GUARDIAN

USDT0

HyperLiquidComposer

Security Assessment

March 2, 2025

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, Robert Reigada, 0xCiphky

Client Firm USDT0

Final Report Date March 2, 2025

Audit Summary

USDT0 engaged Guardian to review the security of their USDT0 composer to link USDT0 on the HyperLiquid L1 and the HyperEVM. From the 17th of February to the 20th of February, a team of 4 auditors reviewed the source code in scope. All findings have been recorded in the following report.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **HyperEVM**

✅ Verify the authenticity of this report on Guardian’s GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/Everdawn-Labs/usdt0-oft-contracts/pull/58

Table of Contents

Project Overview

Project Summary

Project Name	USDT0
Language	Solidity
Codebase	https://github.com/Everdawn-Labs/usdt0-oft-contracts
Commit(s)	ca772d9b1ab1bd798761a7c366a3349f6819bd0f

Audit Summary

Delivery Date	March 2, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	0	0	0	0	0	0
● Medium	0	0	0	0	0	0
● Low	1	0	0	0	0	1
● Info	8	0	0	4	0	4

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	● Critical	● High	● Medium
Likelihood: *Medium*	● High	● Medium	● Low
Likelihood: *Low*	● Medium	● Low	● Low

Impact

High	Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.
Medium	A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.
Low	Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High	The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.
Medium	An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.
Low	Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Findings & Resolutions

ID	Title	Category	Severity	Status
L-01	System Address Receiver Undefined Behavior	Warning	● Low	Resolved
I-01	Trapped Funds With Malformed ComposeMsg	Trapped Funds	● Info	Acknowledged
I-02	Composed Messages Stuck For Certain Receivers	DoS	● Info	Acknowledged
I-03	Incorrect Documentation	Documentation	● Info	Resolved
I-04	Lacking Zero Address Checks	Validation	● Info	Resolved
I-05	Unnecessary Imports	Superfluous Code	● Info	Resolved
I-06	Lacking Token Validation	Validation	● Info	Resolved
I-07	Misleading Comment	Documentation	● Info	Acknowledged
I-08	Trapped ETH In lzCompose	Trapped Funds	● Info	Acknowledged

L-01 | System Address Receiver Undefined Behavior

Category	Severity	Location	Status
Warning	● Low	Global	Resolved

Description

The HyperLiquidComposer contract allows the user to transfer their USDT0 to any receiver address on the HyperLiquid L1. Included in this is the HyperLiquid system contract 0x2222222222222222222222222222222222222222.

It is unclear how the HyperLiquid system will handle a transfer from the composer contract to the system contract followed by another transfer from the system contract to the system contract.

Depending on the integration of USDT0 in the spot balances on the HyperLiquid L1 this may cause unexpected issues and has the potential to duplicate tokens depending on the behavior.

Recommendation

Consider explicitly disallowing a receiver address of the HyperLiquid system contract 0x2222222222222222222222222222222222222222 to avoid any unexpected behavior.

Resolution

USDT0 Team: Resolved.

I-01 | Trapped Funds With Malformed ComposeMsg

Category	Severity	Location	Status
Trapped Funds	● Info	HyperLiquidComposer.sol	Acknowledged

Description

The `lzCompose` function in the `HyperLiquidComposer` contract expects the `composeMsg` to contain an address and only an address in it's contents.

In the event that a `composeMsg` is provided which does not adhere to this structure and has less bytes than an address, the decoding reverts and the `lzCompose` call cannot be executed for the corresponding message.

In this case the USDT0 bridged to the `HyperLiquidComposer` contract is effectively trapped.

Recommendation

Be aware of this risk and be sure to include the proper validations in the UI when sending composed messages to the `HyperLiquidComposer`.

If further mitigation is desired, consider adding a rescue function where a trusted address can withdraw trapped USDT0 from the contract.

Resolution

USDT0 Team: Acknowledged.

I-02 | Composed Messages Stuck For Certain Receivers

Category	Severity	Location	Status
DoS	● Info	Global	Acknowledged

Description

The _beforeTokenTransfer function in the TetherToken base contract prevents transfers where the USDT0 contract or the zero address is the recipient.

Any composed messages with the recipient as the USDT0 contract or the zero address will therefore be unexecutable.

Notice that compose messages with a blacklisted address as the receiver will also be unexecutable, since the transferWithHop function attempts to transfer from the receiver address.

Recommendation

Be sure to validate in the UI that the receiver address is not the USDT0 contract on HyperEVM nor the zero address.

Resolution

USDT0 Team: Acknowledged.

I-03 | Incorrect Documentation

Category	Severity	Location	Status
Documentation	● Info	HyperLiquidComposer.sol	Resolved

Description

The comment for the constructor in the HyperLiquidComposer contract indicates that the constructor Constructs the StableComposer contract. However instead it should read Constructs the HyperLiquidComposer contract.

Additionally, the lzCompose function documentation indicates that the function is intended to perform a token swap and that it expects the encoded compose message to contain the swap amount and recipient address.

This misleading documentation about the expected contents of the compose message could be dangerous for users and integrators.

If a user were to supply a compose message which follows the documentation the compose message would be unexecutable and the user's funds would be trapped.

Recommendation

Update the documentation to reflect that the HyperLiquidComposer contract is constructed, what this contracts intent is, and that only a recipient address is expected.

Resolution

USDT0 Team: Resolved.

I-04 | Lacking Zero Address Checks

Category	Severity	Location	Status
Validation	● Info	HyperLiquidComposer.sol	Resolved

Description

The `HyperLiquidComposer` contract does not implement zero address validations in the constructor for the `_token`, `_endpoint`, or `_oApp` addresses.

Recommendation

Consider implementing zero address validations to reduce the risk of an incorrect deployment.

Resolution

USDT0 Team: Resolved.

I-05 | Unnecessary Imports

Category	Severity	Location	Status
Superfluous Code	● Info	HyperLiquidComposer.sol	Resolved

Description

In the HyperLiquidComposer contract the IOAppCore and OStableWrapper imports are unnecessary. Additionally, the SafeERC20 library usage and import is unnecessary.

Recommendation

Remove the IOAppCore, OStableWrapper, and SafeERC20 imports.

Resolution

USDT0 Team: Resolved.

I-06 | Lacking Token Validation

Category	Severity	Location	Status
Validation	● Info	HyperLiquidComposer.sol	Resolved

Description

The lzCompose function assumes that the token and OApp are connected, however there is no explicit validation in the constructor which asserts this at the Smart Contract level. If these addresses are somehow misconfigured it could lead to failed bridges and trapped funds.

Recommendation

Consider adding validation in the constructor to ensure that the OApp and token pair are correctly linked via the HyperLiquidExtension.oftContract and OUpgradeable.token functions.

Resolution

USDT0 Team: Resolved.

I-07 | Misleading Comment

Category	Severity	Location	Status
Documentation	● Info	TetherToken.sol: 27	Acknowledged

Description

The base TetherToken contract indicates that the isTrusted variable is an Unused variable retained to preserve storage slots across upgrades. However in the HyperliquidExtension this mapping is used to validate transferWithHop callers.

Recommendation

Consider removing or updating the comment for the Hyperliquid deployment. Otherwise add a new comment in the HyperliquidExtension contract.

Resolution

USDT0 Team: Acknowledged.

I-08 | Trapped ETH In lzCompose

Category	Severity	Location	Status
Trapped Funds	● Info	HyperLiquidComposer.sol	Acknowledged

Description

The lzCompose function is payable in the LayerZero interface, however the implementation in the HyperLiquidComposer contract does not expect to receive any native funds. As a result any native funds sent by an executor when executing the lzCompose function would be lost.

Recommendation

Consider validating that the msg.value provided to the lzCompose function is explicitly 0 since the payable modifier is necessary to adhere to the LayerZero interface.

Resolution

USDT0 Team: Acknowledged.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits