Quantstamp

SSV Network - Diff 2

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	Registry and Payment
Timeline	2024-01-29 through 2024-02-02
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	• bloxapp/ssv-network #f51d071
Auditors	• Cameron Biniamow Auditing Engineer • Jennifer Wu Auditing Engineer • Roman Rohleder Senior Auditing Engineer

Documentation quality	High
Test quality	High
Total Findings	5 Fixed: 1 Acknowledged: 2 Mitigated: 2
High severity findings	1 Fixed: 1
Medium severity findings	1 Acknowledged: 1
Low severity findings	0
Undetermined severity findings	0
Informational findings	3 Acknowledged: 1 Mitigated: 2

Summary of Findings

This audit report is a second **diff** audit of the SSV.network contracts, specifically highlighting the changes made between **v1.0.2 and v1.1.0-bulks**. It is crucial for readers to review this diff audit alongside the initial SSV.network audit report and the first diff SSV.network audit report, as the scope of this diff audit is strictly limited to changes between v1.0.2 and v1.1.0-bulks. Between v1.0.2 and v1.1.0-bulks, the contracts underwent minor refactorings and the addition of bulk validator registration, removal, and exiting.

The diff audit resulted in five findings: one high, one medium, one informational, and two gas optimizations. The gas optimization issues are suggestions to improve gas efficiency without major refactoring. We recommend the client to consider all identified issues.

Fix Review: Following the initial audit, the SSV team implemented fixes for the sole high-severity issue, SSV-1. All remaining issues have been acknowledged or mitigated.

ID	DESCRIPTION	SEVERITY	STATUS
SSV-1	**Manipulate the Cluster Validator Count to Pay Operator Less**	High	Fixed
SSV-2	**Remove Validator Before Liquidation to Avoid Deactivation**	Medium	Acknowledged
SSV-3	**Gas Optimization: Cache Variables**	Informational	Mitigated
SSV-4	**Gas Optimization: Delete Validator and Emit `ValidatorRemoved` in Redundant `for` Loops**	Informational	Mitigated
SSV-5	**Inconsistent Use of `ISSVNetworkCore.Cluster`**	Informational	Acknowledged

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

> **Disclaimer**
>
> Please note that the audit scope is limited to the changes between v1.0.2 and v1.1.0-bulks for the smart contracts supporting the registry and payment distribution between validators and operators. As a result, the following areas of concern are considered out of scope:
>
> - Malicious operators: This refers to the risk of operators working together to manipulate the consensus process in their favor, which could lead to a validator being slashed for behaving dishonestly.
> - Private key compromise: This risk involves the possibility of an attacker reconstructing a validator's private key from shares, which could allow them to access the validator.
> - Idle validator slashing: This risk involves idle operators in the consensus process, which could result in validators losing out on block proposals and attestation rewards.
> - Validator unstaking after the Shanghai fork: This risk refers to the possibility that validators may unstake their funds following the Shanghai fork, which could result in the potential incompatibility of the SSV network.
> - SSV Cli key generation: The registry relies on off-chain mechanisms to handle the generation of key shares for operators.
>
> The integration of these contracts with the remainder of the system was not subject to auditing.
>
> Only features contained within the repositories at the commit hashes specified on the report's front page are within the audit and fix review scope. All features added in future code revisions are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
    1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
    2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
    1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

Files Included

- `contracts/SSVNetwork.sol`
- `contracts/interfaces/ISSVClusters.sol`
- `contracts/interfaces/ISSVNetworkCore.sol`
- `contracts/libraries/ClusterLib.sol`
- `contracts/libraries/CoreLib.sol`
- `contracts/libraries/OperatorLib.sol`
- `contracts/libraries/ValidatorLib.sol`
- `contracts/modules/SSVClusters.sol`

Findings

SSV-1 Manipulate the Cluster Validator Count to Pay Operator Less

High Fixed

Update

The client fixed the issue in `a58162e83a3713ecb5db7293719283ab67ecdc96`. The function `bulkRemoveValidator()` now indirectly ensures the uniqueness of the validator public keys by iterating over the validators to remove and verifying the validator's state before deleting the validator. Therefore, if `publicKeys` contains duplicates, the error `IncorrectValidatorStateWithData` will occur.

File(s) affected: `SSVClusters.sol`

Description: The `bulkRemoveValidator()` function is designed to remove validators from the SSV Network for a given cluster of operator IDs. However, it lacks duplicate public key checks, allowing a cluster owner to inaccurately reduce `cluster.validatorCount` by submitting duplicate removal requests for the same validator. This manipulation results in incorrect balance calculations, leading to potential underpayment to operators and discrepancies in off-chain components that rely on accurate `ValidatorRemoved` events.

Exploit Scenario:

1. A cluster owner registers multiple unique validators within the same cluster of operator IDs.
2. The owner invokes `bulkRemoveValidator()` with an array containing duplicate public keys for the same validator.
3. The `cluster.validatorCount` is reduced more than the actual count of unique validators removed, leading to an incorrect balance.
4. Off-chain components monitoring `ValidatorRemoved` events process multiple removal events for the same validator, removing only one instance for the associated public key.
5. Operators are under-compensated based on the inaccurately reduced `cluster.validatorCount`.

Recommendation: Implement checks within the `bulkRemoveValidator()` function to prevent duplicate public keys while removing validators.

SSV-2
Remove Validator Before Liquidation to Avoid Deactivation

Medium Acknowledged

Update

The client acknowledged the issue and provided the following explanation:

> This finding has been thoroughly reviewed by the SSV team. A key safeguard against the risks outlined is the network's liquidator component, which continuously monitors clusters for liquidation eligibility. This mechanism ensures that irrespective of updates to a cluster's status, any cluster meeting the criteria for liquidation will be addressed promptly by the liquidator. The design of the liquidation threshold is intended to balance the needs of cluster owners with the operational requirements of the protocol. The liquidation process is incentivized to ensure prompt action when thresholds are breached, thereby preserving the financial integrity of the protocol. This system ensures that the protocol remains financially stable, whether or not cluster owners manage to replenish funds in time. After careful consideration of the recommendations presented by Quantstamp, the SSV team believes that the current implementation effectively upholds the financial stability of the protocol. Our analysis concludes that the existing mechanisms adequately mitigate the risks of insolvency, and we remain committed to continuous monitoring and improvement to safeguard the network's financial health.

File(s) affected: `SSVClusters.sol`

Description: The SSV protocol relies on timely liquidation to maintain sufficient SSV deposits to cover unclaimed network and operator fees. However, if a cluster's SSV balance drops below the liquidation threshold and liquidation is delayed, operators risk being unable to withdraw their earnings due to an insufficient SSV balance in the SSVNetwork. Furthermore, the cluster may remain insolvent upon reactivation, with a mismatch between the operators' and the cluster's balances. This issue was previously noted as "Delayed Liquidation Results in Insufficient SSV Balance", emphasizing the importance of prompt liquidation. The `updateBalance()` function presents a variation of this issue. It allows for removing validators from the protocol, even if the cluster's balance is insufficient to pay the operators. It sets the cluster balance to zero if usage exceeds the current balance, potentially masking insolvency issues. Moreover, the `removeValidator()` function permits validator removal without considering the cluster's liquidation status, potentially bypassing the liquidation process.

Recommendation: The client should consider investigating to understand the current risks and potential for abuse inherent in the design of the `removeValidator()` and `updateBalance()` functions. The client can consider analyzing if validators are currently exploiting this loophole to bypass the liquidation process and assess the timeliness and effectiveness of liquidation enforcement. This analysis can help to gauge the protocol's exposure to insolvency risks. Based on the findings, the client may need to consider revising the protocol's design or implementing additional monitoring and control mechanisms to mitigate these risks and safeguard the protocol's financial stability.

SSV-3 Gas Optimization: Cache Variables

Informational Mitigated

Update

The client mitigated the issue in `a58162e83a3713ecb5db7293719283ab67ecdc96` and provided the following explanation:

> The functions ValidatorLib.registerPublicKeys() and ValidatorLib.validateStates() were removed.

1. Issue is deprecated since `ValidatorLib.registerPublicKeys()` was removed.
2. Issue is deprecated since `ValidatorLib.validateStates()` was removed.
3. Unresolved `publicKeys.length` should be cached before the first `if` statement in `SSVClusters.bulkRegisterValidator()`.
4. Unresolved `publicKeys.length` remains uncached in `SSVClusters.bulkExitValidator()`.

File(s) affected: `ValidatorLib.sol, SSVClusters.sol`

Description: Repeatedly accessing certain variables can be gas-intensive. To optimize gas usage, values frequently accessed should be stored in memory variables. Consider storing the value in a memory variable and reference the memory variable for the following variables:

1. `publicKeys.length` from `ValidatorLib.registerPublicKeys()`
2. `publicKeys.length` from `ValidatorLib.validateStates()`
3. `publicKeys.length` from `SSVClusters.bulkRegisterValidator()`
4. `publicKeys.length` from `SSVClusters.bulkExitValidator()`

Recommendation: Consider applying variable caching as per the recommendation provided in the issue.

SSV-4
Gas Optimization: Delete Validator and Emit `ValidatorRemoved` in Redundant `for` Loops

Informational · Mitigated

Update

While some refactoring has been done to function `bulkRemoveValidator()`, the two mentioned loops still exist.

File(s) affected: `SSVClusters.sol`

Description: In the function `SSVClusters.bulkRemoveValidator()`, there are two `for` loops that iterate over the same range. The first `for` loop deletes each validator from the `validatorPKs` mapping. The second `for` loop emits a `ValidatorRemoved` event for each deleted validator. Performing both operations in separate `for` loops increases gas costs.

Recommendation: Merge the logic from both `for` loops so that the validator is deleted from the `validatorPKs` mapping and the `ValidatorRemoved` event is emitted in the same `for` loop.

SSV-5 Inconsistent Use of `ISSVNetworkCore.Cluster`

Informational · Acknowledged

Update

The client acknowledged the issue and provided the following explanation:

> We acknowledged this finding and plan to revisit all imports in a future release.

File(s) affected: `SSVNetwork.sol`

Description: The function `SSVNetwork.bulkRemoveValidator()` declares the type for the parameter `cluster` as `Cluster` whereas all other instances throughout the contract use `ISSVNetworkCore.Cluster`. The inconsistent usage of `ISSVNetworkCore.Cluster` and `Cluster` decreases code readability and could lead to confusion.

Recommendation: Change `Cluster` to `ISSVNetworkCore.Cluster`.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.

- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Code Documentation

1. Fixed Correct the function notice definition for `ISSVClusters.bulkRemoveValidator()`.
2. Fixed Correct the function notice definition for `ISSVClusters.removeValidator()`.
3. Fixed The function `ValidatorLib.resgisterPublickey()` is misspelled. The naming and its reference throughout the codebase should be corrected.
4. Fixed Correct the `Cluster` struct indentation in `ISSVNetworkCore.sol`.

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

- `026...a3c ./contracts/SSVNetwork.sol`
- `e31...b2b ./contracts/SSVNetworkViews.sol`
- `249...f2e ./contracts/SSVProxy.sol`
- `001...894 ./contracts/interfaces/ISSVOperators.sol`
- `906...40e ./contracts/interfaces/ISSVNetwork.sol`
- `f40...e05 ./contracts/interfaces/ISSVViews.sol`
- `014...6dc ./contracts/interfaces/ISSVDAO.sol`
- `7c9...928 ./contracts/interfaces/ISSVClusters.sol`
- `993...b85 ./contracts/interfaces/ISSVNetworkCore.sol`
- `536...d02 ./contracts/modules/SSVOperators.sol`
- `7b7...592 ./contracts/modules/SSVViews.sol`
- `8b6...bc7 ./contracts/modules/SSVClusters.sol`
- `838...f42 ./contracts/modules/SSVDAO.sol`
- `237...e14 ./contracts/libraries/SSVStorage.sol`
- `7ea...f65 ./contracts/libraries/SSVStorageProtocol.sol`
- `815...c4e ./contracts/libraries/ValidatorLib.sol`
- `fdf...7d5 ./contracts/libraries/OperatorLib.sol`
- `a0e...aac ./contracts/libraries/CoreLib.sol`
- `fa2...e38 ./contracts/libraries/ClusterLib.sol`
- `cef...bc5 ./contracts/libraries/ProtocolLib.sol`
- `2ad...ade ./contracts/libraries/Types.sol`

Tests

- `1eb...316 ./test/helpers/utils.ts`
- `28a...c07 ./test/helpers/gas-usage.ts`
- `90e...b73 ./test/helpers/types.ts`
- `6c4...8e5 ./test/helpers/contract-helpers.ts`
- `386...1bb ./test/account/deposit.ts`
- `8ca...731 ./test/account/withdraw.ts`

- `0ed...4bb ./test/validators/exit.ts`
- `019...3d0 ./test/validators/remove.ts`
- `d94...cd6 ./test/validators/register.ts`
- `b5e...270 ./test/deployment/version.ts`
- `97d...765 ./test/deployment/deploy.ts`
- `4fe...fee ./test/sanity/balances.ts`
- `411...a99 ./test/liquidate/liquidate.ts`
- `5b5...40e ./test/liquidate/liquidated-cluster.ts`
- `342...d5f ./test/liquidate/reactivate.ts`
- `273...e7f ./test/dao/liquidation-collateral.ts`
- `314...9c7 ./test/dao/liquidation-threshold.ts`
- `ceb...0f7 ./test/dao/network-fee-change.ts`
- `f79...db1 ./test/dao/network-fee-withdraw.ts`
- `3a9...4eb ./test/dao/operational.ts`
- `fb4...01a ./test/operators/others.ts`
- `534...8c8 ./test/operators/update-fee.ts`
- `1fa...11a ./test/operators/remove.ts`
- `7cb...b67 ./test/operators/register.ts`

Automated Analysis

N/A

Test Suite Results

```
  Deposit Tests
    ✔ Deposit to a non liquidated cluster I own emits "ClusterDeposited" (76ms)
    ✔ Deposit to a cluster I own gas limits (47ms)
    ✔ Deposit to a cluster I do not own emits "ClusterDeposited" (44ms)
    ✔ Deposit to a cluster I do not own gas limits (44ms)
    ✔ Deposit to a cluster I do own with a cluster that does not exist reverts "ClusterDoesNotExists"
(48ms)
    ✔ Deposit to a cluster I do not own with a cluster that does not exist reverts "ClusterDoesNotExists"
    ✔ Deposit to a liquidated cluster emits "ClusterDeposited" (145ms)

  Withdraw Tests
    ✔ Withdraw from cluster emits "ClusterWithdrawn" (54ms)
    ✔ Withdraw from cluster gas limits (52ms)
    ✔ Withdraw from operator balance emits "OperatorWithdrawn"
    ✔ Withdraw from operator balance gas limits
    ✔ Withdraw the total operator balance emits "OperatorWithdrawn"
    ✔ Withdraw the total operator balance gas limits
    ✔ Withdraw from a cluster that has a removed operator emits "ClusterWithdrawn" (78ms)
    ✔ Withdraw more than the cluster balance reverts "InsufficientBalance" (66ms)
    ✔ Sequentially withdraw more than the cluster balance reverts "InsufficientBalance" (331ms)
    ✔ Withdraw from a liquidatable cluster reverts "InsufficientBalance" (liquidation threshold) (63ms)
    ✔ Withdraw from a liquidatable cluster reverts "InsufficientBalance" (liquidation collateral) (57ms)
    ✔ Withdraw from a liquidatable cluster after liquidation period reverts "InsufficientBalance" (51ms)
    ✔ Withdraw balance from an operator I do not own reverts "CallerNotOwner"
    ✔ Withdraw more than the operator balance reverts "InsufficientBalance"
    ✔ Sequentially withdraw more than the operator balance reverts "InsufficientBalance" (104ms)
    ✔ Withdraw the total balance from an operator I do not own reverts "CallerNotOwner"
    ✔ Withdraw more than the operator total balance reverts "InsufficientBalance"
    ✔ Withdraw from a cluster without validators (116ms)

  Liquidation Collateral Tests
    ✔ Change minimum collateral emits "MinimumLiquidationCollateralUpdated"
    ✔ Change minimum collateral gas limits
    ✔ Get minimum collateral
    ✔ Change minimum collateral reverts "caller is not the owner"

  Liquidation Threshold Tests
```

```
  ✔ Change liquidation threshold period emits "LiquidationThresholdPeriodUpdated"
  ✔ Change liquidation threshold period gas limits
  ✔ Get liquidation threshold period
  ✔ Change liquidation threshold period reverts "NewBlockPeriodIsBelowMinimum"
  ✔ Change liquidation threshold period reverts "caller is not the owner"

Network Fee Tests
  ✔ Change network fee emits "NetworkFeeUpdated"
  ✔ Change network fee providing UINT64 max value reverts "Max value exceeded"
  ✔ Change network fee when it was set emits "NetworkFeeUpdated"
  ✔ Change network fee gas limit
  ✔ Get network fee
  ✔ Change the network fee to a number below the minimum fee reverts "Max precision exceeded"
  ✔ Change the network fee to a number that exceeds allowed type limit reverts "Max value exceeded"
  ✔ Change network fee from an address thats not the DAO reverts "caller is not the owner"

DAO Network Fee Withdraw Tests
  ✔ Withdraw network earnings emits "NetworkEarningsWithdrawn"
  ✔ Withdraw network earnings gas limits
  ✔ Get withdrawable network earnings
  ✔ Get withdrawable network earnings as not owner
  ✔ Withdraw network earnings with not enough balance reverts "InsufficientBalance"
  ✔ Withdraw network earnings from an address thats not the DAO reverts "caller is not the owner"
  ✔ Withdraw network earnings providing UINT64 max value reverts "Max value exceeded"
  ✔ Withdraw network earnings sequentially when not enough balance reverts "InsufficientBalance" (89ms)

DAO operational Tests
  ✔ Starting the transfer process does not change owner
  ✔ Ownership is transferred in a 2-step process
  ✔ Get the network validators count (add/remove validaotor) (297ms)
  ✔ Get the network validators count (add/remove validaotor) (304ms)

Deployment tests
  ✔ Check default values after deploying (83ms)
  ✔ Upgrade SSVNetwork contract. Check new function execution (112ms)
  ✔ Upgrade SSVNetwork contract. Deploy implemetation manually (69ms)
  ✔ Upgrade SSVNetwork contract. Check base contract is not re-initialized (46ms)
  ✔ Upgrade SSVNetwork contract. Check state is only changed from proxy contract (49ms)
  ✔ Update a module (SSVOperators) (62ms)
  ✔ ETH can not be transferred to SSVNetwork / SSVNetwork views

Version upgrade tests
  ✔ Upgrade contract version number (55ms)

Liquidate Tests
  ✔ Liquidate a cluster via liquidation threshold emits "ClusterLiquidated" (76ms)
  ✔ Liquidate a cluster via minimum liquidation collateral emits "ClusterLiquidated" (72ms)
  ✔ Liquidate a cluster after liquidation period emits "ClusterLiquidated" (66ms)
  ✔ Liquidatable with removed operator (74ms)
  ✔ Liquidatable with removed operator after liquidation period (68ms)
  ✔ Liquidate validator with removed operator in a cluster (117ms)
  ✔ Liquidate and register validator in a disabled cluster reverts "ClusterIsLiquidated" (134ms)
  ✔ Liquidate cluster (4 operators) and check isLiquidated true (98ms)
  ✔ Liquidate cluster (7 operators) and check isLiquidated true (306ms)
  ✔ Liquidate cluster (10 operators) and check isLiquidated true (363ms)
  ✔ Liquidate cluster (13 operators) and check isLiquidated true (441ms)
  ✔ Liquidate a non liquidatable cluster that I own (97ms)
  ✔ Liquidate cluster that I own (107ms)
  ✔ Liquidate cluster that I own after liquidation period (90ms)
  ✔ Get if the cluster is liquidatable (50ms)
  ✔ Get if the cluster is liquidatable after liquidation period (45ms)
  ✔ Get if the cluster is not liquidatable (47ms)
  ✔ Liquidate a cluster that is not liquidatable reverts "ClusterNotLiquidatable" (115ms)
  ✔ Liquidate a cluster that is not liquidatable reverts "IncorrectClusterState"
  ✔ Liquidate already liquidated cluster reverts "ClusterIsLiquidated" (95ms)
  ✔ Is liquidated reverts "ClusterDoesNotExists" (98ms)

Liquidate Cluster Tests
  ✔ Liquidate -> deposit -> reactivate (215ms)
  ✔ RegisterValidator -> liquidate -> removeValidator -> deposit -> withdraw (288ms)
  ✔ Withdraw -> liquidate -> deposit -> reactivate (407ms)
  ✔ Remove validator -> withdraw -> try liquidate reverts "ClusterNotLiquidatable" (341ms)
```

```
  Reactivate Tests
    ✔ Reactivate a disabled cluster emits "ClusterReactivated" (157ms)
    ✔ Reactivate a cluster with a removed operator in the cluster (176ms)
    ✔ Reactivate an enabled cluster reverts "ClusterAlreadyEnabled"
    ✔ Reactivate a cluster when the amount is not enough reverts "InsufficientBalance" (144ms)
    ✔ Reactivate a liquidated cluster after making a deposit (181ms)
    ✔ Reactivate a cluster after liquidation period when the amount is not enough reverts
"InsufficientBalance" (143ms)

  Others Operator Tests
    ✔ Add fee recipient address emits "FeeRecipientAddressUpdated"
    ✔ Remove operator whitelisted address (48ms)
    ✔ Non-owner remove operator whitelisted address reverts "CallerNotOwner" (51ms)
    ✔ Update operator whitelisted address
    ✔ Non-owner update operator whitelisted address reverts "CallerNotOwner"
    ✔ Get the maximum number of validators per operator

  Register Operator Tests
    ✔ Register operator emits "OperatorAdded"
    ✔ Register operator gas limits
    ✔ Get operator by id (40ms)
    ✔ Get private operator by id (51ms)
    ✔ Set operator whitelist gas limits
    ✔ Get non-existent operator by id (39ms)
    ✔ Get operator removed by id (56ms)
    ✔ Register an operator with a fee thats too low reverts "FeeTooLow"
    ✔ Register an operator with a fee thats too high reverts "FeeTooHigh"
    ✔ Register same operator twice reverts "OperatorAlreadyExists"

  Remove Operator Tests
    ✔ Remove operator emits "OperatorRemoved"
    ✔ Remove private operator emits "OperatorRemoved" (73ms)
    ✔ Remove operator gas limits
    ✔ Remove operator with a balance emits "OperatorWithdrawn" (154ms)
    ✔ Remove operator with a balance gas limits (170ms)
    ✔ Remove operator I do not own reverts "CallerNotOwner"
    ✔ Remove same operator twice reverts "OperatorDoesNotExist" (38ms)

  Operator Fee Tests
    ✔ Declare fee emits "OperatorFeeDeclared"
    ✔ Declare fee gas limits"
    ✔ Declare fee with zero value emits "OperatorFeeDeclared"
    ✔ Declare a lower fee gas limits
    ✔ Declare a higher fee gas limit
    ✔ Cancel declared fee emits "OperatorFeeDeclarationCancelled"
    ✔ Cancel declared fee gas limits
    ✔ Execute declared fee emits "OperatorFeeExecuted" (47ms)
    ✔ Execute declared fee gas limits (46ms)
    ✔ Get operator fee
    ✔ Get fee from operator that does not exist returns 0
    ✔ Get operator maximum fee limit
    ✔ Declare fee of operator I do not own reverts "CallerNotOwner"
    ✔ Declare fee with a wrong Publickey reverts "OperatorDoesNotExist"
    ✔ Declare fee when previously set to zero reverts "FeeIncreaseNotAllowed" (65ms)
    ✔ Declare same fee value as actual reverts "SameFeeChangeNotAllowed" (61ms)
    ✔ Declare fee after registering an operator with zero fee reverts "FeeIncreaseNotAllowed"
    ✔ Declare fee above the operators max fee increase limit reverts "FeeExceedsIncreaseLimit"
    ✔ Declare fee above the operators max fee limit reverts "FeeTooHigh"
    ✔ Declare fee too high reverts "FeeTooHigh" -> DAO updates limit -> declare fee emits
"OperatorFeeDeclared" (74ms)
    ✔ Cancel declared fee without a pending request reverts "NoFeeDeclared"
    ✔ Cancel declared fee of an operator I do not own reverts "CallerNotOwner"
    ✔ Execute declared fee of an operator I do not own reverts "CallerNotOwner" (38ms)
    ✔ Execute declared fee without a pending request reverts "NoFeeDeclared"
    ✔ Execute declared fee too early reverts "ApprovalNotWithinTimeframe"
    ✔ Execute declared fee too late reverts "ApprovalNotWithinTimeframe"
    ✔ Reduce fee emits "OperatorFeeExecuted" (67ms)
    ✔ Reduce fee emits "OperatorFeeExecuted"
    ✔ Reduce fee with an increased value reverts "FeeIncreaseNotAllowed"
    ✔ Reduce fee after declaring a fee change (65ms)
    ✔ Reduce maximum fee limit after declaring a fee change reverts "FeeTooHigh (49ms)
```

```
    ✔ DAO increase the fee emits "OperatorFeeIncreaseLimitUpdated"
    ✔ DAO update the maximum operator fee emits "OperatorMaximumFeeUpdated"
    ✔ DAO increase the fee gas limits"
    ✔ DAO update the declare fee period emits "DeclareOperatorFeePeriodUpdated"
    ✔ DAO update the declare fee period gas limits"
    ✔ DAO update the execute fee period emits "ExecuteOperatorFeePeriodUpdated"
    ✔ DAO update the execute fee period gas limits
    ✔ DAO update the maximum fee for operators using SSV gas limits
    ✔ DAO get fee increase limit
    ✔ DAO get declared fee
    ✔ DAO get declared and execute fee periods
    ✔ Increase fee from an address thats not the DAO reverts "caller is not the owner"
    ✔ Update the declare fee period from an address thats not the DAO reverts "caller is not the owner"
    ✔ Update the execute fee period from an address thats not the DAO reverts "caller is not the owner"
    ✔ DAO declared fee without a pending request reverts "NoFeeDeclared"

  Balance Tests
    ✔ Check cluster balance with removing operator (1265ms)
    ✔ Check cluster balance after removing operator, progress blocks and confirm (452ms)
    ✔ Check cluster balance in three blocks, one after the other (139ms)
    ✔ Check cluster balance in two and twelve blocks, after network fee updates (192ms)
    ✔ Check DAO earnings in three blocks, one after the other (39ms)
    ✔ Check DAO earnings in two and twelve blocks, after network fee updates (71ms)
    ✔ Check operators earnings in three blocks, one after the other (179ms)
    ✔ Check cluster balance with removed operator (69ms)
    ✔ Check cluster balance with not enough balance (44ms)
    ✔ Check cluster balance in a non liquidated cluster (44ms)
    ✔ Check cluster balance in a liquidated cluster reverts "ClusterIsLiquidated" (115ms)
    ✔ Check operator earnings, cluster balances and network earnings (571ms)
    ✔ Check operator earnings and cluster balance when reducing operator fee" (88ms)
    ✔ Check cluster balance after withdraw and deposit (393ms)
    ✔ Check cluster and operators balance after 10 validators bulk registration and removal (730ms)
    ✔ Remove validators from a liquidated cluster (696ms)

  Exit Validator Tests
    ✔ Exiting a validator emits "ValidatorExited"
    ✔ Exiting a validator gas limit
    ✔ Exiting one of the validators in a cluster emits "ValidatorExited" (146ms)
    ✔ Exiting a removed validator reverts "IncorrectValidatorStateWithData" (95ms)
    ✔ Exiting a non-existing validator reverts "IncorrectValidatorStateWithData"
    ✔ Exiting a validator with empty operator list reverts "IncorrectValidatorStateWithData"
    ✔ Exiting a validator with empty public key reverts "IncorrectValidatorStateWithData"
    ✔ Exiting a validator using the wrong account reverts "IncorrectValidatorStateWithData"
    ✔ Exiting a validator with incorrect operators (unsorted list) reverts with
"IncorrectValidatorStateWithData"
    ✔ Exiting a validator with incorrect operators (too many operators) reverts with
"IncorrectValidatorState" (292ms)
    ✔ Exiting a validator with incorrect operators reverts with "IncorrectValidatorStateWithData"
    ✔ Bulk exiting a validator emits "ValidatorExited" (493ms)
    ✔ Bulk exiting 10 validator (4 operators cluster) gas limit (494ms)
    ✔ Bulk exiting 10 validator (7 operators cluster) gas limit (629ms)
    ✔ Bulk exiting 10 validator (10 operators cluster) gas limit (768ms)
    ✔ Bulk exiting 10 validator (13 operators cluster) gas limit (923ms)
    ✔ Bulk exiting removed validators reverts "IncorrectValidatorStateWithData" (556ms)
    ✔ Bulk exiting non-existing validators reverts "IncorrectValidatorStateWithData" (457ms)
    ✔ Bulk exiting validators with empty operator list reverts "IncorrectValidatorStateWithData" (439ms)
    ✔ Bulk exiting validators with empty public key reverts "ValidatorDoesNotExist" (426ms)
    ✔ Bulk exiting validators using the wrong account reverts "IncorrectValidatorStateWithData" (437ms)
    ✔ Bulk exiting validators with incorrect operators (unsorted list) reverts with
"IncorrectValidatorStateWithData" (437ms)

  Register Validator Tests
    ✔ Register validator with 4 operators emits "ValidatorAdded" (126ms)
    ✔ Register validator with 4 operators gas limit (134ms)
    ✔ Register 2 validators into the same cluster gas limit (257ms)
    ✔ Register 2 validators into the same cluster and 1 validator into a new cluster gas limit (375ms)
    ✔ Register 2 validators into the same cluster with one time deposit gas limit (263ms)
    ✔ Bulk register 10 validators with 4 operators into the same cluster (557ms)
    ✔ Bulk register 10 validators with 4 operators new cluster (416ms)
    ✔ Register validator with 7 operators gas limit (167ms)
    ✔ Register 2 validators with 7 operators into the same cluster gas limit (338ms)
    ✔ Register 2 validators with 7 operators into the same cluster and 1 validator into a new cluster
```

with 7 operators gas limit (521ms)
    ✔ Register 2 validators with 7 operators into the same cluster with one time deposit gas limit (326ms)
    ✔ Bulk register 10 validators with 7 operators into the same cluster (722ms)
    ✔ Bulk register 10 validators with 7 operators new cluster (530ms)
    ✔ Register validator with 10 operators gas limit (220ms)
    ✔ Register 2 validators with 10 operators into the same cluster gas limit (426ms)
    ✔ Register 2 validators with 10 operators into the same cluster and 1 validator into a new cluster with 10 operators gas limit (640ms)
    ✔ Register 2 validators with 10 operators into the same cluster with one time deposit gas limit (407ms)
    ✔ Bulk register 10 validators with 10 operators into the same cluster (890ms)
    ✔ Bulk register 10 validators with 10 operators new cluster (678ms)
    ✔ Register validator with 13 operators gas limit (257ms)
    ✔ Register 2 validators with 13 operators into the same cluster gas limit (510ms)
    ✔ Register 2 validators with 13 operators into the same cluster and 1 validator into a new cluster with 13 operators gas limit (780ms)
    ✔ Register 2 validators with 13 operators into the same cluster with one time deposit gas limit (501ms)
    ✔ Bulk register 10 validators with 13 operators into the same cluster (1105ms)
    ✔ Bulk register 10 validators with 13 operators new cluster (798ms)
    ✔ Get cluster burn rate (227ms)
    ✔ Get cluster burn rate when one of the operators does not exist
    ✔ Register validator with incorrect input data reverts "IncorrectClusterState" (175ms)
    ✔ Register validator in a new cluster with incorrect input data reverts "IncorrectClusterState" (57ms)
    ✔ Register validator when an operator does not exist in the cluster reverts "OperatorDoesNotExist" (86ms)
    ✔ Register validator with a removed operator in the cluster reverts "OperatorDoesNotExist" (76ms)
    ✔ Register cluster with unsorted operators reverts "UnsortedOperatorsList" (47ms)
    ✔ Register cluster with duplicated operators reverts "OperatorsListNotUnique" (72ms)
    ✔ Register validator with not enough balance reverts "InsufficientBalance" (113ms)
    ✔ Register validator in a liquidatable cluster with not enough balance reverts "InsufficientBalance" (244ms)
    ✔ Register an existing validator with same operators setup reverts "ValidatorAlreadyExistsWithData" (51ms)
    ✔ Register an existing validator with different operators setup reverts "ValidatorAlreadyExistsWithData" (51ms)
    ✔ Register whitelisted validator in 1 operator with 4 operators emits "ValidatorAdded" (156ms)
    ✔ Register a non whitelisted validator reverts "CallerNotWhitelisted" (124ms)
    ✔ Retrieve an existing validator
    ✔ Retrieve a non-existing validator

  Remove Validator Tests
    ✔ Remove validator emits "ValidatorRemoved" (68ms)
    ✔ Bulk remove validator emits "ValidatorRemoved" (549ms)
    ✔ Remove validator after cluster liquidation period emits "ValidatorRemoved" (69ms)
    ✔ Remove validator gas limit (4 operators cluster) (75ms)
    ✔ Bulk remove 10 validator gas limit (4 operators cluster) (556ms)
    ✔ Remove validator gas limit (7 operators cluster) (267ms)
    ✔ Bulk remove 10 validator gas limit (7 operators cluster) (727ms)
    ✔ Remove validator gas limit (10 operators cluster) (345ms)
    ✔ Bulk remove 10 validator gas limit (10 operators cluster) (865ms)
    ✔ Remove validator gas limit (13 operators cluster) (404ms)
    ✔ Bulk remove 10 validator gas limit (13 operators cluster) (1030ms)
    ✔ Remove validator with a removed operator in the cluster (96ms)
    ✔ Register a removed validator and remove the same validator again (255ms)
    ✔ Remove validator from a liquidated cluster (103ms)
    ✔ Remove validator with an invalid owner reverts "ClusterDoesNotExists"
    ✔ Remove validator with an invalid operator setup reverts "ClusterDoesNotExists"
    ✔ Remove the same validator twice reverts "ValidatorDoesNotExist" (99ms)
    ✔ Remove the same validator with wrong input parameters reverts "IncorrectClusterState" (94ms)
    ✔ Bulk Remove validator that does not exist in a valid cluster reverts "IncorrectValidatorStateWithData" (453ms)
    ✔ Bulk remove validator with an invalid operator setup reverts "ClusterDoesNotExists" (436ms)
    ✔ Bulk Remove the same validator twice reverts "IncorrectValidatorStateWithData" (578ms)
    ✔ Remove validators from a liquidated cluster (552ms)
    ✔ Bulk remove 10 validator with duplicated public keys reverts "IncorrectValidatorStateWithData" (473ms)
    ✔ Bulk remove 10 validator with empty public keys reverts "IncorrectValidatorStateWithData"

```
  265 passing (4m)
```

Code Coverage

The code coverage was generated for using `npm run solidity-coverage` and `.solcover.js` provided below.

```
module.exports = {
    skipFiles: ['deprecated', 'test', 'upgrades', 'mocks', 'token'],
};
```

We recommend adding more tests to ensure the branch coverage is larger than 90% before deploying the contracts.

Update: The SSV team added additional tests to cover the fixes introduced after the initial audit. We still recommend improving the test suite so that branch coverage is above 90% for all contracts.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	71.88	98.28	100	
SSVNetwork.sol	100	80.77	100	100	
SSVNetworkViews.sol	100	33.33	95.45	100	
SSVProxy.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
ISSVClusters.sol	100	100	100	100	
ISSVDAO.sol	100	100	100	100	
ISSVNetwork.sol	100	100	100	100	
ISSVNetworkCore.sol	100	100	100	100	
ISSVOperators.sol	100	100	100	100	
ISSVViews.sol	100	100	100	100	
contracts/libraries/	100	77.5	100	94.12	
ClusterLib.sol	100	75	100	100	
CoreLib.sol	100	75	100	84.62	15,21
OperatorLib.sol	100	92.31	100	95.12	54,82
ProtocolLib.sol	100	75	100	91.67	43
SSVStorage.sol	100	100	100	100	
SSVStorageProtocol.sol	100	100	100	100	
Types.sol	100	100	100	100	
ValidatorLib.sol	100	40	100	83.33	21,27

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/modules/	99.58	93.48	97.92	99.38	
SSVClusters.sol	100	90.48	100	99.3	100
SSVDAO.sol	100	100	100	100	
SSVOperators.sol	100	97.22	100	100	
SSVViews.sol	98.18	90	94.74	98.31	203
All files	99.73	83.82	98.54	98.24	

Changelog

- 2024-02-02 - Initial report
- 2024-02-14 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.