MixBytes()

stETH & wstETH on Unichain Deployment Verification Report

FEBRUARY 19, 2025

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

Scope

Network: Unichain

Scope:

Asset	Link	Comment
OpStackTokenRatePusher	0x3F9600439Ad97fC6f55C2AC7C118f8Fd0595eB74	Rate pusher for the Unichain network
OssifiableProxy	0x755610f5Be536Ad7afBAa7c10F3E938Ea3aa1877	Ossifiable proxy for the L1LidoTokensBridge
L1LidoTokensBridge	0x6078232C54d956c901620fa4590e0F7E37c2B82f	Implementation of the L1LidoTokensBridge
OssifiableProxy	0xc02fE7317D4eb8753a02c35fe019786854A92001	Ossifiable proxy for the ERC20BridgedPermit
ERC20BridgedPermit	0xB5CF096A406C1D5297D2493073168F44EB4a1A1d	WstETH
OssifiableProxy	0x81f2508AAC59757EF7425DDc9717AB5c2AA0A84F	Ossifiable proxy for the ERC20RebasableBridgedPermit
ERC20RebasableBridgedPermit	0x5A007D6E37633FB297b82c074b94Bb29546BEbc3	StETH
OssifiableProxy	0xD835fAC9080396CCE95bDf9EcC7cc27Bab12c9f8	Ossifiable proxy for the TokenRateOracle
TokenRateOracle	0x537A7F9D551da3C2800cB11ca17f2946D21029AF	Implementation of the TokenRateOracle

Asset	Link	Comment
OssifiableProxy	0x1A513e9B6434a12C7bB5B9AF3B21963308DEE372	Ossifiable proxy for the L2ERC20ExtendedTokensBridge
L2ERC20ExtendedTokensBridge	0x332CA368dd09AD309c51dC6350730e0Bca85CffE	Implementation of the L2ERC20ExtendedTokensBridge
Governance Bridge Executor	0x3b00f262e39372DF2756f809DD5DC36aeEdFC4A0	Governance executor for the Unichain network

Audit reports:

OpStackTokenRatePusher, L1LidoTokensBridge, ERC20BridgedPermit, ERC20RebasableBridgedPermit, TokenRateOracle, L2ERC20ExtendedTokensBridge: report

Deployment scripts:
https://github.com/lidofinance/multichain-automaton/commit/a903c01d8e056f59ee998789a9864768146d4e6a

Initialized roles:
Proxy_admin for L1LidoTokensBridge: Lido Aragon Agent

Proxy_admin for L2ERC20ExtendedTokensBridge, ERC20BridgedPermit, ERC20RebasableBridgedPermit, TokenRateOracle: OptimismBridgeExecutor

Lido Aragon Agent (0x3e40d73eb977dc6a537af587d48316fee66e9c8c) is granted DEFAULT_ADMIN_ROLE, DEPOSITS_ENABLER_ROLE, DEPOSITS_DISABLER_ROLE, WITHDRAWALS_ENABLER_ROLE and WITHDRAWALS_DISABLER_ROLE roles in the L1LidoTokensBridge contract.

Emergency Brakes Multisig (0x73b047fe6337183a454c5217241d780a932777bd) is granted DEPOSITS_DISABLER_ROLE and WITHDRAWALS_DISABLER_ROLE in the L1LidoTokensBridge contract.

Lido Multisig (0xac8bc65814Dd0501674f6940aff1a4Ea78Fc20eF) is granted RATE_UPDATE_DISABLER_ROLE role in the TokenRateOracle contract and DEPOSITS_DISABLER_ROLE, WITHDRAWALS_DISABLER_ROLE roles in the L2ERC20ExtendedTokensBridge contract.

Optimism Bridge Executor (0x3b00f262e39372DF2756f809DD5DC36aeEdFC4A0) is granted DEFAULT_ADMIN_ROLE, RATE_UPDATE_ENABLER_ROLE, RATE_UPDATE_DISABLER_ROLE in the TokenRateOracle contract and DEFAULT_ADMIN_ROLE, DEPOSITS_ENABLER_ROLE, DEPOSITS_DISABLER_ROLE, WITHDRAWALS_ENABLER_ROLE, WITHDRAWALS_DISABLER_ROLE roles in the L2ERC20ExtendedTokensBridge contract.

Verification checklist

☑ Network specific behavior

All the network features affecting the protocol's operation are being studied. The virtual machine, the message transmission process within the main network, and vice versa (all distinctive network features and how they can impact the protocol's operation) are being researched. A comparison of the network's operation is conducted for deployment with networks where the wstETH token has already been deployed.

Results

Unichain is built on the OP stack with no significant differences, except for faster block times (it is stated that the block time will be 250ms, and currently, it is around 1 second). It has been verified that this assumption does not affect any part of the protocol — it won't cause `TokenRateData` delivery races, overwrites, or loss of data.

☑ Scope checking

This stage involves auditors researching the provided scope for verification, studying project dependencies, and building the protocol's architecture. Project documentation is examined. Existing tests are also run at this stage, and the test coverage level is checked. Contract mocks are investigated for logical errors. The protocol's architecture is examined for conceptual errors.

Results

The scope provided for verification was audited by the same team conducting the deployment verification. All necessary steps to enhance the security of the protocol were implemented during the audit, and no changes were made after the audit.

☑ Audit report investigation

At this stage, the presence of an audit report is verified, along with the alignment of the scope in the report with the deployed scope. It is checked whether all critical vulnerabilities have either been fixed or there is evidence that the vulnerability cannot be fixed without posing a threat to the protocol. Recommendations and the conclusion in the report are studied, as well as the alignment of the final commit with all the recommendations.

Results

The report was prepared by the team conducting the deployment verification, ensuring that it does not contain any unresolved issues. Most of the findings, especially severe ones, were addressed and correctly resolved. The protocol version that was audited was used for the deployment.

☑ Deploy script check

Auditors study the deployment script for contracts, examining initialization parameters. It is verified that interrupting the protocol deployment will not lead to incorrect initialization (for example, a front-run on initialization should result in both the script's reversion and require re-deployment).

Results

The deployment script available at this link correctly deploys all the necessary contracts and initializes the proxy in a manner that prevents front-running attacks.

☑ Deployment verification

The bytecode of the deployed contracts is checked to match the final commit in the report. An additional check is performed to verify all contracts on the explorer. Further verification is conducted to confirm that the bytecode of deployed contracts cannot be altered (https://mixbytes.io/blog/metamorphic-smart-contracts-is-evm-code-truly-immutable).

Results

The bytecode of the deployed contracts fully matches the audited version. All contracts were deployed from an EOA, eliminating the risk of metamorphic contracts.

☑ Initialization parameters check

At this stage, values are gathered from the storage in verified contracts, and they are checked for compliance with the parameters from the deployment script. Auditors ensure that all contracts are initialized and cannot be reinitialized by malicious users.

Results

All contracts have been correctly initialized, and all implementations are protected against reinitialization. All parameters used to configure the contracts and set their initial values are accurate.
An important detail about wstETH on the Unichain is that it is initialized to the version 2, even though it has never been upgraded on this network. This does not affect the contract's functionality and is considered reasonable since the code containing this initialization was previously audited.

☑ Role model verification

The protocol's access control structure is examined to identify redundant roles or roles with more privileges than intended. It is checked that all access rights are set by the previously studied structure. If a role is assigned to a multisig, multisig owners are validated.

Results

All roles have been granted correctly to the appropriate addresses. There are no unknown addresses used in the configured multisig. The Lido multisig deployed on Unichain has an unverified bytecode, but it is still possible to retrieve the signers of the multisig using the `getOwners` function signature. The multisig owners configured on Unichain are the same as those used for Gnosis Safe multisig on the Ethereum mainnet.

Social Links


https://mixbytes.io/


https://github.com/mixbytes/audits_public


hello@mixbytes.io


https://x/mixbytes