

Produced for



by



Contents

1	Executive Summary	3
2	Assessment Overview	5
3	Limitations and use of report	7
4	Terminology	8
5	Findings	9
6	Resolved Findings	10

1 Executive Summary

Dear Hyperlane team,

Thank you for trusting us to help Hyperlane with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Superchain USDT according to [Scope](#) to support you in forming an opinion on their security risks.

Hyperlane implements a bridging mechanism that enables the transfer of USDT from the CELO network to other chains within the superchain cluster. To accomplish this, USDT is wrapped into Super USDT with a mechanism similar to what Velodrome uses.

The subjects covered in our audit are functional correctness, upgradeability, testing and documentation. Only minor issues were found which have all been addressed. Documentation is sufficient. Testing could be improved regarding upgrades.

In summary, we find that the codebase provides a high level of security.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered, and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,

ChainSecurity

1.1 Overview of the Findings

Below we provide a brief numerical overview of the findings and how they have been addressed.

-Severity Findings	0
-Severity Findings	0
-Severity Findings	0
-Severity Findings	0

2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

2.1 Scope

The assessment was performed on the source code files inside the Superchain USDT repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	14 Feb 2025	976f76cd5785db88ddc2dab0b8f323370e53808d	Initial Version
2	14 Feb 2025	5b7ad15b6f3a2a702c483e368bfbf52b504a27c0	Fixes
3	14 Feb 2025	ff1e46fd60dec49e3839d55187e13e40fe1c1e4e	Final Version

For the solidity smart contracts, the compiler version 0.8.24 was chosen.

The following contracts in the folder `src/` are in the scope of the review:

```
xerc20:
  MintLimits.sol
  XERC20.sol
  XERC20Factory.sol
  XERC20Lockbox.sol
```

2.1.1 Excluded from scope

Any contracts that are not explicitly listed above are out of the scope of this review. Namely, third-party libraries and integrations, such as OpenZeppelin, are explicitly out of the scope of this review. They are assumed to be secure and conform to their specification. We assume that the Superchain bridge functions as documented.

2.2 System Overview

This system overview describes the initially received version () of the contracts as defined in the [Assessment Overview](#).

Hyperlane offers a bridging mechanism that enables the transfer of USDT from the CELO network to other chains within the superchain cluster. To accomplish this, USDT is wrapped into Super USDT. A lockbox is deployed to escrow USDT and mint Super USDT (XERC20 token), with both minting and burning operations being capped by a rate-limiting buffer.

The system comprises the following contracts:

XERC20Lockbox: Acts as a secure intermediary that allows users to deposit ERC20 tokens to mint corresponding XERC20 tokens, and to burn XERC20 tokens to withdraw the underlying ERC20 tokens.

DRAFT

XERC20: An extension of the standard ERC20 token that incorporates cross-chain capabilities, token wrapping/unwrapping support via the designated lockbox, and integrated rate-limiting on minting and burning operations. Minting and burning are restricted by a buffer, with buffer management handled by the `MintLimits` contract, which is inherited by XERC20. It is important to note that the token in its current implementation doesn't support any of the USDT features such as fees on transfer (if enabled) or blacklisting.

XERC20Factory: Facilitates the deterministic deployment of XERC20 tokens (and their corresponding lockboxes) across chains using the CreateX mechanism.

2.2.1 Trust Model

We infer the following trust model:

Owner: The Owner has full administrative control over sensitive configuration functions in the system. This includes setting or updating rate limits (buffer caps and rate limits per second), adding or removing bridges, and deploying new instances via the factory. The Owner is fully trusted and is responsible for establishing appropriate buffers to mitigate the risk of untrusted parties arbitrarily minting or burning tokens.

Superchain ERC20 Bridge: This designated component (hardcoded as 0x420000000000000000000000000000000028 for all chains in the XERC20 contract) is responsible for initiating cross-chain minting and burning operations. It is fully trusted. However, if compromised, the bridge could mint unbacked tokens, posing a significant risk. Additionally, the bridge is vital for the system's overall liveness.

End Users: End Users interact with the system by depositing ERC20 tokens into the lockbox (to mint XERC20 tokens), withdrawing underlying ERC20 tokens (by burning XERC20 tokens), and transferring tokens. They do not have any administrative privileges and are considered untrusted.

2.2.2 Changes in

The main change introduced in the [XERC20 upgrade](#) is that the XERC20 tokens become upgradeable. This means that their owner, which is trusted, could in the future upgrade the underlying logic or give the ownership of the proxy admin which controls the logic upgrades to another user.

3 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

4 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- *Likelihood* represents the likelihood of a finding to be triggered or exploited in practice
- *Impact* specifies the technical and business-related consequences of a finding
- *Severity* is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

Likelihood	Impact		
	High	Medium	Low
High			
Medium			
Low			

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

5 Findings

In this section, we describe any open findings. Findings that have been resolved have been moved to the [Resolved Findings](#) section. The findings are split into these different categories:

Below we provide a numerical overview of the identified findings, split up by their severity.

-Severity Findings	0
-Severity Findings	0
-Severity Findings	0
-Severity Findings	0

6 Resolved Findings

Here, we list findings that have been resolved during the course of the engagement. Their categories are explained in the [Findings](#) section.

Below we provide a numerical overview of the identified findings, split up by their severity.

-Severity Findings	0
-Severity Findings	0
-Severity Findings	0
-Severity Findings	0
Informational Findings	2

- [Missing ProxyAdmin](#)
- [Wrong Factory Specification](#)

6.1 Missing ProxyAdmin

CS-HLX20-001

XERC20 is deployed behind a `TransparentUpgradeableProxy`. The owner of the proxy is a `ProxyAdmin` for chains other than CELO. However, for the XERC20 deployed on CELO, it's just the `XERC20Factory` owner.

Code corrected:

A proxy admin is now used for both chains.

6.2 Wrong Factory Specification

CS-HLX20-002

The specs of the lockbox creation documentation state "Supports 18 decimal tokens only", but the underlying arithmetic and token handling are implemented for tokens with 6 decimals.

Specification changed:

The specification has been updated to reflect that contract should only be used for tokens with 6 decimals.