs, AMMs, structured products, derivatives, and money market protocols, as well as ERC20, 721, and 1155 token design. Our research and data analytics focuses on three key areas: technical due diligence, market research, and DeFi research. Utilizing a data-driven approach, we offer in-depth insights and outlooks on various industry themes.

Our suite of L2 tooling: Warp is Starknet's approach to EVM compatibility. It allows developers to take their Solidity smart contracts and transpile them to Cairo, Starknet's smart contract language. In the short time since its inception, the project has accomplished many achievements, including successfully transpiling Uniswap v3 onto Starknet using Warp.

- **Voyager** is a user-friendly Starknet block explorer that offers comprehensive insights into the Starknet network. With its intuitive interface and powerful features, Voyager allows users to easily search for and examine transactions, addresses, and contract details. As an essential tool for navigating the Starknet ecosystem, Voyager is the go-to solution for users seeking in-depth information and analysis;
- **Horus** is an open-source formal verification tool for StarkNet smart contracts. It simplifies the process of formally verifying Starknet smart contracts, allowing developers to express various assertions about the behavior of their code using a simple assertion language;
- **Juno** is a full-node client implementation for Starknet, drawing on the expertise gained from developing the Nethermind Client.

Learn more about us at nethermind.io.