Incident Report

Router Chain Cross-Chain Request Exploit

v0.1

July 7, 2025

Table of Contents

License

Disclaimer

THE CONTENT OF THIS REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS REPORT.

THIS REPORT WAS PREPARED EXCLUSIVELY FOR AND IN THE INTEREST OF THE CLIENT AND SHALL NOT CONSTRUE ANY LEGAL RELATIONSHIP TOWARDS THIRD PARTIES. IN PARTICULAR, THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THIRD PARTIES AND PROVIDE NO WARRANTIES REGARDING THE FACTUAL ACCURACY OR COMPLETENESS OF THE REPORT.

FOR THE AVOIDANCE OF DOUBT, NOTHING CONTAINED IN THIS REPORT SHALL BE CONSTRUED TO IMPOSE ADDITIONAL OBLIGATIONS ON COMPANY, INCLUDING WITHOUT LIMITATION WARRANTIES OR LIABILITIES.

This review has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Executive Summary

An exploit occurred on Router Chain, leveraging a vulnerability in the cross-chain request logic related to improper validation of the `RequestSender` field. This allowed an attacker to craft unauthorized requests and bypass access controls on the receiving side.

The logic in question was within the scope of the audit performed by Oak Security and was partially covered in Finding 4 titled "Anyone can submit forged cross-chain requests" of our audit report "2024-04-30 Audit Report - Router Chain 2 v1.0.pdf". However, the specific instance that led to the exploit was not explicitly included as an example in the report, and the issue was incorrectly marked as resolved.

Technical Details

- **Vulnerability Type**: Improper Access Control
- **Root Cause**: Absence of proper validation on the RequestSender field in cross-chain messaging logic
- **Impact**: Malicious cross-chain requests could be forged by passing arbitrary RequestSender values, including the AssetForwarderMiddleware contract address
- **Location**: Cross-chain logic within Router Chain smart contracts
- **Fix**: Validation logic was updated to explicitly verify the RequestSender field (PR #235). This change has been reviewed by three of Oak Security's auditors and no additional issues were found.

Audit Context and Responsibility

The relevant code was in scope during our security audit, and our report highlighted the general class of issue under Finding 4. However, the specific exploit vector was not included as a concrete example, and the issue was erroneously marked as resolved. This represents a miss in our review process, which we are now actively addressing through an internal post-mortem and methodological improvements.

At Oak Security and our brand Solidified, we have performed more than 600 audits, and this is the first instance known to us where code we have audited has been exploited.

It is crucial to emphasize that while audits are rigorous, they can't promise an entirely vulnerability-free codebase. That is why we consistently advise adopting a multi-layered security approach.

Detailed Issue Description

Insufficient access control allows forged outbound token requests

Severity: Critical

In `x/crosschain/keeper/native_token_flow.go:44-74`, the `HandleOutboundNativeTokenFlow` function processes outbound `types.CrosschainRequest` requests.

However, it does not validate that the `RequestSender` field of the request corresponds to the actual actor that initiated the message.

This lack of verification allows a malicious actor to forge a `CrosschainRequest` with an arbitrary address and submit it to the Router Chain.

By impersonating the Asset Forwarder Middleware contract, the attacker can bypass security checks in the Solidity contract, which assumes that only the Asset Forwarder Middleware contract addresses are allowed to initiate specific actions.

As a result, an attacker could issue unauthorized outbound token requests, potentially draining funds from the chain.

Recommendation

We recommend implementing strict validation of the `RequestSender` field in the `HandleOutboundNativeTokenFlow` function to ensure that it matches the actual message sender.

Status: Resolved

Recommendations for Future Risk Reduction

To enhance the security posture of Router and reduce the likelihood and impact of future incidents, we strongly recommend implementing a multi-layered security strategy, including but not limited to the following:

Development and Testing

- Follow best practices for clarity, modularity, and code quality
- Maintain high-quality documentation for developers and reviewers
- Implement exhaustive unit, integration, and end-to-end testing

- Ensure thorough test coverage of edge cases and non-happy paths

Review and Deployment

- Enforce internal peer reviews for all code changes
- Require formal code approvals before merging or deploying
- Apply strict deployment controls to prevent accidental or malicious rollouts

Third-Party Security Measures

- Engage multiple external security providers for audits and code reviews
- Consider formal verification for critical logic and invariants
- Sponsor audit contests or public code reviews for broad feedback
- Launch and maintain an incentivized bug bounty program

Monitoring and Response

- Deploy monitoring and alerting systems to detect abnormal behavior
- Introduce rate-limiting mechanisms to reduce potential exploit impact
- Implement circuit breakers or kill switches for emergency deactivation of components

Commitment to Transparency and Improvement

At Oak Security, we take our role as a security partner seriously. We are currently conducting an internal review of our processes to better catch such edge cases in the future and refine verification workflows.

We commend the Router Chain team for their prompt response and patch, and we remain available to assist with post-incident analysis, improved safeguards, and additional reviews.