Quantstamp

SSV Network - Diff 3

Executive Summary

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

Type	Registry
Timeline	2024-06-12 through 2024-06-19
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	• ssvlabs/ssv-network #c928cf2
Auditors	• Jennifer Wu Auditing Engineer • Roman Rohleder Senior Auditing Engineer • Cameron Biniamow Auditing Engineer

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

Summary of Findings

This audit report is a third diff audit of the SSV.network contracts, explicitly highlighting the changes made between tags v1.1.0 and v1.2.0. Readers should review this diff audit alongside the initial SSV.network audit report and the other two diff SSV.network audit reports, as the scope of this diff audit is strictly limited to changes between v1.1.0 and v1.2.0.

The main modifications between v1.1.0 and v1.2.0 focus on updating operator whitelisting functionality to allow multiple whitelisted addresses. Consequently, a new `SSVOperatorsWhitelist` module contract was created, and the `OperatorLib` library was heavily refactored. Other minor changes include upgrading the Solidity version from 0.8.18 to 0.8.24, updating imports, and performing minor refactoring.

The diff audit resulted in five findings and best practices. We recommend the client to consider all identified issues. We also recommend enhancing the test suite by adding additional assertions to validate the expected outcomes after specific contract function calls, ensuring all conditions and edge cases are thoroughly tested.

Fix Review: The SSV team has fixed or acknowledged all issues listed in this report.

Fix Review 2: The SSV team updated the `updateClusterOperatorsOnRegistration()` function in bac7ff4 by replacing the `InvalidWhitelistingContract` error with the `CallerNotWhitelistedWithData` error to ensure clearer and more consistent messaging when the caller is not properly whitelisted, due to support for legacy EOAs and generic contracts.

Fix Review 3: The SSV team updated the interface files to use unlocked pragma for improved compatibility in 59abd6b.

ID	DESCRIPTION	SEVERITY	STATUS
SSV-1	Denial-of-Service When Number Of Registered Operators is Sufficiently High	Low	Fixed
SSV-2	Non-Removable Whitelisting Contract	Low	Fixed
SSV-3	Front Run to Add Operator Before Whitelist Update	Informational	Acknowledged
SSV-4	Missing Input Validation	Informational	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
SSV-5	Whitelist Does Not Apply During Cluster Reactivation	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 v1.1.0 and v1.2.0 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.

Findings

SSV-1

Denial-of-Service When Number Of Registered Operators is Sufficiently High

Low Fixed

Update

The client fixed the issue in commit `2f04029afd76f4208a9fb27736bd93ee0e26ff33` by refactoring the `generateBlockMasks()` and `updateMultipleWhitelists()` functions to reduce the size of the `masks` array based on the first operator ID. However, if the operator IDs are sufficiently spaced apart, the `masks` array can still become large. This issue can be mitigated by ensuring that the `operatorIds` input is within a similar range.

The client provided the following explanation:

```
Functions updateMultipleWhitelists and generateBlockMasks were changed to make the first blockIndex
according to the first operator ID in the list.
```

File(s) affected: `OperatorLib.sol`, `SSVOperatorsWhitelist.sol`, `SSVViews.sol`

Description: If the number of registered operators is sufficiently high, calls to `OperatorLib.updateMultipleWhitelists()` and `SSVViews.getWhitelistedOperators()` could run out of gas as they iterate over a large array of masks created in `OperatorLib.generateBlockMasks()`. There are concerns about gas costs caused by `OperatorLib.generateBlockMasks()` when a wide range of operator IDs are used. The mask array can potentially have many unused entries if the operator IDs are sparse and have large gaps between them. Additionally, `blockIndex` starts at 0, so larger operator IDs will incur higher gas costs to update the whitelist due to the number of iterations of `blockIndex`.

Recommendation: While it is unlikely that the number of registered operators will become large enough to cause a denial of service, it is recommended to refactor these functions to avoid iterating over a potentially large array.

SSV-2 Non-Removable Whitelisting Contract

Low Fixed

Update

The client fixed the issue in `2f04029afd76f4208a9fb27736bd93ee0e26ff33` by checking the `whitelistAddress` only when registering.

The client provided the following explanation:

```
The check for SSV compatible whitelisting contracts were removed when removing whitelisted addresses.
```

File(s) affected: `OperatorLib.sol`

Description: The `setOperatorsWhitelistingContract()` function updates the whitelisting contract for a list of operators. It validates whether the provided contract conforms to the expected whitelisting contract interface. If the current whitelisted contract is not a valid whitelisting contract according to the ERC165 standard, it is moved to a bitmask-based legacy support. This ensures that operators can continue to function even if their associated whitelisting contracts become invalid or change interfaces. However, if a whitelisting contract is upgraded and no longer passes the ERC165 checker, it will still be moved to legacy support during the next whitelist update. Similarly, when removing an operator whitelist through `removeOperatorsWhitelists()`, if the ERC165 check does not pass, it will not be possible to remove the operator whitelist from the bitmask legacy approval `addressWhitelistedForOperators`. Therefore, an approved whitelisted address in the bitmask legacy can always deny the operator from removing access.

Exploit Scenario:

1. An operator whitelists `WhitelistingContract` using `setOperatorsWhitelistingContract()`.
2. The `WhitelistingContract` upgrades and now fails the ERC165 checker.
3. The operator changes the whitelisting contract to `WhitelistingContract2` using `setOperatorsWhitelistingContract()`.
4. The previous `WhitelistingContract` is moved to the `addressWhitelistedForOperators` bitmask for legacy support, and `WhitelistingContract2` is set as the new whitelisting contract.
5. The previous `WhitelistingContract` upgrades again and now passes the ERC165 checker.
6. The operator tries to remove `WhitelistingContract` from the `addressWhitelistedForOperators` bitmask using `removeOperatorsWhitelists()`. The function reverts because `WhitelistingContract` is now a valid whitelisting contract based on the `isWhitelistingContract` check.

Recommendation: Remove the ERC165 check when removing an operator whitelist in the `updateMultipleWhitelists()` function. This adjustment will ensure that the operator whitelist can be removed from the bitmask legacy approval `addressWhitelistedForOperators`, even if the contract later conforms to the ERC165 interface.

SSV-3
Front Run to Add Operator Before Whitelist Update

Informational | Acknowledged

Update
The client acknowledged the issue and updated the documentation in `2f04029afd76f4208a9fb27736bd93ee0e26ff33`.

File(s) affected: `SSVClusters.sol`, `ClusterLib.sol`, `OperatorLib.sol`

Description: The whitelist implementation for validator operators only checks the whitelist during the registration of new validators. Existing validators can continue to operate under an operator even if the operator's whitelist changes, meaning that once registered, validators are not subject to new whitelist restrictions. If an operator changes its status to private and adds or updates a whitelist, new registrations will need to adhere to this whitelist. However, validators that were registered before this change will not be affected. This creates an opportunity to front-run the addition or update of a whitelist by registering validators before the whitelist is modified.

Recommendation: Based on discussions with the client, this behavior is an intentional design choice. It is important to document that the whitelist does not apply to previously registered validators. This documentation should clearly state that any updates to the whitelist will not affect existing validators, and only new registrations will be subject to the updated whitelist criteria.

SSV-4 Missing Input Validation

Informational | Fixed

Update
The client fixed the issue and updated the function `setOperatorsWhitelistingContract` to check if `currentWhitelisted` is `address(0)`.

File(s) affected: `SSVOperatorsWhitelist.sol`

Description: It is crucial to validate inputs, even if the inputs come from trusted addresses, to avoid human error. A lack of robust input validation can only increase the likelihood and impact in the event of mistakes.

Following is the list of places that can potentially benefit from stricter input validation:
1. `SSVOperatorsWhitelist.setOperatorsWhitelistingContract()`: `currentWhitelisted` is not checked for zero address.

Recommendation: Add the validations and checks listed in the description.

SSV-5
Whitelist Does Not Apply During Cluster Reactivation

Informational | Acknowledged

Update
The client acknowledged the issue and provided the following explanation:

```
This is a decision made by design.
```

File(s) affected: `SSVClusters.sol`

Description: Whitelist validation for validator operators only occurs during the initial registration of new validators and the selection of operators. If an operator removes a validator from their whitelist after the validator has already been added to a cluster, the removed validator can still reactivate despite no longer being on the whitelist. Specifically, if a validator gets liquidated and becomes deactivated, it can still reactivate even if it has been removed from the whitelist. This means that once a validator has been added to a cluster, subsequent removals from the whitelist do not affect the validator's ability to reactivate.

Recommendation: Based on discussions with the client, this behavior is intentional by design. The whitelist check is not performed during cluster reactivation. This design ensures that validators already associated with an operator will continue to function and can be reactivated even if they are no longer on the whitelist. Therefore, it is important to document that whitelist checks are not enforced during cluster reactivation. This documentation ensures clarity about the operational behavior of validators and clusters under the current whitelist design.

Auditor Suggestions

SSV-S-1

Split `getWhitelistedOperators()` Function to Improve Readability and Maintainability

Acknowledged

> **Update**
> The client acknowledged the issue and provided the following explanation:
>
> ```
> The SSV UI will use this function. We will consider simplifying it and moving part of its logic off-
> chain in the future.
> ```

File(s) affected: `SSVViews.sol`

Description: The `getWhitelistedOperators()` function is complex and supports both bitmask approval and whitelisting contracts simultaneously. To improve the readability and maintainability of this function, we recommend splitting it into two separate functions. One function will handle whitelisting based on whitelisting contracts, and the other will handle legacy approval. Off-chain components can merge the whitelisted operator IDs off-chain. This approach reduces the complexity of the function and delegates the merging process to off-chain components. Additionally, to avoid resizing the array within the function, we suggest returning a list of booleans indicating whether each operator ID is approved.

Based on communication with the client, the function is primarily used by the SSV UI before registering validators to check if the caller is whitelisted in the chosen operators. This function can be called from external contracts or used off-chain. By simplifying the function and delegating the merging process to off-chain components, we can improve the function's readability and maintainability and reduce complexity on-chain.

Recommendation:

1. Split the `getWhitelistedOperators()` function into two separate functions:
    1. One function to return whitelisted operator IDs based on whitelisting contracts.
    2. Another function to return whitelisted operator IDs based on legacy bitmask approval.
2. Delegate the merging process to off-chain components. Off-chain components should merge the results from the two functions.
3. Modify the functions to return a list of booleans indicating the approval status of each operator ID, instead of resizing arrays within the function.

Code Documentation

1. The following typographical errors have been noted:
    1. Fixed operators.md#L24: ot → to.
    2. Fixed operators.md#L38 and L42: whitelising → whitelisting.
2. Fixed operators.md: Mentions that in function `registerOperator()` "whitelisted" status is set to "false". However, with the new changes whitelisted is set to the new `isPrivate` flag.

Adherence to Best Practices

1. Acknowledged Consider moving the access control check (call to `checkOwner()`) from the `generateBlockMasks()` function to the `updateMultipleWhitelists()` function to mitigate potential access control violations should the code be refactored and the flag `checkOperatorsOwnership` remain unset.
2. Acknowledged The `generateBlockMasks()` function should be renamed to clearly reflect its responsibilities in performing ownership and operator ID validation.
3. Fixed Rename the `setOperatosWhitelists()` function to correct the misspelling of "operators."
4. Fixed The `while` loop in the function `getWhitelistedOperators()` can be converted to a more concise and idiomatic `for` loop by initializing the loop variable `operatorIndex` within the for loop declaration, checking the condition `operatorIndex < operatorsLength` before each iteration, and incrementing `operatorIndex` at the end of each iteration.

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).

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.

Files

- `6d4...2de ./contracts/SSVNetwork.sol`
- `1f1...316 ./contracts/SSVNetworkViews.sol`
- `17b...220 ./contracts/SSVProxy.sol`
- `3cc...06b ./contracts/interfaces/ISSVOperators.sol`
- `c10...b90 ./contracts/interfaces/ISSVNetwork.sol`
- `562...51f ./contracts/interfaces/ISSVOperatorsWhitelist.sol`
- `bc3...937 ./contracts/interfaces/ISSVViews.sol`
- `0a2...43e ./contracts/interfaces/ISSVDAO.sol`
- `a3f...604 ./contracts/interfaces/ISSVClusters.sol`
- `1d8...6c4 ./contracts/interfaces/ISSVNetworkCore.sol`
- `59e...4a1 ./contracts/interfaces/external/ISSVWhitelistingContract.sol`
- `1c8...4a4 ./contracts/modules/SSVOperators.sol`
- `dca...8ad ./contracts/modules/SSVViews.sol`
- `013...bf8 ./contracts/modules/SSVOperatorsWhitelist.sol`
- `c44...53d ./contracts/modules/SSVClusters.sol`
- `79d...a2f ./contracts/modules/SSVDAO.sol`
- `266...1a5 ./contracts/libraries/SSVStorage.sol`
- `816...ff6 ./contracts/libraries/SSVStorageProtocol.sol`
- `dd1...235 ./contracts/libraries/ValidatorLib.sol`
- `768...f90 ./contracts/libraries/OperatorLib.sol`
- `f0d...051 ./contracts/libraries/CoreLib.sol`
- `703...3c5 ./contracts/libraries/ClusterLib.sol`
- `904...8bc ./contracts/libraries/ProtocolLib.sol`
- `084...90c ./contracts/libraries/Types.sol`

Tests

- `9cb...1be ./test-forked/operators-whitelist.ts`
- `e98...902 ./test-forked/v1.1.1/SSVNetworkViews.ts`
- `bed...672 ./test-forked/v1.1.1/SSVNetwork.ts`
- `637...c46 ./test/helpers/gas-usage.ts`
- `7c7...a38 ./test/helpers/types.ts`
- `5ed...824 ./test/helpers/contract-helpers.ts`
- `40a...26c ./test/helpers/utils/test.ts`
- `3b5...7ab ./test/account/deposit.ts`
- `5d5...ee0 ./test/account/withdraw.ts`
- `f0c...d0d ./test/validators/whitelist-register.ts`
- `75d...49b ./test/validators/exit.ts`
- `f1d...f39 ./test/validators/remove.ts`
- `a74...d67 ./test/validators/register.ts`
- `a80...3ff ./test/deployment/deploy.ts`

- `466...446 ./test/sanity/balances.ts`
- `57d...a04 ./test/liquidate/liquidate.ts`
- `ec6...cc7 ./test/liquidate/liquidated-cluster.ts`
- `00a...44c ./test/liquidate/reactivate.ts`
- `3db...c49 ./test/dao/liquidation-collateral.ts`
- `84d...8fa ./test/dao/liquidation-threshold.ts`
- `bf4...4a5 ./test/dao/network-fee-change.ts`
- `13f...2ee ./test/dao/network-fee-withdraw.ts`
- `15b...f83 ./test/dao/operational.ts`
- `9f1...a08 ./test/operators/others.ts`
- `0c6...507 ./test/operators/update-fee.ts`
- `73f...f1b ./test/operators/remove.ts`
- `397...d8c ./test/operators/whitelist.ts`
- `f92...fce ./test/operators/register.ts`

Test Suite Results

The tests were run using `npx hardhat test` and `FORK_TESTING_ENABLED=true npx hardhat test test-forked/*.ts`.

Fix Review: The client added 1 test following the initial audit.

```
Deposit Tests
  ✔ Deposit to a non liquidated cluster I own emits "ClusterDeposited"
  ✔ Deposit to a cluster I own gas limits
  ✔ Deposit to a cluster I do not own emits "ClusterDeposited"
  ✔ Deposit to a cluster I do not own gas limits
  ✔ Deposit to a cluster I do not own with a cluster that does not exist reverts "ClusterDoesNotExists"
  ✔ Deposit to a liquidated cluster emits "ClusterDeposited"
  ✔ Deposit to a cluster I do own with a cluster that does not exist reverts "ClusterDoesNotExists"

Withdraw Tests
  ✔ Withdraw from cluster emits "ClusterWithdrawn"
  ✔ Withdraw from cluster gas limits
  ✔ 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"
  ✔ Withdraw more than the cluster balance reverts "InsufficientBalance"
  ✔ Sequentially withdraw more than the cluster balance reverts "InsufficientBalance"
  ✔ Withdraw from a liquidatable cluster reverts "InsufficientBalance" (liquidation threshold)
  ✔ Withdraw from a liquidatable cluster reverts "InsufficientBalance" (liquidation collateral)
  ✔ Withdraw from a liquidatable cluster after liquidation period reverts "InsufficientBalance"
  ✔ Withdraw balance from an operator I do not own reverts "CallerNotOwnerWithData"
  ✔ Withdraw more than the operator balance reverts "InsufficientBalance"
  ✔ Sequentially withdraw more than the operator balance reverts "InsufficientBalance"
  ✔ Withdraw the total balance from an operator I do not own reverts "CallerNotOwnerWithData"
  ✔ Withdraw more than the operator total balance reverts "InsufficientBalance"
  ✔ Withdraw from a cluster without validators

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"

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)
  ✔ Get the network validators count (add/remove validaotor)

Deployment tests
  ✔ Check default values after deploying
  ✔ Upgrade SSVNetwork contract. Check new function execution
  ✔ Upgrade SSVNetwork contract. Deploy implemetation manually
  ✔ Upgrade SSVNetwork contract. Check base contract is not re-initialized
  ✔ Upgrade SSVNetwork contract. Check state is only changed from proxy contract
  ✔ ETH can not be transferred to SSVNetwork / SSVNetwork views

Liquidate Tests
  ✔ Liquidate a cluster via liquidation threshold emits "ClusterLiquidated"
  ✔ Liquidate a cluster via minimum liquidation collateral emits "ClusterLiquidated"
  ✔ Liquidate a cluster after liquidation period emits "ClusterLiquidated"
  ✔ Liquidatable with removed operator
  ✔ Liquidatable with removed operator after liquidation period
  ✔ Liquidate validator with removed operator in a cluster
  ✔ Liquidate and register validator in a disabled cluster reverts "ClusterIsLiquidated"
  ✔ Liquidate cluster (4 operators) and check isLiquidated true
  ✔ Liquidate cluster (7 operators) and check isLiquidated true
  ✔ Liquidate cluster (10 operators) and check isLiquidated true
  ✔ Liquidate cluster (13 operators) and check isLiquidated true
  ✔ Liquidate a non liquidatable cluster that I own
  ✔ Liquidate cluster that I own
  ✔ Liquidate cluster that I own after liquidation period
  ✔ Get if the cluster is liquidatable
    ✔ Get if the cluster is liquidatable after liquidation period
    ✔ Get if the cluster is not liquidatable
    ✔ Liquidate a cluster that is not liquidatable reverts "ClusterNotLiquidatable"
    ✔ Liquidate a cluster that is not liquidatable reverts "IncorrectClusterState"
    ✔ Liquidate already liquidated cluster reverts "ClusterIsLiquidated"
    ✔ Is liquidated reverts "ClusterDoesNotExists" (1086ms)

  Liquidate Cluster Tests
    ✔ Liquidate -> deposit -> reactivate
    ✔ RegisterValidator -> liquidate -> removeValidator -> deposit -> withdraw (41ms)
    ✔ Withdraw -> liquidate -> deposit -> reactivate (1101ms)
    ✔ Remove validator -> withdraw -> try liquidate reverts "ClusterNotLiquidatable"

  Reactivate Tests
    ✔ Reactivate a disabled cluster emits "ClusterReactivated"
    ✔ Reactivate a cluster with a removed operator in the cluster
    ✔ Reactivate an enabled cluster reverts "ClusterAlreadyEnabled"
    ✔ Reactivate a cluster when the amount is not enough reverts "InsufficientBalance"
    ✔ Reactivate a liquidated cluster after making a deposit
    ✔ Reactivate a cluster after liquidation period when the amount is not enough reverts
"InsufficientBalance"

  Others Operator Tests
    ✔ Add fee recipient address emits "FeeRecipientAddressUpdated"
```

```
    ✔ Get the maximum number of validators per operator

  Register Operator Tests
    ✔ Register operator emits "OperatorAdded" and "OperatorPrivacyStatusUpdated" if setPrivate is true
    ✔ Register operator emits "OperatorAdded" and "OperatorPrivacyStatusUpdated" if setPrivate is false
    ✔ Register operator gas limits
    ✔ Get operator by id with setPrivate false
    ✔ Get operator by id with setPrivate true
    ✔ Get non-existent operator by id
    ✔ Get operator removed by id
    ✔ Register an operator with a fee thats too low reverts "FeeTooLow", setPrivate false
    ✔ Register an operator with a fee thats too low reverts "FeeTooLow", setPrivate true
    ✔ Register an operator with a fee thats too high reverts "FeeTooHigh", setPrivate false
    ✔ Register an operator with a fee thats too high reverts "FeeTooHigh", setPrivate false
    ✔ Register same operator twice reverts "OperatorAlreadyExists", setPrivate false
    ✔ Register same operator twice reverts "OperatorAlreadyExists", setPrivate true

  Remove Operator Tests
    ✔ Remove operator emits "OperatorRemoved"
    ✔ Remove private operator emits "OperatorRemoved"
    ✔ Remove operator gas limits
    ✔ Remove operator with a balance emits "OperatorWithdrawn"
    ✔ Remove operator with a balance gas limits
    ✔ Remove operator I do not own reverts "CallerNotOwnerWithData"
    ✔ Remove same operator twice reverts "OperatorDoesNotExist"

  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"
    ✔ Execute declared fee gas limits
    ✔ 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 "CallerNotOwnerWithData"
    ✔ Declare fee with a wrong Publickey reverts "OperatorDoesNotExist"
    ✔ Declare fee when previously set to zero reverts "FeeIncreaseNotAllowed"
    ✔ Declare same fee value as actual reverts "SameFeeChangeNotAllowed"
    ✔ 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"
    ✔ Cancel declared fee without a pending request reverts "NoFeeDeclared"
    ✔ Cancel declared fee of an operator I do not own reverts "CallerNotOwnerWithData"
    ✔ Execute declared fee of an operator I do not own reverts "CallerNotOwnerWithData"
    ✔ 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"
    ✔ Reduce fee emits "OperatorFeeExecuted"
    ✔ Reduce fee with a fee thats too low reverts "FeeTooLow"
    ✔ Reduce fee with an increased value reverts "FeeIncreaseNotAllowed"
    ✔ Reduce fee after declaring a fee change
    ✔ Reduce maximum fee limit after declaring a fee change reverts "FeeTooHigh
    ✔ 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"

  Whitelisting Operator Tests
    ✔ Set operator whitelisting contract (1 operator) gas limits
    ✔ Update operator whitelisting contract (1 operator) gas limits
    ✔ Set operator whitelisting contract (10 operators) gas limits
    ✔ Remove operator whitelisting contract (1 operator) gas limits
    ✔ Remove operator whitelisting contract (10 operators) gas limits
    ✔ Set 10 whitelist addresses (EOAs) for 10 operators gas limits
    ✔ Remove 10 whitelist addresses (EOAs) for 10 operators gas limits
    ✔ Set operators private (10 operators) gas limits
    ✔ Set operators public (10 operators) gas limits
    ✔ Set operator whitelisting contract (10 operators) emits "OperatorWhitelistingContractUpdated"
    ✔ Remove operator whitelisting contract (10 operators) emits "OperatorWhitelistingContractUpdated"
    ✔ Set 10 whitelist addresses (EOAs) for 10 operators emits "OperatorMultipleWhitelistUpdated"
    ✔ Remove 10 whitelist addresses (EOAs) for 10 operators emits "OperatorMultipleWhitelistRemoved"
    ✔ Set operators private (10 operators) emits "OperatorPrivacyStatusUpdated"
    ✔ Set operators public (10 operators) emits "OperatorPrivacyStatusUpdated"
    ✔ Set operator whitelisted address (EOA) in non-existing operator reverts "OperatorDoesNotExist"
    ✔ Set multiple operator whitelisted addresses (zero address) reverts "ZeroAddressNotAllowed"
    ✔ Non-owner sets multiple operator whitelisted addresses (EOA) reverts "CallerNotOwnerWithData"
    ✔ Set multiple operator whitelisted addresses (EOA) with empty operator IDs reverts
"InvalidOperatorIdsLength"
    ✔ Set multiple operator whitelisted addresses (EOA) with empty addresses IDs reverts
"InvalidWhitelistAddressesLength"
    ✔ Set multiple operator whitelisted addresses (EOA) passing unsorted operator IDs reverts
"UnsortedOperatorsList"
    ✔ Set multiple operator whitelisted addresses (EOA) passing a whitelisting contract reverts
"AddressIsWhitelistingContract"
    ✔ Non-owner removes multiple operator whitelisted addresses (EOA) reverts "CallerNotOwnerWithData"
    ✔ Remove multiple operator whitelisted addresses (EOA) passing unsorted operator IDs reverts
"UnsortedOperatorsList"
    ✔ Remove multiple operator whitelisted addresses (EOA) with empty operator IDs reverts
"InvalidOperatorIdsLength"
    ✔ Remove multiple operator whitelisted addresses (EOA) with empty addresses IDs reverts
"InvalidWhitelistAddressesLength"
    ✔ Remove multiple operator whitelisted addresses (EOA) passing a whitelisting contract reverts
"AddressIsWhitelistingContract"
    ✔ Set operator whitelisting contract with an EOA reverts "InvalidWhitelistingContract"
    ✔ Set operator whitelisting contract with empty operator IDs reverts "InvalidOperatorIdsLength"
    ✔ Non-owner sets operator whitelisting contract reverts "CallerNotOwnerWithData"
    ✔ Sets operator whitelisting contract for a non-existing operator reverts "OperatorDoesNotExist"
    ✔ Remove operator whitelisting contract with empty operator IDs reverts "InvalidOperatorIdsLength"
    ✔ Non-owner removes operator whitelisting contract reverts "CallerNotOwnerWithData"
    ✔ Set operators private with empty operator IDs reverts "InvalidOperatorIdsLength"
    ✔ Set operators public with empty operator IDs reverts "InvalidOperatorIdsLength"
    ✔ Non-owner set operators private reverts "CallerNotOwnerWithData"
    ✔ Non-owner set operators public reverts "CallerNotOwnerWithData"
    ✔ Whitelist accounts passing repeated operator IDs reverts "OperatorsListNotUnique"
    ✔ Remove whitelist addresses passing repeated operator IDs reverts "OperatorsListNotUnique"
    ✔ Get whitelisted address for no operators returns empty list
    ✔ Get whitelisted zero address for operators returns empty list
    ✔ Get whitelisted address for operators returns the whitelisted operators (only SSV whitelisting
module) (1326ms)
    ✔ Get whitelisted address for operators returns the whitelisted operators (only externally
whitelisted) (1302ms)
    ✔ Get whitelisted address for operators returns the whitelisted operators (internally and externally
whitelisted) (1383ms)
    ✔ Get whitelisted address for a single operator whitelisted both internally and externally
    ✔ Get whitelisted address for overlapping internal and external whitelisting (38ms)
    ✔ Get whitelisted address for a list containing non-whitelisted operators
    ✔ Get whitelisted address for non-existent operator IDs
    ✔ Get whitelisted address for mixed whitelisted and non-whitelisted addresses
    ✔ Get whitelisted address for unsorted operators (1094ms)
    ✔ Get whitelisted address for duplicate operator IDs (1097ms)
    ✔ Get whitelisted address for a large number of operator IDs (4832ms)
    ✔ Get private operator by id
    ✔ Get removed private operator by id
    ✔ Check if an address is a whitelisting contract
    ✔ Set operators private (10 operators)
```

```
  ✔ Set operators private (10 operators) (74ms)
  ✔ Check account is whitelisted in a whitelisting contract
  ✔ Check account is not whitelisted in a whitelisting contract
  ✔ Check address(0) account in a whitelisting contract
  ✔ Check account in an address(0) contract
  ✔ Set multiple whitelisted addresses to one operator
  ✔ Set 10 whitelist addresses (EOAs) for 10 operators
  ✔ Set 1 whitelist addresses for 1 operator
  ✔ Custom test: Operators balances sync (42ms)

Balance Tests
  ✔ Check cluster balance with removing operator
  ✔ Check cluster balance after removing operator, progress blocks and confirm (1098ms)
  ✔ Check cluster balance in three blocks, one after the other
  ✔ Check cluster balance in two and twelve blocks, after network fee updates
  ✔ Check DAO earnings in three blocks, one after the other
  ✔ Check DAO earnings in two and twelve blocks, after network fee updates
  ✔ Check operators earnings in three blocks, one after the other
  ✔ Check cluster balance with removed operator
  ✔ Check cluster balance with not enough balance
  ✔ Check cluster balance in a non liquidated cluster
  ✔ Check cluster balance in a liquidated cluster reverts "ClusterIsLiquidated" (1069ms)
  ✔ Check operator earnings, cluster balances and network earnings
  ✔ Check cluster balance after withdraw and deposit
    ✔ Check cluster and operators balance after 10 validators bulk registration and removal (175ms)
    ✔ Remove validators from a liquidated cluster (185ms)

  Balance Tests (reduce fee)
    ✔ Check operator earnings and cluster balance when reducing operator fee"

  Exit Validator Tests
    ✔ Exiting a validator emits "ValidatorExited"
    ✔ Exiting a validator gas limit
    ✔ Exiting one of the validators in a cluster emits "ValidatorExited"
    ✔ Exiting a removed validator reverts "IncorrectValidatorStateWithData"
    ✔ 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"
    ✔ Exiting a validator with incorrect operators reverts with "IncorrectValidatorStateWithData"
    ✔ Bulk exiting a validator emits "ValidatorExited" (181ms)
    ✔ Bulk exiting 10 validator (4 operators cluster) gas limit (171ms)
    ✔ Bulk exiting 10 validator (7 operators cluster) gas limit (203ms)
    ✔ Bulk exiting 10 validator (10 operators cluster) gas limit (230ms)
    ✔ Bulk exiting 10 validator (13 operators cluster) gas limit (263ms)
    ✔ Bulk exiting removed validators reverts "IncorrectValidatorStateWithData" (168ms)
    ✔ Bulk exiting non-existing validators reverts "IncorrectValidatorStateWithData" (170ms)
    ✔ Bulk exiting validators with empty operator list reverts "IncorrectValidatorStateWithData" (170ms)
    ✔ Bulk exiting validators with empty public key reverts "ValidatorDoesNotExist" (164ms)
    ✔ Bulk exiting validators using the wrong account reverts "IncorrectValidatorStateWithData" (166ms)
    ✔ Bulk exiting validators with incorrect operators (unsorted list) reverts with
"IncorrectValidatorStateWithData" (165ms)

  Register Validator Tests
    ✔ Register validator with 4 operators emits "ValidatorAdded"
    ✔ Register validator with 4 operators gas limit
    ✔ Register 2 validators into the same cluster gas limit (50ms)
    ✔ Register 2 validators into the same cluster and 1 validator into a new cluster gas limit (54ms)
    ✔ Register 2 validators into the same cluster with one time deposit gas limit
    ✔ Bulk register 10 validators with 4 operators into the same cluster (176ms)
    ✔ Bulk register 10 validators with 4 operators new cluster (157ms)
    ✔ Register validator with 7 operators gas limit
    ✔ Register 2 validators with 7 operators into the same cluster gas limit (48ms)
    ✔ Register 2 validators with 7 operators into the same cluster and 1 validator into a new cluster
with 7 operators gas limit (72ms)
    ✔ Register 2 validators with 7 operators into the same cluster with one time deposit gas limit (45ms)
    ✔ Bulk register 10 validators with 7 operators into the same cluster (215ms)
    ✔ Bulk register 10 validators with 7 operators new cluster (188ms)
```

```
    ✔ Register validator with 10 operators gas limit
    ✔ Register 2 validators with 10 operators into the same cluster gas limit (51ms)
    ✔ Register 2 validators with 10 operators into the same cluster and 1 validator into a new cluster
with 10 operators gas limit (75ms)
    ✔ Register 2 validators with 10 operators into the same cluster with one time deposit gas limit
(50ms)
    ✔ Bulk register 10 validators with 10 operators into the same cluster (247ms)
    ✔ Bulk register 10 validators with 10 operators new cluster (220ms)
    ✔ Register validator with 13 operators gas limit
    ✔ Register 2 validators with 13 operators into the same cluster gas limit (58ms)
    ✔ Register 2 validators with 13 operators into the same cluster and 1 validator into a new cluster
with 13 operators gas limit (85ms)
    ✔ Register 2 validators with 13 operators into the same cluster with one time deposit gas limit
(54ms)
    ✔ Bulk register 10 validators with 13 operators into the same cluster (284ms)
    ✔ Bulk register 10 validators with 13 operators new cluster (254ms)
    ✔ Get cluster burn rate
    ✔ Get cluster burn rate when one of the operators does not exist (1087ms)
    ✔ Register validator with incorrect input data reverts "IncorrectClusterState" (38ms)
    ✔ Register validator in a new cluster with incorrect input data reverts "IncorrectClusterState"
    ✔ Register validator when an operator does not exist in the cluster reverts "OperatorDoesNotExist"
    ✔ Register validator with a removed operator in the cluster reverts "OperatorDoesNotExist"
    ✔ Register cluster with unsorted operators reverts "UnsortedOperatorsList"
    ✔ Register cluster with duplicated operators reverts "OperatorsListNotUnique"
    ✔ Register validator with not enough balance reverts "InsufficientBalance"
    ✔ Register validator in a liquidatable cluster with not enough balance reverts "InsufficientBalance"
(44ms)
    ✔ Register an existing validator with same operators setup reverts "ValidatorAlreadyExistsWithData"
    ✔ Register an existing validator with different operators setup reverts
"ValidatorAlreadyExistsWithData"
    ✔ Register validator with an empty public key reverts "InvalidPublicKeyLength"
    ✔ Bulk register 10 validators with empty public keys list reverts "EmptyPublicKeysList"
    ✔ Bulk register 10 validators with different pks/shares lenght reverts
"PublicKeysSharesLengthMismatch" (110ms)
    ✔ Bulk register 10 validators with wrong operators length reverts "InvalidOperatorIdsLength" (140ms)
    ✔ Bulk register 10 validators with empty operators list reverts "InvalidOperatorIdsLength" (140ms)
    ✔ Retrieve an existing validator
    ✔ Retrieve a non-existing validator

  Remove Validator Tests
    ✔ Remove validator emits "ValidatorRemoved"
    ✔ Bulk remove validator emits "ValidatorRemoved" (164ms)
    ✔ Remove validator after cluster liquidation period emits "ValidatorRemoved"
    ✔ Remove validator gas limit (4 operators cluster)
    ✔ Bulk remove 10 validator gas limit (4 operators cluster) (163ms)
    ✔ Remove validator gas limit (7 operators cluster)
    ✔ Bulk remove 10 validator gas limit (7 operators cluster) (192ms)
    ✔ Remove validator gas limit (10 operators cluster) (51ms)
    ✔ Bulk remove 10 validator gas limit (10 operators cluster) (227ms)
    ✔ Remove validator gas limit (13 operators cluster) (57ms)
    ✔ Bulk remove 10 validator gas limit (13 operators cluster) (258ms)
    ✔ Remove validator with a removed operator in the cluster
    ✔ Register a removed validator and remove the same validator again
    ✔ Remove validator from a liquidated cluster
    ✔ Remove validator with an invalid owner reverts "ClusterDoesNotExists"
    ✔ Remove validator with an invalid operator setup reverts "ClusterDoesNotExists"
    ✔ Remove the same validator twice reverts "ValidatorDoesNotExist"
    ✔ Remove the same validator with wrong input parameters reverts "IncorrectClusterState"
    ✔ Bulk Remove validator that does not exist in a valid cluster reverts
"IncorrectValidatorStateWithData" (161ms)
    ✔ Bulk remove validator with an invalid operator setup reverts "ClusterDoesNotExists" (159ms)
    ✔ Bulk Remove the same validator twice reverts "IncorrectValidatorStateWithData" (169ms)
    ✔ Remove validators from a liquidated cluster (163ms)
    ✔ Bulk remove 10 validator with duplicated public keys reverts "IncorrectValidatorStateWithData"
(167ms)
    ✔ Bulk remove 10 validator with empty public keys reverts "IncorrectValidatorStateWithData"

  Register Validator Tests
    Generic Tests
      ✔ Register whitelisted validator in 1 operator with 4 operators emits "ValidatorAdded"/gas
limits/logic
      ✔ Register whitelisted validator in 4 operators in 4 operators cluster gas limits/logic
```

```
      ✔ Register non-whitelisted validator in 1 public operator with 4 operators emits
"ValidatorAdded"/logic
      ✔ Register whitelisted validator in 4 operator in 4 operators existing cluster gas limits (39ms)
      ✔ Register using non-authorized account for 1 operator with 4 operators cluster reverts
"CallerNotWhitelistedWithData"
      ✔ Register using non-authorized account for 1 operator with 4 operators cluster reverts
"CallerNotWhitelistedWithData"
      ✔ Register using fake whitelisting contract reverts
      ✔ Read-only reentrancy attack reverts (44ms)
      Register using whitelisting contract
        ✔ Register using whitelisting contract and SSV whitelisting module for 2 operators
        ✔ Register using whitelisting contract for 1 operator in 4 operators cluster gas
limits/events/logic
        ✔ Bulk register 10 validators using whitelisting contract for 1 operator in 4 operators cluster
gas limits/logic (180ms)
        ✔ Register using whitelisting contract for 1 public operator in 4 operators cluster
        ✔ Register using whitelisting contract for 1 operator & EOA for 1 operator in 4 operators cluster
        ✔ Register using whitelisting contract with an unauthorized account reverts
"CallerNotWhitelistedWithData"
        ✔ Register using whitelisting contract but a public operator allows registration
    Whitelist Edge Cases Tests
      ✔ WT-1 - Register validator, 13 whitelisted operators, 1 block index each (4652ms)
      ✔ WT-2 - Register 2 validators using 2 accounts and remove the first (52ms)


  348 passing (54s)


  Whitelisting Tests (fork) - Pre-upgrade SSV Core Contracts Tests
    ✔ Check an existing whitelisted operator is whitelisted but not using an external contract (2177ms)
    ✔ Register with an operator that uses a non-whitelisting contract reverts
"InvalidWhitelistingContract" (7342ms)
    ✔ Register using legacy whitelisted operators in 4 operators cluster events/logic (1639ms)
    ✔ Replace a whitelisted address by an external whitelisting contract (63ms)
    ✔ Whitelist multiple operators for an already whitelisted operator (1099ms)

  Whitelisting Tests (fork) - Ongoing SSV Core Contracts upgrade Tests
    ✔ WT-3 - Check backward compatibility with existing generic contracts (4923ms)


  6 passing (2m)
```

Code Coverage

The code coverage was generated using `SOLIDITY_COVERAGE=true NO_GAS_ENFORCE=1 npx hardhat coverage`. It is recommended to enhance the test suite by adding additional assertions to validate the expected outcomes after specific contract function calls, ensuring all conditions and edge cases are thoroughly tested. The client can consider using SuMo, a mutation testing tool for solidity smart contracts to further enhance the testsuite quality.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	97.01	71.88	95.45	97.5	
SSVNetwork.sol	97.62	80.77	97.5	98.11	315
SSVNetworkViews.sol	96	33.33	92	96.15	168
SSVProxy.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
ISSVClusters.sol	100	100	100	100	
ISSVDAO.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
ISSVNetwork.sol	100	100	100	100	
ISSVNetworkCore.sol	100	100	100	100	
ISSVOperators.sol	100	100	100	100	
ISSVOperatorsWhitelist.sol	100	100	100	100	
ISSVViews.sol	100	100	100	100	
contracts/interfaces/external/	100	100	100	100	
ISSVWhitelistingContract.sol	100	100	100	100	
contracts/libraries/	98.28	82.08	97.44	95.48	
ClusterLib.sol	100	75	100	100	
CoreLib.sol	77.78	50	80	69.23	10,15,21,44
OperatorLib.sol	100	90.38	100	96.97	84,95,124
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	80	100	100	
contracts/modules/	99.31	92.98	98.21	98.96	
SSVClusters.sol	100	93.18	100	99.29	101
SSVDAO.sol	100	100	100	100	
SSVOperators.sol	100	97.22	100	100	
SSVOperatorsWhitelist.sol	95.45	75	100	92.86	59,61
SSVViews.sol	98.75	88.46	95.45	98.95	312
All files	98.73	85.71	96.89	97.82	

Changelog

- 2024-06-19 - Initial report
- 2024-06-27 - Final report
- 2024-07-03 - Final report update
- 2024-07-04 - Final report update

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

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

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.