Lagrange: LAToken

Security Assessment

April 19, 2025

Prepared for:
Amir Rezaizadeh
Lagrange

Prepared by: **Anish Naik**

Table of Contents

Project Summary

Contact Information

The following project manager was associated with this project:

Emily Doucette, Project Manager
emily.doucette@trailofbits.com

The following engineering director was associated with this project:

Jim Miller, Engineering Director, Blockchain and Cryptography
james.miller@trailofbits.com

The following consultant was associated with this project:

Anish Naik, Consultant
anish.naik@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
April 10, 2025	Pre-project kickoff call
April 16, 2025	Delivery of report draft
April 17, 2025	Report readout meeting
April 19, 2025	Delivery of final comprehensive report

Executive Summary

Engagement Overview

Lagrange engaged Trail of Bits to review the security of its LAToken, an ERC-20, LayerZero-compatible Omnichain Fungible Token (OFT) that is deployed across a variety of EVM-compatible chains. To support ERC-20 permit and OpenZeppelin access control functionality, the OFT library was forked and vendored.

One consultant conducted the review from April 14 to April 16, 2025, for a total of three engineer-days of effort. Our testing efforts focused on validating the LayerZero integration, access controls, and upgradeability of the system. With full access to source code and documentation, we performed static and dynamic testing of the target, using automated and manual processes.

Observations and Impact

We did not identify any issues that would affect the confidentiality, integrity, or availability of the LAToken system. The contracts have sufficient data validation and access controls to prevent malicious behavior. Additionally, the changeset to the OFT library logic is minimal and does not adversely affect the expected behavior of the integration.

The only concern that we identified during this audit was the lack of testing of the LayerZero integration (TOB-LATOKEN-1). Although manual review did not identify any issues regarding the integration, the lack of testing will cause issues in the maintenance of the product. As the OFT library changes, it will not be possible to dynamically validate whether the vendored code causes issues with cross-chain transfers.

Recommendations

Based on the codebase maturity evaluation and findings identified during the security review, Trail of Bits recommends that Lagrange take the following steps before deployment of the LAToken across the supported chains:

- **Add testing for the LayerZero integration.** At a minimum, unit tests that locally mock a transfer of LATokens between two chains would be beneficial. If the OFT library is updated, these tests should aid in dynamically capturing issues early.

Finding Severities and Categories

The following tables provide the number of findings by severity and category.

EXPOSURE ANALYSIS

Severity	*Count*
High	0
Medium	0
Low	0
Informational	1
Undetermined	0

CATEGORY BREAKDOWN

Category	*Count*
Testing	1

Project Goals

The engagement was scoped to provide a security assessment of Lagrange's LAToken. Specifically, we sought to answer the following non-exhaustive list of questions:

- Can an attacker bypass access controls and mint LATokens?
- Do the modifications to the LayerZero OFT library cause storage layout or access control issues?
- Do the modifications to the base ERC-20 implementation lead to the violation of any invariants?

Project Targets

The engagement involved reviewing and testing the following target.

Lagrange LAToken Contracts

Repository	https://github.com/Lagrange-Labs/lagrange-lpn-contracts-private
Version	`91c79e3`
Type	Solidity
Platform	Ethereum and other EVM-compatible chains

Project Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. Our approaches included the following:

- **`LATokenMintable` and `LAToken`:** The `LATokenMintable` contract is deployed on Ethereum and allows the owner to mint tokens at a given rate. The `LAToken` contract will be deployed on other EVM-compatible chains and enables cross-chain transfers using LayerZero. We performed a manual review of the contracts and investigated the following:
    - We reviewed the linear inflation model for general correctness and checked whether the arithmetic is vulnerable to overflows, excessive precision loss, or division by zero.
    - We reviewed the access controls and checked whether the custom token logic violates any core ERC-20 invariants.
    - We reviewed the upgradeability pattern of the tokens for general correctness.
    - We reviewed the deployment contracts for general correctness.
- **Vendored LayerZero OFT library code:** The LayerZero OFT library was vendored to support ERC-20 permit and OpenZeppelin access control functionality. We performed a manual review of this logic and investigated the following:
    - We assessed whether the changes made to the inheritance tree of the contracts would lead to storage collisions.
    - We assessed whether introducing the ERC-20 permit or OpenZeppelin access control functionality would lead to undefined behavior or the violation of any LayerZero OFT invariants.

Codebase Maturity Evaluation

Trail of Bits uses a traffic-light protocol to provide each client with a clear understanding of the areas in which its codebase is mature, immature, or underdeveloped. Deficiencies identified here often stem from root causes within the software development life cycle that should be addressed through standardization measures (e.g., the use of common libraries, functions, or frameworks) or training and awareness programs.

Category	Summary	Result
Arithmetic	The system uses Solidity v0.8, which has built-in checked arithmetic. There is only one arithmetic formula of concern, the linear inflation curve, which has sufficient unit testing and is not vulnerable to excessive precision loss, overflow, or division by zero.	**Strong**
Auditing	All the state-changing functions emit the necessary events.	**Strong**
Authentication / Access Controls	We did not identify any opportunities to bypass the system's access controls.	**Strong**
Complexity Management	To support ERC-20 permit and OpenZeppelin access control functionality, the LayerZero OFT library had to be forked and vendored. Thus, maintaining upstream bug fixes and changes increases system complexity. The overall system logic itself is well composed and easy to reason through.	**Moderate**
Cryptography and Key Management	Cryptographic primitives are not used in the system.	**Not Applicable**
Decentralization	The owner of the token is a trusted actor that is responsible for upgrading the system, configuring cross-chain transfers, and managing the overall token liquidity. The owner cannot rug pull users (except through a malicious upgrade), pause the contracts, or disable transfers.	**Satisfactory**
Documentation	The inline documentation is sufficient to understand the	**Satisfactory**

	functionality of the system. It would be beneficial to document all the changes that have been made to the OFT library. This will aid in future maintenance of the system.	
Low-Level Manipulation	Assembly is not used in the system.	**Not Applicable**
Testing and Verification	There are sufficient tests for the core functionality of the token. However, there are no tests to validate the LayerZero integration and cross-chain transfers. Since the OFT library has been vendored, ensuring that the integration works as expected would be beneficial.	**Moderate**
Transaction Ordering	There are no transaction ordering risks associated with this system.	**Not Applicable**

Summary of Findings

The table below summarizes the findings of the review, including details on type and severity.

ID	Title	Type	Severity
1	Insufficient testing of cross-chain transfers	Testing	**Informational**

Detailed Findings

1. Insufficient testing of cross-chain transfers	
Severity: **Informational**	Difficulty: **Low**
Type: Testing	Finding ID: TOB-LATOKEN-1
Target: `src/latoken`, `vendor/layerzero`	

Description
There are no tests to validate that the LAToken can be transferred between two chains using the LayerZero OFT integration.

The OFT library had to be forked and vendored to support ERC-20 permit and OpenZeppelin access control functionalities. The impact of these changes on cross-chain transfers has not been dynamically tested locally or on a testnet.

Adding these tests is critical for the future maintenance of the LAToken.

Exploit Scenario
LayerZero releases a new version of the OFT library that changes how cross-chain transfers work. Part of the custom code written by the Lagrange team now breaks the ability to transfer tokens across chains.

Since there are no tests, the tokens are upgraded across all chains. However, the tokens can no longer be transferred, which causes the liquidity to become segmented and causes a large token sell-off.

Recommendations
Short term, add unit and testnet testing for the LAToken to validate that cross-chain transfers still work with the vendored code.

A. Vulnerability Categories

The following tables describe the vulnerability categories, severity levels, and difficulty levels used in this document.

Vulnerability Categories	
Category	**Description**
Access Controls	Insufficient authorization or assessment of rights
Auditing and Logging	Insufficient auditing of actions or logging of problems
Authentication	Improper identification of users
Configuration	Misconfigured servers, devices, or software components
Cryptography	A breach of system confidentiality or integrity
Data Exposure	Exposure of sensitive information
Data Validation	Improper reliance on the structure or values of data
Denial of Service	A system failure with an availability impact
Error Reporting	Insecure or insufficient reporting of error conditions
Patching	Use of an outdated software package or library
Session Management	Improper identification of authenticated users
Testing	Insufficient test methodology or test coverage
Timing	Race conditions or other order-of-operations flaws
Undefined Behavior	Undefined behavior triggered within the system

Severity Levels	
Severity	**Description**
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The flaw is well known; public tools for its exploitation exist or can be scripted.
Medium	An attacker must write an exploit or will need in-depth knowledge of the system.
High	An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.

B. Code Maturity Categories

The following tables describe the code maturity categories and rating criteria used in this document.

Code Maturity Categories	
Category	**Description**
Arithmetic	The proper use of mathematical operations and semantics
Auditing	The use of event auditing and logging to support monitoring
Authentication / Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system
Complexity Management	The presence of clear structures designed to manage system complexity, including the separation of system logic into clearly defined functions
Cryptography and Key Management	The safe use of cryptographic primitives and functions, along with the presence of robust mechanisms for key generation and distribution
Decentralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Documentation	The presence of comprehensive and readable codebase documentation
Low-Level Manipulation	The justified use of inline assembly and low-level calls
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage
Transaction Ordering	The system's resistance to transaction-ordering attacks

Rating Criteria	
Rating	**Description**
Strong	No issues were found, and the system exceeds industry standards.
Satisfactory	Minor issues were found, but the system is compliant with best practices.
Moderate	Some issues that may affect system safety were found.
Weak	Many issues that affect system safety were found.

Missing	A required component is missing, significantly affecting system safety.
Not Applicable	The category does not apply to this review.
Not Considered	The category was not considered in this review.
Further Investigation Required	Further investigation is required to reach a meaningful conclusion.

C. Recommendations for Maintaining a Forked Repository

The following are recommendations for simplifying the management of the forked LayerZero OFT library:

- **Minimize the changes required to the original code.** The most important factor in managing upstream changes while retaining custom code is to minimize the changeset. It is preferable to add all custom code to separate files with a clear prefix/suffix. If this is not possible, using inline comments highlighting custom code will make it easier to identify such code during merges.
- **Merge upstream changes frequently.** Creating an automated or semiautomated process to consistently merge upstream changes will reduce risks associated with reviewing multithousand-line PRs that have a large number of breaking changes.
- **Avoid directly updating the API of the original code.** Updating the API directly can have unexpected consequences within the system and any integrations. Instead, extend the API with any custom behavior.
- **Upstream custom changes.** Although it is not always possible, it is best to upstream any changes to the original repository. This will reduce the size of the changeset that needs to be maintained. Upstreaming such changes can also allow for extensibility of the original code (e.g., a PR can be submitted that allows the new code to be put in a separate file).
- **Validate that all tests are passing.** All the original tests and the custom tests must always be validated when upstream changes are merged.

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries and government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on X and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact or email us at info@trailofbits.com.

Trail of Bits, Inc.
228 Park Ave S #80688
New York, NY 10003
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

Trail of Bits performed all activities associated with this project in accordance with a statement of work and an agreed-upon project plan.

Security assessment projects are time-boxed and often rely on information provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test software controls and security properties. These techniques augment our manual security review work, but each has its limitations. For example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. A project's time and resource constraints also limit their use.