

USDT0 · SECURITY REVIEW

USDT0

Hyperliquid Deployment.

A security review of Hyperliquid Deployment.

CLIENT

USDT0

PROJECT

Hyperliquid
Deployment

AUDITORS

Guardian

AUDIT WINDOW

February 27 –
March 5, 2025

FINAL REPORT

March 5, 2025

§ 00

Contents.

20 pages: overview, confidence ranking, scope, 11 findings, disclaimer, and about Guardian.

§ 01	Overview	P. 03
§ 02	Confidence Ranking	P. 04
§ 03	Scope & Classifications	P. 05
§ 04	Findings	P. 06
	Main Review · 9 findings	P. 07
	Remediation Review · 2 findings	P. 16
§ 05	Disclaimer	P. 19
§ 06	About Guardian	P. 20

§ 01

Overview.

Guardian reviewed Hyperliquid Deployment across 2 review rounds. The final report includes 1 Low, 10 Informational findings.

AUDIT FIRM	REPOSITORY
Guardian	https://github.com/Everdawn-Labs/usdt0-oft-contracts, https://github.com/Everdawn-Labs/usdt0-tether-contracts-hardhat
CLIENT	METHODOLOGY
USDT0	Smart contract · Manual · Main Review · Remediation Review
PROJECT	MAIN REVIEW COMMIT
Hyperliquid Deployment	ca772d9b1ab1
PREPARED BY	REMEDIAION REVIEW COMMIT
Guardian	51582a6e98b5
AUDIT WINDOW	
February 27 – March 5, 2025	
FINAL REPORT	
March 5, 2025	

1.1 Confidence Ranking

5 / 5

Very High Confidence

Guardian assigns a Confidence Ranking of 5 out of 5, combining finding severity, remediation outcome, and the reviewer’s assessment of latent risk. The full scale is on the next page.

1	2	3	4	5
VERY LOW CONFIDENCE	LOW CONFIDENCE	MODERATE CONFIDENCE	HIGH CONFIDENCE	VERY HIGH CONFIDENCE

1.2 Findings at a Glance

SEVERITY	TOTAL	PENDING	ACK	PARTIAL	RESOLVED	FIX REVIEW
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	1	0
Informational	10	0	4	0	4	2
All severities	11	0	4	0	5	2

§ 02

Confidence Ranking.

Guardian assigns a project-level confidence score using the scale below. The ranking combines finding severity, remediation outcome, and the reviewer's assessment of latent risk.

CONFIDENCE RANKING	DEFINITION AND RECOMMENDATION	RISK PROFILE
5 Very High Confidence ASSIGNED	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. Recommendation: code is highly secure at time of audit.	0 High/Critical findings and few Low/Medium severity findings.
4 High Confidence	Code is clean, well-structured, and adheres to best practices. Design patterns are sound and test coverage is strong. Recommendation: suitable for deployment after remediations; consider periodic review with changes.	0-1 High/Critical findings per engagement week and little to no Medium severity findings.
3 Moderate Confidence	Code is functional but shows gaps in defensive programming or test coverage. Recommendation: remediate the findings and re-review the changes before deployment.	Some High/Critical findings, with various Medium severity findings.
2 Low Confidence	Recurring issues suggest systemic gaps rather than isolated mistakes. Recommendation: remediate, then commission a follow-up review at a frozen commit.	Over 6 High and Critical findings across the engagement.
1 Very Low Confidence	The volume and severity of findings indicate the codebase is not ready for production. Recommendation: substantial rework, then a full re-review.	Over 12 High and Critical findings across the engagement.

§ 03

Scope & Classifications.

Guardian reviewed Hyperliquid Deployment in <https://github.com/Everdawn-Labs/usdt0-oft-contracts>, <https://github.com/Everdawn-Labs/usdt0-tether-contracts-hardhat>.

3.1 Severity Matrix

	IMP · HIGH	IMP · MED	IMP · LOW
LIK · HIGH	Critical	High	Medium
LIK · MED	High	Medium	Low
LIK · LOW	Medium	Low	Low

IMPACT · HIGH

Significant loss of assets, harm to a group of users, or core protocol functionality is disrupted.

IMPACT · MEDIUM

Small amount of funds can be lost, or ancillary functionality is affected. Users may experience reduced or delayed receipt.

IMPACT · LOW

Unexpected behavior in some functionality that is notable but does not meet the criteria for higher severity.

LIKELIHOOD · HIGH

Possible with reasonable assumptions that mimic on-chain conditions; cost is low compared to amount gained.

LIKELIHOOD · MEDIUM

Possible only in uncommon cases or requires a large amount of capital relative to disruption caused.

LIKELIHOOD · LOW

Unlikely to ever occur in production.

§ 04

Findings.

Grouped by round. Visible IDs reset by severity within each round, so a finding is named the same way in the report as it was in the channel it was written in. Status is where each finding ended up; its own page says how.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
MAIN REVIEW					
L-01	System Address Receiver Undefined Behavior	Warning	LOW	RESOLVED	p. 7
I-01	Misleading Comment	Documentation	INFORMATIONAL	ACKNOWLEDGED	p. 8
I-02	Trapped Funds With Malformed ComposeMsg	Trapped Funds	INFORMATIONAL	ACKNOWLEDGED	p. 9
I-03	Lacking Zero Address Checks	Validation	INFORMATIONAL	RESOLVED	p. 10
I-04	Unnecessary Imports	Superfluous Code	INFORMATIONAL	RESOLVED	p. 11
I-05	Lacking Token Validation	Validation	INFORMATIONAL	RESOLVED	p. 12
I-06	Composed Messages Stuck For Certain Receivers	DoS	INFORMATIONAL	ACKNOWLEDGED	p. 13
I-07	Incorrect Documentation	Documentation	INFORMATIONAL	RESOLVED	p. 14
I-08	Trapped ETH In IzCompose	Trapped Funds	INFORMATIONAL	ACKNOWLEDGED	p. 15
REMEDIATION REVIEW					
I-01	Hyperliquid Deployment Concerns	Warning	INFORMATIONAL	REPORTED	p. 16
I-02	Unclear ERC20 Transfer Behavior	Warning	INFORMATIONAL	REPORTED	p. 18

RESOLVED

CATEGORY	Warning
LOCATION	Global
REVIEW	Main Review

CONFIDENTIAL P. 07 / 20

INFORMATIONAL I-01

ACKNOWLEDGED

Misleading Comment

CATEGORY Documentation
LOCATION TetherToken.sol: 27
REVIEW Main Review

— 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

Status: Acknowledged.

INFORMATIONAL I-02

ACKNOWLEDGED

Trapped Funds With Malformed ComposeMsg

CATEGORY	Trapped Funds
LOCATION	HyperLiquidComposer.sol
REVIEW	Main Review

DESCRIPTION

The `lzCompose` function in the `HyperLiquidComposer` contract expects the `composeMsg` to contain an address and only an address in its 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

Status: Acknowledged.

INFORMATIONAL I-03

RESOLVED

Lacking Zero Address Checks

CATEGORY	Validation
LOCATION	HyperLiquidComposer.sol
REVIEW	Main Review

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

Status: Resolved.

INFORMATIONAL I-04

RESOLVED

Unnecessary Imports

CATEGORY	Superfluous Code
LOCATION	HyperLiquidComposer.sol
REVIEW	Main Review

DESCRIPTION

In the `HyperLiquidComposer` contract the `I0AppCore` and `OStableWrapper` imports are unnecessary. Additionally, the `SafeERC20` library usage and import is unnecessary.

RECOMMENDATION

Remove the `I0AppCore`, `OStableWrapper`, and `SafeERC20` imports.

RESOLUTION

Status: Resolved.

INFORMATIONAL I-05

RESOLVED

Lacking Token Validation

CATEGORY	Validation
LOCATION	HyperLiquidComposer.sol
REVIEW	Main Review

DESCRIPTION

The `lzCompose` function assumes that the `token` and `0App` 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 `0App` and `token` pair are correctly linked via the `HyperLiquidExtension.oftContract` and `0Upgradeable.token` functions.

RESOLUTION

Status: Resolved.

INFORMATIONAL I-06

ACKNOWLEDGED

Composed Messages Stuck For Certain Receivers

CATEGORY	DoS
LOCATION	Global
REVIEW	Main Review

— 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

Status: Acknowledged.

INFORMATIONAL I-07

RESOLVED

Incorrect Documentation

CATEGORY	Documentation
LOCATION	HyperLiquidComposer.sol
REVIEW	Main Review

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 contract's intent is, and that only a recipient address is expected.

RESOLUTION

Status: Resolved.

INFORMATIONAL I-08

ACKNOWLEDGED

Trapped ETH In lzCompose

CATEGORY	Trapped Funds
LOCATION	HyperLiquidComposer.sol
REVIEW	Main Review

— 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

Status: Acknowledged.

INFORMATIONAL I-01

Hyperliquid Deployment Concerns

CATEGORY	Warning
LOCATION	Global
REVIEW	Remediation Review

— DESCRIPTION

The deployment and bridging of an ERC20 token on HyperEVM, intended to interact with a native spot asset on Hyperliquid L1, involves specific initialization requirements and token supply configurations that raise concerns about correctness and clarity.

When a spot token is deployed on Hyperliquid L1, there are multiple settings to take into account, especially, the maximum supply and the user genesis balances. The maximum supply represents the maximum total supply of the token across all chains and it is used mainly for internal accounting, to display market caps etc.

The user genesis balances are the accounts that receive the maximum supply when the token is created, or which is the same, the original minted balances.

The USDT token current `totalSupply` is equal to 76,918,584,460.692261 on Ethereum mainnet and, therefore, during the deployment into the Hyperliquid L1, it is required to configure the maximum supply to be 76,918,584,460.692261 as well. Moreover, in the user genesis `[0x222...222, 76,918,584,460.692261]` must be set which basically would say that the total supply should be assigned to the bridge. This, in effect, is just for virtual accounting as the bridge does not have the `76918584460692261` tokens, however, neither does the L1.

Later on, when a transfer from the EVM to the L1 occurs, all Hyperliquid does on the L1 is debit that amount from the `0x222...222` system address and credit the amount to the account that the amount was transferred from.

For example, see the genesis balance of UBTC which is configured to have the entire BTC supply of 21 Million: [<https://app.hyperliquid.xyz/explorer/token/0x8f254b963e8468305d409b33aa137c67>] (<https://app.hyperliquid.xyz/explorer/token/0x8f254b963e8468305d409b33aa137c67>)

— RECOMMENDATION

I-01 Hyperliquid Deployment Concerns

CONTINUED

To ensure the successful deployment and bridging of the USDT0 ERC20 token on HyperEVM with Hyperliquid L1, the total balance and user genesis balance for the bridge contract should be assigned to the total supply of USDT on Ethereum Mainnet at the time of listing. It is unclear how to handle tokens which may increase or decrease in totalSupply over time such as USDT, further clarity should be made by the Hyperliquid team before proceeding.

Finally, this process should be fully tested earlier on the Hyperliquid testnet following the steps mentioned above.

— RESOLUTION

Status: Reported during the fix review period.

INFORMATIONAL I-02

Unclear ERC20 Transfer Behavior

CATEGORY	Warning
LOCATION	Global
REVIEW	Remediation Review

— DESCRIPTION

In [this article](https://medium.com/@ambitlabs/the-not-so-definitive-guide-to-hyperliquid-precompiles-f0b6025bb4a3) regarding the functionality of the HyperEVM precompiles it is mentioned that:

```
```markdown
```

Accounts must already exists on the L1

When you first perform an interaction on the L1 from a new smart contract, the action will fail silently on the L1 if the account doesnt not already exists. This must be handled manually such that once the smart contract has been deployed and the address is known, a dust amount of USDC can be sent on the L1 from any account to the address of the smart contract. This forces the account to be created on the L1 and from there everything will work accordingly.

```
```
```

It is unclear whether this behavior would only apply to the 0x33333... system precompile address or if it also applies to ERC20 transfers with the 0x22222... system precompile. We are currently confirming this behavior with the HyperLiquid team.

— RECOMMENDATION

Be aware of this behavior with the 0x333333 system precompile address, and ensure that we are able to confirm the behavior of the 0x222222 ERC20 interactions before moving forward with deployment.

— RESOLUTION

Status: Reported during the fix review period.

§ 05

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.

§ 06

About Guardian.

Guardian is a smart contract security firm. Every engagement is worked by two independent teams competing on the same scope, each with its own tooling and methods, so no single reviewer's blind spot survives.

\$45B+DIGITAL ASSETS UNDER
REVIEW**200+**

ENGAGEMENTS COMPLETED

500+CRITICAL & HIGH FINDINGS
REPORTED

Reach us at audits@guardianaudits.com.