

# STATE MIND

**Lido V2 upgrade template**

**13-04-2023 – 10-05-2023**

# Table of contents



1. Project Brief		3
2. Finding Severity breakdown		4
3. Summary of findings		5
4. Conclusion		5
5. Findings report		6
Informational	Unused external function	6
	Redundant ACL check	6
	Unused function	6
	Function renaming	6
	Useless check	7
	No checks that the EOA deployer passed to the deployed contracts	7
	Not granted role	7
	Redundant local variable	8
	Gas efficient constants	8
	Missing dummyImplementation check	9
	Loop gas optimization	9
	EOA has access to change implementations before the startUpgrade	10
	Gas optimization	10
Not deployed dummy implementation	10	
6. Appendix A. Linter		11
7. Appendix B. Slither		13

# 1. Project Brief




Title	Description
Client	Lido
Project name	Lido V2 upgrade template
Timeline	13-04-2023 - 10-05-2023
Initial commit	8f9bfb2f0616fec031d382c4ec5e3455e7ebcd07
Final commit	a19c6b7e2d661de12e2ba585c251c8d70a1da230

## Short Overview

Lido V2 upgrade changes almost every previously deployed contract and adds new ones. From the prospects of the governance process, an on-chain vote for the upgrade should perform an atomic (all or nothing) transition. The upgrade template encompasses migrations, permissions granting procedures, and overall protocol integrity checks.

## Project Scope

The audit covered the following files:

 `ShapellaUpgradeTemplate.sol`

## 2. Finding Severity breakdown



All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds to be transferred to any party.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss of funds.
Informational	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

### 3. Summary of findings



Severity	# of Findings
Critical	0 (0 fixed, 0 acknowledged)
High	0 (0 fixed, 0 acknowledged)
Medium	0 (0 fixed, 0 acknowledged)
Informational	14 (7 fixed, 7 acknowledged)
Total	14 (7 fixed, 7 acknowledged)

### 4. Conclusion



#### Deployment

File name	Contract deployed on mainnet
ShapellaUpgradeTemplate.sol	<u><a href="#">0xa818fF9EC93122Bf9401ab4340C42De638CD600a</a></u>

## 5. Findings report



INFORMATIONAL-01	Unused external function	Fixed at <a href="#">5651a8</a>
<p><b>Description</b></p> <p>The <code>upgrade_shapella_2_revoke_roles.py</code> upgrade script's <u>comments say upgrade finish should be checked by <code>assertUpgradelsFinishedCorrectly()</code></u>. Although the script calls only <code>revertIfUpgradeNotFinished</code> while <u><code>assertUpgradelsFinishedCorrectly</code></u> function is not used anywhere except in tests.</p> <p>If <code>revertIfUpgradeNotFinished</code> does not revert, then the upgrade is finished (<code>_isUpgradeFinished == True</code>). <code>assertUpgradelsFinishedCorrectly</code> may revert if one of the checked parameters is altered even if the upgrade is finished.</p> <p><b>Recommendation</b></p> <p>We recommend removing unused external functions.</p>		
INFORMATIONAL-02	Redundant ACL check	Fixed at <a href="#">5651a8</a>
<p><b>Description</b></p> <p>During <u><code>_assertInitialACL</code></u> and <u><code>_assertFinalACL</code></u>, we check if <code>StakingRouter</code> has the necessary roles. However, the <code>STAKING_MODULE_RESUME_ROLE</code> role is never used in the context of an update (unlike other <code>RESUME_ROLE</code> roles for <code>WithdrawalQueue</code> and <code>ValidatorsExitBusOracle</code>).</p> <p><b>Recommendation</b></p> <p>We recommend removing the redundant check.</p>		
INFORMATIONAL-03	Unused function	Fixed at <a href="#">5651a8</a>
<p><b>Description</b></p> <p>The <u><code>_assertLocatorAddresses</code></u> function is defined but never used.</p> <p><b>Recommendation</b></p> <p>We recommend adding the <code>_assertLocatorAddresses</code> invocation during <code>_startUpgrade</code> function.</p>		
INFORMATIONAL-04	Function renaming	Fixed at <a href="#">5651a8</a>
<p><b>Description</b></p> <p>Name of function <u><code>_calcInitialEpochForAccountingOracleHashConsensus()</code></u> is confusing because it is not only used for updating <code>HashConsensus</code> for <code>Accounting Oracle</code>, but also for <code>Validators Exit Bus Oracle</code>.</p> <p><b>Recommendation</b></p> <p>It is recommended to rename function.</p>		

INFORMATIONAL-05	Useless check	Fixed at <a href="#">5651a8</a>
------------------	---------------	---------------------------------

Description

In the function `_assertInitialACL` there is a call of the `_assertAdminsOfProxies` function in which there are checks of proxies and their's admin addresses.

It is useless because in the function `_startUpgrade` before a call of the function `_assertInitialACL` there is a call to the function `_upgradeProxyImplementations` which updates implementations of the proxies. Implementations updates will fail if the admin of these proxies won't be this contract.

The function `_assertAdminsOfProxies` will pass all checks if the function `_upgradeProxyImplementations` is passed.

Recommendation

Call the `_assertAdminsOfProxies` function before the call of the `_upgradeProxyImplementations` function.

INFORMATIONAL-06	No checks that the EOA deployer passed to the deployed contracts	Fixed at <a href="#">5651a8</a>
------------------	--	---------------------------------

Description

For some contracts there are not enough checks on parameters that EOA deployer has passed to the contract. In some cases if this variables aren't checked, the EOA deployer can exploit the LIDO contracts or break them.

- The Burner contract
  - No checks of the roles `RECOVER_ASSETS_ROLE` and `REQUEST_BURN_MY_STETH_ROLE`
  - No checks of the variables `STETH`, `TREASURY`, `totalCoverSharesBurnt`, and `totalNonCoverSharesBurnt`
- In the HashConsensus contract for the AccountingOracle and ValidatorsExitBusOracle contracts
  - No checks of the roles `MANAGE_MEMBERS_AND_QUORUM_ROLE`, `DISABLE_CONSENSUS_ROLE`, `MANAGE_FRAME_CONFIG_ROLE`, `MANAGE_FAST_LANE_CONFIG_ROLE`, `MANAGE_REPORT_PROCESSOR_ROLE`
  - No checks that the EOA deployer didn't add new members and in the contract there are zero members
  - There is no check of the getChainConfig and getFrameConfig functions
- In the contract DepositSecurityModule
  - No check of the `guardians` array.
  - No checks of the `LIDO`, `STAKING_ROUTER`, and `DEPOSIT_CONTRACT` addresses

Recommendation

All appropriate checks should be added for this global variables.

Client's comments

Some of the checks were added, while the lack of others is acknowledged. The risks for the acknowledged ones are mitigated by irreversibly passing ownership early and having a dedicated flow for the pre-vote tests and verifications.

- The Burner contract
  - `REQUEST_BURN_MY_STETH_ROLE` check added
  - no check for `RECOVER_ASSETS_ROLE` needed because it was removed

INFORMATIONAL-07	Not granted role	Fixed at <a href="#">5651a8</a>
------------------	------------------	---------------------------------

Description

`StakingRouter`'s role `STAKING_MODULE_RESUME_ROLE` isn't granted to any contract during the upgrade process. It is used for access to `StakingRouter.resumeStakingModule`, which is called inside `DepositSecurityModule.unpauseDeposits`. So `DepositSecurityModule` should get this role.

Recommendation

It is recommended granting `StakingRouter.STAKING_MODULE_RESUME_ROLE` to `DepositSecurityModule`.

INFORMATIONAL-08	Redundant local variable	Acknowledged
------------------	--------------------------	--------------

**Description**

At function `_assertFinalACL()` local variable `agent` created, but `_agent` is constant. It's more gas efficient to use this constant

**Recommendation**

We recommend to change passed arguments to `_agent`. As example:

```

...
_assertAdminsOfProxies(_agent);

if (_depositSecurityModule.getOwner() != _agent) revert IncorrectDsmOwner();
...

```

**Client's comments**

It’s done on purpose to reduce the contract’s bytecode size at the cost of gas. With the local agent variable it is 73 bytes less.

INFORMATIONAL-09	Gas efficient constants	Acknowledged
------------------	-------------------------	--------------

**Description**

All constants such as at Line 363 can be more gas efficient for deployment and contract bytecode size. Setting them in private won't generate function with same name for reading this constant.

**Recommendation**

We recommend setting constants as private instead of public.

**Client's comments**

This is done on purpose to simplify verification of the deployed template, but some of the constants made private to save bytecode size.



INFORMATIONAL-10	Missing dummyImplementation check	Acknowledged
------------------	-----------------------------------	--------------

### Description

At function `_assertInitialProxyImplementations()` we check 4 new proxies. But, there is **LidoLocator** and in `_upgradeProxyImplementations()` we upgrade all new proxies.

### Recommendation

We recommend add dummyImplementation check for **LidoLacator**

```
function _assertInitialProxyImplementations() internal view {
    if (_withdrawalVault.implementation() != _withdrawalVaultImplementation)
        revert IncorrectInitialImplementation(address(_withdrawalVault));
    _assertInitialDummyImplementation(_locator);
    _assertInitialDummyImplementation(_accountingOracle);
    _assertInitialDummyImplementation(_stakingRouter);
    _assertInitialDummyImplementation(_validatorsExitBusOracle);
    _assertInitialDummyImplementation(_withdrawalQueue);
}
```

or add new comment.

### Client's comments

Done on purpose. To allow the template work in tests with either dummy or some “debug” implementation. The debug implementation might be set to debug oracle daemons. Before the upgrade aragon voting start the implementation is planned to be set back to dummy.

INFORMATIONAL-11	Loop gas optimization	Acknowledged
------------------	-----------------------	--------------

### Description

The internal `_migrateLidoOracleCommitteeMembers` function could be optimized by reducing the number of loops

### Recommendation

We recommend using one loop.

```
hcForAO.grantRole(manage_members_role, address(this));
hcForVEBO.grantRole(manage_members_role, address(this));

for (uint256 i; i < members.length; ++i) {
    hcForAO.addMember(members[i], quorum);
    hcForVEBO.addMember(members[i], quorum);
}

hcForAO.renounceRole(manage_members_role, address(this));
hcForVEBO.renounceRole(manage_members_role, address(this));
```

### Client's comments

Done on purpose to group events related to the same HachConsensus contracts. It is used in the checks for emmitted events in test\_upgrade\_shapella.py

INFORMATIONAL-12	EOA has access to change implementations before the startUpgrade	Acknowledged
------------------	--	--------------

**Description**

The LIDO team should consider that the EOA deployer has access to set any implementation to the proxies. For example he can set some malicious implementation that sets certain values to certain slots and then upgrades proxies to a dummy implementation. This values can be used by malicious EOA deployer after the upgrades finishes.

**Recommendation**

Before the upgrade the LIDO team and voters should check that EOA deployer has deployed the proxies correctly and that there weren't any other implementations except the dummy implementation.

**Client's comments**

Absence of malicious EOA deployer actions is planned to be checked after irreversibly passing EOA deployer rights to the template. Propability of such malicious actions reduced by passing EAO deployer rights to the template right after protocol contracts deployment, reducing the time gap.

INFORMATIONAL-13	Gas optimization	Acknowledged
------------------	------------------	--------------

**Description**

In the function **\_assertLocatorAddresses** there are calls of the **LidoLocator** contract. It will be cheaper to call the **\_locatorImplementation** address directly. It can be done because all of the checked variables are immutable.

**Recommendation**

Use call to the implementation directly.

**Client's comments**

We'd like to keep it less gas-optimized but more straightforward.

INFORMATIONAL-14	Not deployed dummy implementation	Acknowledged
------------------	-----------------------------------	--------------

**Description**

A **dummy implementation** is not planned for deployment due to a lack of need. However, it is better to deploy an empty contract to avoid the problem of subsequent deployment of malicious code to the specified address.

**Recommendation**

It is recommended deploying a dummy implementation contract.

**Client's comments**

It was planned for the deployment along with the other protocol contracts.

## 6. Appendix A. Linter



### Error/max-line-length

- ShapellaUpgradeTemplate.sol:73 – Line length must be no more than 120 but current length is 128.
- ShapellaUpgradeTemplate.sol:209 – Line length must be no more than 120 but current length is 132.
- ShapellaUpgradeTemplate.sol:316 – Line length must be no more than 120 but current length is 133.
- ShapellaUpgradeTemplate.sol:352 – Line length must be no more than 120 but current length is 126.
- ShapellaUpgradeTemplate.sol:366 – Line length must be no more than 120 but current length is 141.
- ShapellaUpgradeTemplate.sol:371 – Line length must be no more than 120 but current length is 135.
- ShapellaUpgradeTemplate.sol:375 – Line length must be no more than 120 but current length is 130.
- ShapellaUpgradeTemplate.sol:376 – Line length must be no more than 120 but current length is 137.
- ShapellaUpgradeTemplate.sol:377 – Line length must be no more than 120 but current length is 126.
- ShapellaUpgradeTemplate.sol:378 – Line length must be no more than 120 but current length is 147.
- ShapellaUpgradeTemplate.sol:383 – Line length must be no more than 120 but current length is 132.
- ShapellaUpgradeTemplate.sol:384 – Line length must be no more than 120 but current length is 123.
- ShapellaUpgradeTemplate.sol:390 – Line length must be no more than 120 but current length is 135.
- ShapellaUpgradeTemplate.sol:467 – Line length must be no more than 120 but current length is 125.
- ShapellaUpgradeTemplate.sol:469 – Line length must be no more than 120 but current length is 129.
- ShapellaUpgradeTemplate.sol:600 – Line length must be no more than 120 but current length is 121.
- ShapellaUpgradeTemplate.sol:693 – Line length must be no more than 120 but current length is 126.
- ShapellaUpgradeTemplate.sol:694 – Line length must be no more than 120 but current length is 130.
- ShapellaUpgradeTemplate.sol:704 – Line length must be no more than 120 but current length is 146.
- ShapellaUpgradeTemplate.sol:712 – Line length must be no more than 120 but current length is 124.
- ShapellaUpgradeTemplate.sol:721 – Line length must be no more than 120 but current length is 127.
- ShapellaUpgradeTemplate.sol:729 – Line length must be no more than 120 but current length is 143.
- ShapellaUpgradeTemplate.sol:753 – Line length must be no more than 120 but current length is 126.

- ShapellaUpgradeTemplate.sol:908 – Line length must be no more than 120 but current length is 125.
- ShapellaUpgradeTemplate.sol:944 – Line length must be no more than 120 but current length is 124.
- ShapellaUpgradeTemplate.sol:1009 – Line length must be no more than 120 but current length is 127.

## 7. Appendix B. Slither



### Medium/High/incorrect-equality

ShapellaUpgradeTemplate.\_assertUpgradelsFinishedCorrectly() uses a dangerous strict equality: - \_upgradeBlockNumber == UPGRADE\_NOT\_STARTED

ShapellaUpgradeTemplate.\_finishUpgrade() uses a dangerous strict equality: - \_upgradeBlockNumber == UPGRADE\_NOT\_STARTED

### Medium/Medium/uninitialized-local

ShapellaUpgradeTemplate.\_migrateLidoOracleCommitteeMembers().i is a local variable never initialized

ShapellaUpgradeTemplate.\_migrateLidoOracleCommitteeMembers().i\_scope\_0 is a local variable never initialized

# STATE MIND