MixBytes()

stETH and wstETH on Soneium Deployment Verification Report

JANUARY 2024

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: Soneium

Scope:

Asset	Link	Comment
OpStackTokenRatePusher	0x927C99fC46226bd5131420B16aF0b0371165C3FC	Rate pusher for the Soneium network
OssifiableProxy	0x2F543A7C9cc80Cc2427c892B96263098d23ee55a	Ossifiable proxy for the L1LidoTokensBridge
L1LidoTokensBridge	0xf034dE8BD85A434d9Dc68F03382B589f86791425	Implementation of the L1LidoTokensBridge
OssifiableProxy	0xaA9BD8c957D803466FA92504BDd728cC140f8941	Ossifiable proxy for the ERC20BridgedPermit
ERC20BridgedPermit	0x7591f6BD2301f7EE92677380390540047b5B395B0	WstETH
OssifiableProxy	0x0Ce031AEd457C870D74914eCAA7971dd3176cDAF	Ossifiable proxy for the ERC20RebasableBridged Permit
ERC20RebasableBridgedPermit	0x3BC5d0551F48902bDcC036d59F5D23987F581c28	StETH
OssifiableProxy	0xDff6f372e8c16b2b9e95c55bDfe74C0bA3F90265	Ossifiable proxy for the TokenRateOracle
TokenRateOracle	0xA2f12f7C109c0b9aa5FFAe71612a68B6b8B2eFC4	Implementation of the TokenRateOracle

Asset	Link	Comment
OssifiableProxy	0xb4a0Cc7bE277DC9F9CBB6fbE8574B6f5221018D8	Ossifiable proxy for the L2ERC20ExtendedTokensBridge
L2ERC20ExtendedTokensBridge	0x3e2DcBe31617577d9CF934A9fb97DdC8FD844fa0	Implementation of the L2ERC20ExtendedTokensBridge
Governance Bridge Executor	0xB0F7894b3740F68eAca6e3792B14d2C2c25eF5D4	Governance executor for the soneium network

Audit reports:

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

Deployment scripts:
https://github.com/lidofinance/multichain-automaton/tree/04b16037e927f82a11ac5a8dffe9bf6644737484

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 Gnosis Safe Multisig (0x993F92e031B86b229D639463325f9d6a51609b43) 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 (0xB0F7894b3740F68eAca6e3792B14d2C2c25eF5D4) 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

☑ Framework-Based Testing

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

The Soneium Network is part of the Superchain Network, which is a network of Optimism forks. Therefore, it operates exactly the same as Optimism, a network for which deployment verification was previously conducted for this scope.
In the Soneium Network documentation, it is stated that there may be potential restrictions at the RPC level if an address violates intellectual property rules or is deemed "potentially harmful."

☑ 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. There are unnecessary files visible in the Soneium Network block explorer for the Governance Bridge Executor contract, which were appended after the source code verification.

☑ 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. There is a slight difference in the `TOKEN_RATE_OUTDATED_DELAY` parameter in the `TokenRateOracle` contract compared to previous deployments – it has been set to a higher value (93600) to account for potential delays in message delivery to L2.
An important detail about wstETH on the Soneium Network 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 and to the appropriate addresses. There are no unknown addresses used in the configured multisig. The addresses used in the multisig on the L2 side are the same as those on L1.

Social Links


https://mixbytes.io/


https://github.com/mixbytes/audits_public


hello@mixbytes.io


https://x/mixbytes