STATE MIND

EasyTrack: Simple DVT factories

11-12-2023 - 24-12-2023

Table of contents

1. Project Brief

Title	Description
Client	Lido
Project name	EasyTrack: Simple DVT factories
Timeline	11-12-2023 - 24-12-2023
Initial commit	8e1c2aa5dbfcdf18c2511badd28dbe07daa6be43
Final commit	bccc99912b8cd03cb152ebc13295cc3a3ea28664

Short Overview

This bench of Easy Track factories provides the ability to manage Node Operators Registry with committee multisig.

- ActivateNodeOperators – creates EVMScript to activate several node operators
- AddNodeOperators – creates EVMScript to add a new batch of node operators
- DeactivateNodeOperators – creates EVMScript to deactivate several node operators
- ChangeNodeOperatorManagers – creates EVMScript to change signing keys manager for several node operators
- IncreaseVettedValidatorsLimit – creates EVMScript to increase the staking limit for a node operator
- SetNodeOperatorNames – creates EVMScript to set the name of several node operators
- SetNodeOperatorRewardAddresses – creates EVMScript to set the reward address of several node operators
- SetVettedValidatorsLimits – creates EVMScript to set the staking limit for node operators
- UpdateTargetValidatorLimits – creates EVMScript to set the node operator's target validators limit

Project Scope

The audit covered the following files:

- ActivateNodeOperators.sol
- AddNodeOperators.sol
- ChangeNodeOperatorManagers.sol
- DeactivateNodeOperators.sol
- IncreaseVettedValidatorsLimit.sol
- SetNodeOperatorNames.sol
- SetNodeOperatorRewardAddresses.sol
- SetVettedValidatorsLimits.sol
- UpdateTargetValidatorLimits.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	10 (7 fixed, 3 acknowledged)
Total	10 (7 fixed, 3 acknowledged)

4. Conclusion

During the audit of the codebase, 10 issues were found in total:

- 10 informational severity issues (7 fixed, 3 acknowledged)

The final reviewed commit is bccc99912b8cd03cb152ebc13295cc3a3ea28664

Deployment

Contract	Address
AddNodeOperators	0xcAa3AF7460E83E665EEFeC73a7a542E5005C9639
ActivateNodeOperators	0xCBb418F6f9BFd3525CE6aADe8F74ECFEfe2DB5C8
ChangeNodeOperatorManagers	0xE31A0599A6772BCf9b2bFc9e25cf941e793c9a7D
DeactivateNodeOperators	0x8B82C1546D47330335a48406cc3a50Da732672E7
SetVettedValidatorsLimits	0xD75778b855886Fc5e1eA7D6bFADA9EB68b35C19D
IncreaseVettedValidatorsLimit	0xcc993499E03DdA45ae8804AA1620257A1d7FB996

SetNodeOperatorNames	0x7d509BFF310d9460b1F613e4e40d342201a83Ae4
SetNodeOperatorRewardAddresses	0x589e298964b9181D9938B84bB034C3BB9024E2C0
UpdateTargetValidatorLimits	0x41CF3DbDc939c5115823Fba1432c4EC5E7bD226C

5. Findings report

INFORMATIONAL-01	ChangeNodeOperatorManagers redundant memory allocation	Fixed at f6b982

Description

Lines:

- ChangeNodeOperatorManagers.sol#L88
- ChangeNodeOperatorManagers.sol#L100

ChangeNodeOperatorManagers contract uses only the **acl** address to create the script, so the **toAddresses** array always contains the same addresses. **EVMScriptCreator** library contains a function for encoding multiple calls to different methods within the same contract EVMScriptCreator.sol#L50-L54

Impact: increased gas consumption

Recommendation

We recommend removing the **toAddresses** array and using the **EVMScriptCreator.createEVMScript(address,bytes4[],bytes[])** function.

INFORMATIONAL-02	Duplication of INodeOperatorsRegistry interface	Fixed at c206ea

Description

Line: IncreaseVettedValidatorsLimit.sol#L9-L33

EasyTrack repository has an interface INodeOperatorRegestry.sol, but **IncreaseVettedValidatorsLimit.sol** also has a copy of **INodeOperatorRegestry.sol**.

Recommendation

We recommend using a single **INodeOperatorsRegistry** interface in contracts.

INFORMATIONAL-03	IncreaseVettedValidatorsLimit can temporarily block the addition of a new motion to EasyTrack	Acknowledged

Description

IncreaseVettedValidatorsLimit contract allows the **manager/reward** address to increase the vetted limit on an arbitrary value of **_stakingLimit**. Usually, the limit is increased by tens or hundreds of validator keys using a single motion Motion#549, Motion#548. In general, the node operator is economically interested in increasing the limit agreed with the Lido committee, and in case of incorrect behavior, it can be excluded from the operator pool using the DAO, and redundant motions can be canceled.

However, the **EasyTrack** contract doesn't limit the **_stakingLimit** parameter. Thus, in case of compromise of the **manager/reward** address or the addition of a new operator who acts maliciously, the node operator can create many motions with an increase in the limit by 1 key and can fill the entire EasyTrack queue, which has a limit of 12/24 parallel motions.

An attacker will not be able to gain economic benefits directly, however, it can use financial instruments to benefit from potential **LDO** price changes after creating an emergency.

Recommendation

We recommend considering the possibility of adding a register that stores a schedule for increasing limits for each operator and is coordinated by the Lido committee.

Client's comments

This script factory will not be used in the current installation, but it has been implemented to be consistent with the current Node Operators Registry workflow.

INFORMATIONAL-04	Gas optimizations: cycles	Fixed at a3edfe

Description

Lines:

1. **ActivateNodeOperatos.sol#L86**
2. **ActivateNodeOperatos.sol#L135**
3. **ActivateNodeOperatos.sol#L150**
4. **AddNodeOperators.sol#L89**
5. **AddNodeOperators.sol#L163**
6. **AddNodeOperators.sol#L167**
7. **ChangeNodeOperatorManagers.sol#L88**
8. **ChangeNodeOperatorManagers.sol#L142**
9. **ChangeNodeOperatorManagers.sol#L150**
10. **DeactivateNodeOperators.sol#L83**
11. **DeactivateNodeOperators.sol#L129**
12. **SetNodeOperatorNames.sol#L64**
13. **SetNodeOperatorNames.sol#L108**
14. **SetNodeOperatorRewardAddresses.sol#L67**
15. **SetNodeOperatorRewardAddresses.sol#L111**
16. **SetVettedValidatirsLimits.sol#L65**
17. **SetVettedValidatirsLimits.sol#L107**
18. **UpdateTargetValidatorLimits.sol#L74**
19. **UpdateTargetValidatorLimits.sol#L113**

There are places with cycles, but they can consume less gas.
Instead of:

```
for (uint256 i = 0; i < _decodedCallData.length; i++) {
    ...
}
```

Make:

```
uint256 calldataLength = _decodedCallData.length;

for (uint256 i; i < calldataLength;) {
    ...

    unchecked {
        ++i;
    }
}
```

or less optimized:

```
...

for (uint256 i; i < calldataLength; ++i) {
    ...
}
```

Recommendation

We recommend replacing these parts of the code.

INFORMATIONAL-05	Gas optimizations: external calls	Fixed at bccc99

Description

Lines:

1. **SetNodeOperatorRewardAddresses.sol#L133**
2. **SetNodeOperatorRewardAddresses.sol#L135**
3. **SetNodeOperatorNames.sol#L132**
4. **SetNodeOperatorNames.sol#L133**

There are some external calls which don't provide any additional information.

There's an external call to get the address of **lido()** every time, but this address can be set during construction only once.

Recommendation

We recommend replacing these parts of the code.

INFORMATIONAL-06	Typos in comments/names	Fixed at 176dc3

Description

- **Argon** instead of **Aragon**.
- **caldataLength** instead of **calldataLength**.
- **INodeOperatorRegestry.sol** instead of **INodeOperatorRegistry.sol**.
- Possible invalid error name **here** and **here** in **ChangeNodeOperatorManagers.sol** contract.

Recommendation

We recommend fixing these typos.

INFORMATIONAL-07	Not used parts of code	Fixed at c9015a

Description

There's an error **ERROR_REWARD_ADDRESSES_HAS_DUPLICATE**, which is not used.

Recommendation

We recommend deleting this error.

INFORMATIONAL-08	Insufficient checks	Fixed at 0ebb5e

Description

- There're no checks inside **SetVettedValidators.sol** that **NodeOperator** is **active** and **_decodedCallData[i].stakingLimit** is greater than vetted before.
- There's no check inside **AddNodeOperators.sol** that **length** of the array is greater than zero.

Recommendation

We recommend adding these checks.

Client's comments

> There is no check that **_decodedCallData[i].stakingLimit** is greater than vetted before, because the module management committee should be able to do so by design.

INFORMATIONAL-09	General architectural guideline	Acknowledged

Description

EasyTrack contract can be given a **STAKING_MODULE_MANAGE_ROLE** role instead of **STAKING_ROUTER_ROLE**. It allows us to call the **updateTargetValidatorsLimits** function via **StakingRouter.sol**. It may resolve some problems, however, this decision also carries certain risks.

Recommendation

We suggest considering a possible alternative.

Client's comments

> **STAKING_MODULE_MANAGE_ROLE** role is not applicable in this case because this role is too general and gives rights to control all protocol modules. Solving this problem requires changes outside the scope of this project.

INFORMATIONAL-10	Inconsistent type of stakingLimit	Acknowledged

Description

Lines:

- IncreaseVettedValidatorsLimit.sol#L39
- SetVettedValidatorsLimits.sol#L15

The given contracts have a **stakingLimit** variable as an input, and it has a **uint256** type. However, the **NodeOperatorsRegistry** contract gets it as a **uint64** type. Currently, it doesn't result in any problems, but it leaves an inconsistency.

Recommendation

We recommend changing the type of the input parameter **stakingLimit** to **uint64**.

Client's comments

> **uint64** type in Node Operators Registry was used for backwards compatibility with the previous version. Currently, we use the default types for external interfaces, where it is possible

7. Appendix B. Slither

Informational/High/boolean-equal

DeactivateNodeOperators._validateInputData(DeactivateNodeOperators.DeactivateNodeOperatorInput[]) compares to a boolean constant: -require(bool,string)(nodeOperatorsRegistry.getNodeOperatorIsActive(_decodedCallData[i].nodeOperatorId) == true,ERROR_WRONG_OPERATOR_ACTIVE_STATE)

Medium/Medium/unused-return

SetNodeOperatorNames._validateInputData(SetNodeOperatorNames.SetNameInput[]) ignores return value by nodeOperatorsRegistry.getNodeOperator(_decodedCallData[i].nodeOperatorId,true)

SetNodeOperatorRewardAddresses._validateInputData(SetNodeOperatorRewardAddresses.SetRewardAddressInput[]) ignores return value by nodeOperatorsRegistry.getNodeOperator(_decodedCallData[i].nodeOperatorId,false)

SetNodeOperatorNames._validateInputData(SetNodeOperatorNames.SetNameInput[]) ignores return value by nodeOperatorsRegistry.getNodeOperator(_decodedCallData[i].nodeOperatorId,true)

SetNodeOperatorRewardAddresses._validateInputData(SetNodeOperatorRewardAddresses.SetRewardAddressInput[]) ignores return value by nodeOperatorsRegistry.getNodeOperator(_decodedCallData[i].nodeOperatorId,false)

8. Appendix C. Tests

Tests result

1 failed, 471 passed, 1 skipped, 38974 warnings in 1535.67s (0:25:35)

STATE
MIND