Quantstamp Security Assessment Certificate

November 5th 2020 — Quantstamp Verified

Teku

This security assessment was prepared by Quantstamp, the leader in blockchain security

Executive Summary

Type	Ethereum 2.0 Client
Auditors	Jan Gorzny, Blockchain Researcher Jake Goh Si Yuan, Research Engineer Poming Lee, Research Engineer Leonardo Passos, Senior Research Engineer Kacper Bąk, Senior Research Engineer Martin Derka, Senior Research Engineer
Timeline	2020-08-31 through 2020-10-21
Languages	Java, Kotlin
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Ethereum 2.0 Specification Teku Manual
Documentation Quality	High
Test Quality	High

Source Code

Repository	Commit
teku	a3f6ed9
signers	1069ade
jvm-libp2p	3fd9dd9
discovery	840e90b

Goals

• Review vulnerabilities related to denial-of-service conditions or attacks, data integrity loss, and exfiltration of private keys.

• Investigate the possibility of unwanted, unexpected or remote code execution, including unspecified or unexpected behaviours, underflows, and overflows.

• Review the possibility of consensus split scenarios, or those which may result in halted pools, reduced rewards, or increased penalties.

• Review any operational threats related to the node: configuration and deployment scripts; dependencies.

Total Issues	**29** (21 Resolved)
High Risk Issues	**1** (1 Resolved)
Medium Risk Issues	**8** (6 Resolved)
Low Risk Issues	**8** (5 Resolved)
Informational Risk Issues	**11** (8 Resolved)
Undetermined Risk Issues	**1** (1 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	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.

Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
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).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

Quantstamp has reviewed the Teku ETH 2.0 client implementation. The audit has revealed several issues of various severity within the code, although all issues have since been acknowledged, fixed, or mitigated. The code of the client is well-written and easy to follow, although it could be further improved to follow all best practices in some places. In addition to the available ETH 2.0 specifications, the code has some in-line documentation and external documentation to assist with future development and other contributors. Many issues arose likely due to the size of the codebase, and no deviations from the ETH 2.0 specification were found, though this was not always perfectly clear.
Most packages within the repositories include tests. Coverage was not computed for the system as a whole - rather, it was pieced together from running coverage on each package and putting them together (however this is done using the provided packages); as a result, it may not be fully accurate.
Additionally, the reader should note that the audit was conducted under the assumption that the execution environment (JVM) is trusted. This is a very sensible requirement, but it needs to be understood that as the JVM resides on the machine, an unauthorized attacker could replace it with a malicious implementation and completely change the semantics of the Java code being executed. Similarly, through unauthorized access to compiled Teku jar packages, an attacker could inject malicious classes and code into the package itself. It is imperative that all parties operating Teku ensure that their environment is private and does not provide access to untrusted parties.
Finally, note that no file hashes appear in this report; refer to the commit for each repository for a signature of files audited in this report.

ID	Description	Severity	Status
QSP-1	Unlimited Inbound Message Queue	High	Fixed
QSP-2	Critical Component Dependent on Unaudited External Library	Medium	Acknowledged
QSP-3	DDoS Attack Vector Through A Mapping Between Public Keys and Validators' IPs	Medium	Mitigated
QSP-4	Method `processNewData()` Does Not Check Deposit Index	Medium	Fixed
QSP-5	Open Admin Endpoint	Medium	Acknowledged
QSP-6	Message Encoding is Changeable	Medium	Fixed
QSP-7	`beacon_block` Logic Does Not Reject Blocks that Fail Ancestry Condition	Medium	Fixed
QSP-8	`beacon_aggregate_and_proof` Logic Does Not Reject Aggregates that Fail Ancestry Condition	Medium	Fixed
QSP-9	`beacon_attestation_{subnet_id}` Logic Does Not Reject Attestations that Fail Ancestry Condition	Medium	Fixed
QSP-10	Weak Passwords Allowed for Validator Accounts	Low	Fixed
QSP-11	Possible Loss of Precision	Low	Acknowledged
QSP-12	Generic Exception Caught	Low	Fixed
QSP-13	Incorrect Custom Equality and Hash Functions	Low	Fixed
QSP-14	Possible `NullPointerException`	Low	Fixed
QSP-15	Users Behind a NAT Must Configure Their Inbound Traffic Setup	Low	Acknowledged
QSP-16	Gossip Parameter `seen_ttl` Missing in Current Implementation	Low	Fixed
QSP-17	Non-Deterministic Filter Leads to Non-Deterministic Behaviour	Low	Acknowledged
QSP-18	Beta Imports	Informational	Acknowledged
QSP-19	No Call Rate Limiting	Informational	Acknowledged
QSP-20	Non-Final Variable	Informational	Fixed
QSP-21	`beacon_aggregate_and_proof` Has Stricter Committee Size Validation	Informational	Fixed
QSP-22	Incompatibility Between Gradle and Java 13	Informational	Fixed
QSP-23	Noise Library is Cloned	Informational	Fixed
QSP-24	Multicast DNS is a clone from JMDNS	Informational	Acknowledged
QSP-25	Clone method does not copy entire state	Informational	Fixed
QSP-26	jvm-libp2p Project `README.md` Has No Build Information	Informational	Fixed
QSP-27	Ignored `InterruptedException`	Informational	Fixed
QSP-28	Clone May Return `null`	Informational	Fixed
QSP-29	Reads Without Locks	Undetermined	Fixed

Quantstamp Audit Breakdown

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

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

- Timestamp dependence
- Mishandled exceptions and call stack limits
- Integer overflow / underflow
- Number rounding errors
- Cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Business logic contradicting the specification
- Code clones, functionality duplication

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the codebase.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. 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:
   i. 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.
   ii. Static analysis to detect common bugs and issues.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, 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

QSP-1 Unlimited Inbound Message Queue

Severity: *High Risk*

Status: Fixed

Description: According to the Eth2 p2p spec, *"for any optional queueing, clients SHOULD maintain maximum queue sizes to avoid DoS vectors"*. Currently, Teku queues inbound messages indirectly by running their processing function asynchronously; there does not seem to exist any limitation on the maximum size of the queue. Without such a control, the node is potentially subject to DoS attacks.

Recommendation: Following the Eth2 p2p specification to avoid a DoS attack, control the size of the inbound message queue.
Update: A limit has been added to the gossip queues, and changes have been made to how errors are handled.

QSP-2 Critical Component Dependent on Unaudited External Library

Severity: *Medium Risk*

Status: Acknowledged

Description: The `bls` module has two implementations of BLS. One of them is based on the unaudited `blst` library: supranational/blst: BLS12-381 signature library (https://github.com/supranational/blst).
This might introduce additional attack vectors.

Recommendation: Such dependencies are often unavoidable, but the users of this software should be made explicitly aware of them.
Update: The documentation for Teku has been updated to note that this library has not yet been audited.

QSP-3 DDoS Attack Vector Through A Mapping Between Public Keys and Validators' IPs

Severity: *Medium Risk*

Status: Mitigated

Description: It is possible to DDoS the system by attacking block proposers and/or committee members. Such an attack would rely on a mapping between validators' public keys and their IP addresses. Furthermore, one can detect the roles of a validator with a certain public key based on the tables of validator indices.
Once an attacker has the mapping of public keys and IPs, the attacker can target them based on their roles. The more complete the mapping is, the higher is the possibility for an attacker to DDoS the system from proposing new blocks. There's no need, however, for this mapping to be complete in order to harm the system. When an attacker learns the IP of a validator, the attacker could have the validator slashed by DDoSing the validator when it plays a role of a committee member.
Following previous audit reports on the Eth2 specification, it has been reported that an attacker can link IP addresses with block proposers, giving him the ability to DoS a proposer of a slot to stall the chain, or to keep slots empty. For further information, please refer to:

- https://medium.com/prysmatic-labs/quantstamp-security-audit-results-for-the-prysm-eth2-client-7f949c6c866f
- https://leastauthority.com/static/publications/LeastAuthority-Ethereum-2.0-Specifications-Audit-Report.pdf

Exploit Scenario: The mapping between public key and the IP of a validator could be learned by checking all the attestations sent in by the other peers. If a peer sent in an attestation which is

not an aggregated attestation, one can assume that the peer is the attester of that attestation. Thus, we know the peer's IP and also their public key.

Recommendation: We recommend designing a mechanism in which attesters send only aggregated attestations. The more attestations are in the aggregated attestations, the harder it is for an attacker to learn the mapping between the public key of a peer and its IP.
Currently, the issue cannot be fully resolved until the Eth2 spec incorporates a secret election mechanism.
Update: Teku has a few mechanisms to mitigate attempted DoS attacks:

- At the gossip level, there is a limited thread pool which will begin dropping messages if the node receives more than it can process. See here.
- Teku has a `Firewall` class to close out stale connections.
- At the RPC request level, Teku has a `RateTracker` that is used to throttle requests.

QSP-4 Method `processNewData()` Does Not Check Deposit Index

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `ethereum/statetransition/src/main/java/tech/pegasys/teku/statetransition/genesis/GenesisHandler.java`

Description: The function `processNewData()` doesn't check the deposit index which may result in wrong synchronization and hence the client may be out of sync with the network.

Recommendation: We recommend adding an index validation for the first deposit in the list.
Update: The recommended index check has been added.

QSP-5 Open Admin Endpoint

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `PutLogLevel.java`

Description: This is an admin endpoint, but it does not seem to be authenticated in any way. As it stands, it appears that an unauthorized party would be able to change the logging configuration for the application. While changing the logging level does not seem to expose any sensitive information, it is an undesirable action that should be reserved only to authorized admins.

Recommendation: Require authentication for access.
Update: The REST APIs are disabled by default - users have to take an active step to enable these and we now have warnings in the docs related to exposing these publicly. Running on shared hardware accessible to untrusted users falls under the category of "publicly exposing" these endpoints. Moreover, the REST API's are intended for trusted users. By default, the REST API only accepts incoming requests from localhost so users should be generally protected. Documentation has been added to make it clearer that these API's should not be publicly exposed.

QSP-6 Message Encoding is Changeable

Severity: *Medium Risk*

Status: Fixed

Description: According to specification implementations MUST use a single encoding. Changing an encoding will require coordination between participating implementations.
While the current testnet is using a single encoding, clients are currently allowed to switch encodings for a given client from regular SSZ to Snappy, which could lead to network partitions.

Recommendation: The risk could be mitigated by better documentation and defining a strategy for mainnet transition. When transitioning to mainnet that uses Snappy, two options are possible: a fallback support for non-snappy, or a coordination of network participants to switch to Snappy and changing the implementation to use Snappy by default. For further discussion on this topic, please refer to https://github.com/ethereum/eth2.0-specs/issues/1931.
Update: This has been fixed by removing the ability to use Snappy.

QSP-7 `beacon_block` Logic Does Not Reject Blocks that Fail Ancestry Condition

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `networking/eth2/src/main/java/tech/pegasys/teku/networking/eth2/gossip/topics/validation/BlockValidator.java`

Description: The validation logic for the `beacon_block` topic does not reject blocks when they DO NOT satisfy the following condition, as per Eth2 p2p spec:
"[REJECT] The current finalized_checkpoint is an ancestor of block – i.e. get_ancestor(store, block.parent_root, compute_start_slot_at_epoch(store.finalized_checkpoint.epoch)) == store.finalized_checkpoint.root"
As such, Teku's client will queue blocks that other clients from other implementations would immediately reject.

Recommendation: Strictly adhere to the rejection logic stated in the Eth2 p2p specification. Instead of queueing the block, reject it right away.
Update: This has been fixed.

QSP-8 `beacon_aggregate_and_proof` Logic Does Not Reject Aggregates that Fail Ancestry Condition

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `networking/eth2/src/main/java/tech/pegasys/teku/networking/eth2/gossip/topics/validation/AttestationValidator.java`

Description: The validation logic for the `beacon_aggregate_and_proof` topic does not reject aggregates when they do not satisfy the following condition, as per Eth2 p2p spec:
"[REJECT] The current finalized_checkpoint is an ancestor of the block defined by aggregate.data.beacon_block_root – i.e. get_ancestor(store, aggregate.data.beacon_block_root, compute_start_slot_at_epoch(store.finalized_checkpoint.epoch)) == store.finalized_checkpoint.root"
As such, Teku's client will queue aggregates that other clients from other implementations would immediately reject.

Recommendation: Strictly adhere to the rejection logic stated in the Eth2 p2p specification. Instead of queueing the aggregate, reject it right away.
Update: This has been fixed.

QSP-9 `beacon_attestation_{subnet_id}` Logic Does Not Reject Attestations that Fail Ancestry Condition

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `networking/eth2/src/main/java/tech/pegasys/teku/networking/eth2/gossip/topics/validation/AttestationValidator.java`

Description: The validation logic for the `beacon_aggregate_and_proof` topic does not reject attestations when they do not satisfy the following condition, as per Eth2 p2p spec:
"[REJECT] The current finalized_checkpoint is an ancestor of the block defined by attestation.data.beacon_block_root – i.e. get_ancestor(store, attestation.data.beacon_block_root, compute_start_slot_at_epoch(store.finalized_checkpoint.epoch)) == store.finalized_checkpoint.root"
As such, Teku's client will queue attestations that other clients from other implementations would immediately reject.

Recommendation: Strictly adhere to the rejection logic stated in the Eth2 p2p specification. Instead of queueing the attestation, reject it right away.
Update: This has been fixed.

QSP-10 Weak Passwords Allowed for Validator Accounts

Severity: *Low Risk*

Status: Fixed

Description: The node allows weak passwords for validator accounts. Validators might have their passwords brute-forced in a relatively small time frame if poor passwords are chosen.

Recommendation: Enforcing a strong password policy on the client side in order to prohibit weak passwords.
Update: Teku has altogether removed our tools that generate validator keys (`teku validator generate, etc`) given a supplied password.
Teku does accept encrypted keystore files with passwords that have been generated outside of teku. It is out of scope to enforce any password constraints at the point where Teku is ingesting already encrypted files. Teku has decided to deprecate these teku validator commands for a short time before removing them altogether.

QSP-11 Possible Loss of Precision

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `validator/coordinator/src/main/java/tech/pegasys/teku/validator/coordinator/DepositProvider.java`

Description: In lines 169 and 170 the function `getDepositsWithProof()` casts `long`s to `int`s. For a number of deposits large enough this may result in a loss of precision.

Recommendation: We recommend verifying whether `int` provides a realistic upper limit on the number of deposits.
Update: This is not an issue. Teku's response: "Formally, the maximum number of deposits defined in the deposit contract is 2^32-1, for which an unsigned integer would be fine, but not necessarily a signed integer, as used here. Nonetheless, the actual upper bound on deposits is the amount of Ether in circulation, since a deposit must be at least one Ether. The current Eth in circulation is less than 113 million and not expected ever to increase by as much as an order of magnitude from this level. Thus, the casts to integers for the deposit index and the deposit count are safe here."

QSP-12 Generic Exception Caught

Severity: *Low Risk*

Status: Fixed

Description: `discovery.util.Functions`: `aesgcm_encrypt` catches `Exception`, which is too general.

Recommendation: The caught exception should be narrowed down to a specific exception type.
Update: This has been fixed.

QSP-13 Incorrect Custom Equality and Hash Functions

Severity: *Low Risk*

Status: Fixed

Description: Several classes have issues with overriding their `equals` and/or `hashCode` functions.

- `org.ethereum.beacon.discovery.message.NodesMessage`: `equals` and `hashCode` do not take in the collection of nodes, just its size.
- `DiscoveryV5Message`: the `equals` and `hashCode` methods are not overriden.
- `src/main/java/io/libp2p/discovery/mdns/impl/DNSRecord.java`: `equals` is overriden, but `hashCode` is not.

According to the Java Language Specification, there is a contract between `equals(Object)` and `hashCode()`:
"If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result. It is not required that if two objects are unequal according to the equals(java.lang.Object) method, then calling the hashCode method on each of the two objects must produce distinct integer results. However, the programmer should be aware that producing distinct integer results for unequal objects may improve the performance of hashtables. In order to comply with this contract, those methods should be either both inherited, or both overridden."

Recommendation: Correctly implement the `equals` and `hashCode` functions in the listed classes, taking care to ensure that class members have their equality checked correctly.
Update: This has been fixed (see here, here, and here).

QSP-14 Possible `NullPointerException`

Severity: *Low Risk*

Status: Fixed

Description: `org.ethereum.beacon.discovery.message.handler.PingHandler` Line 31 calls `.get()` on `Optional<>` without checking the value presence, can result in a `NoSuchElementException`.

Recommendation: Check if the value could be null.
Update: This is no longer relevant.

QSP-15 Users Behind a NAT Must Configure Their Inbound Traffic Setup

Severity: *Low Risk*

Status: Acknowledged

Description: According to the Eth2 p2p spec:
“Nodes operating behind a NAT, or otherwise undialable by default (e.g. container runtime, firewall, etc.), MUST have their infrastructure configured to enable inbound traffic on the announced public listening endpoint”
This issue is not supposed to be handled by the existing code, but rather by instructing node operators on how to configure their deployment. In the given repository, we did not find any information related to the given matter. If users are unaware of this issue, the node will not receive any inbound traffic on its announced public endpoint.

Recommendation: Make users aware of this issue, instructing them or providing them with pointers on how to address this issue.
Update: Documentation has been updated for this issue.

QSP-16 Gossip Parameter `seen_ttl` Missing in Current Implementation

Severity: *Low Risk*

Status: Fixed

Description: According to the Eth2 p2p spec, the gossipsub implementation should support a set of parameters, including `seen_ttl`, which controls the number of heartbeat intervals to retain message IDs. Moreover, `seen_ttl`should be set to 550. This is not yet implemented in the current libp2p code. The existing control in place is limited to a Least Replacement Policy caching, where up to 10K messages will be stored at a time.

Recommendation: Following the Eth2 p2p spec, add and support the `seen_ttl` parameter, setting it to 550.
Update: This hsa been fixed.

QSP-17 Non-Deterministic Filter Leads to Non-Deterministic Behaviour

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `networking/p2p/src/main/peer/DisconnectReason.java`

Description: On line 25 and 26, `Remote_fault` and `Remote_unresponsive` share the same `enum` number. Therefore, `fromReasonCode` will have `.filter` provide a stream of `REMOTE_FAULT` and `UNRESPONSIVE`, and `findAny` will non-deterministically provide out `REMOTE_FAULT` or `UNRESPONSIVE`for code `3`.

Recommendation: Change the use of `filter` or the appropriate `enum` numbers.
Update: The Teku team has determined that this is not an issue.

QSP-18 Beta Imports

Severity: *Informational*

Status: Acknowledged

Description: In `BeaconRestApi`, `import 'com.google.common.io.Resources'` is marked unstable with `@Beta`.

Recommendation: Note, explicitly, that unstable libraries are used, so that users are aware of them.
Update: The Teku team has stated that "Teku is an application - these beta imports mean potentially more work for us (developers) to upgrade in the future, but should not directly affect users."

QSP-19 No Call Rate Limiting

Severity: *Informational*

Status: Acknowledged

Description: There is no rate limiting on the API/RPC calls. While this may be intentional, excessive calls may result in poor or denied service.

Recommendation: Limit the rate at which the node will respond to such calls.
Update: The REST API's are intended for trusted users. By default, the REST API only accepts incoming requests from localhost so users should be generally protected. Teku has added warnings to our documentation to alert users to the fact that these API's are not meant to be publicly exposed.
Additionally, the REST API's are disabled by default. So a user would have to take active steps to enable these endpoints and should be following guidance in the documentation around not exposing these endpoints to the public.

QSP-20 Non-Final Variable

Severity: *Informational*

Status: Fixed

Description: `org.ethereum.beacon.discovery.pipeline.Envelope` Line 13: UUID id is not `final`.

Recommendation: The UUID id should be made `final`.
Update: This has been fixed.

QSP-21 `beacon_aggregate_and_proof` Has Stricter Committee Size Validation

Severity: *Informational*

Status: Fixed

Description: According to the Eth2 p2p spec, “The attestation has participants – that is, len(get_attesting_indices(state, aggregate.data, aggregate.aggregation_bits)) >= 1”. If not, the attestation should be rejected. The code, however, performs the following check:

```
final List<Integer> committee =
    get_beacon_committee(
        state, attestation.getData().getSlot(), attestation.getData().getIndex());
if (committee.size() != attestation.getAggregation_bits().getCurrentSize()) {
  return REJECT;
}
```

This is not the same as stated in the specification; it is a stricter condition: it checks that the number of participants in the attestation is the same as the committee size (a number greater than zero). Although it seems a correct check to make, it does not find a match in the specification.

Recommendation: Review if the implemented check is indeed intended and if so, document the code properly, explaining why it should be as implemented, instead of what the spec states.
Update: An explanation has been added.

QSP-22 Incompatibility Between Gradle and Java 13

Severity: *Informational*

Status: Fixed

Description: When attempting to build the `libp2p/jvm-libp2p.git` repository using a Java 13 setup, Gradle reports the following error:

```
FAILURE: Build failed with an exception.

* Where:
Settings file '/Users/developer/Documents/Projects/audits-code/teku/jvm-libp2p/settings.gradle'

* What went wrong:
Could not compile settings file '/Users/developer/Documents/Projects/audits-code/teku/jvm-libp2p/settings.gradle'.
> startup failed:
  General error during semantic analysis: Unsupported class file major version 57

  java.lang.IllegalArgumentException: Unsupported class file major version 57
        at groovyjarjarasm.asm.ClassReader.<init>(ClassReader.java:184)
        at groovyjarjarasm.asm.ClassReader.<init>(ClassReader.java:166)
        at groovyjarjarasm.asm.ClassReader.<init>(ClassReader.java:152)
        at groovyjarjarasm.asm.ClassReader.<init>(ClassReader.java:273)
        at org.codehaus.groovy.ast.decompiled.AsmDecompiler.parseClass(AsmDecompiler.java:81)
        at org.codehaus.groovy.control.ClassNodeResolver.findDecompiled(ClassNodeResolver.java:254)
        at org.codehaus.groovy.control.ClassNodeResolver.tryAsLoaderClassOrScript(ClassNodeResolver.java:192)
        at org.codehaus.groovy.control.ClassNodeResolver.findClassNode(ClassNodeResolver.java:172)
        at org.codehaus.groovy.control.ClassNodeResolver.resolveName(ClassNodeResolver.java:128)
      [...]
```

The error above stems from the fact the Gradle version in use (5.1.4) does not support Java 13 - see issue 10785 in their Github repository. Support for Java 13 is planned for Gradle 6+. If the build is broken or with missing supporting documentation, external contributors and users seeking to validate and/or experiment with unstable releases may be discouraged to do so.

Recommendation: In the project `README.md` file, explicitly state that only Java 11 is supported.
Update: This has been fixed.

QSP-23 Noise Library is Cloned

Severity: *Informational*

Status: Fixed

Description: Classes:

- com.southernstorm.noise.crypto.*
- com.southernstorm.noise.protocol.*

The Noise library dependency is not managed by Gradle. Instead, its code has been cloned from (https://github.com/rweather/noise-java), without any reference to what commit head was used. As a negative side-effect, any updates or vulnerability fixes in the upstream repository must be manually tracked, pulled, and merged by Teku developers, which if not done regularly, could lead to bugs or security issues being left unfixed.

Recommendation: At the very least, document the commit hash from which the code was taken from. Optimally, we recommend adding the Noise library as a dependency in your setup. See the corresponding POM file.
Update: The library has been added as a dependency.

QSP-24 Multicast DNS is a clone from JMDNS

Severity: *Informational*

Status: Acknowledged

Description: Class: io.libp2p.discovery.mdns.*
The minimal multicast DNS implementation is a clone of certain parts of the JMDNS library (see https://github.com/jmdns/jmdns), without any reference to what commit head was used. As a negative side-effect, any updates or vulnerability fixes in the upstream repository must be manually tracked, pulled, and merged by Teku developers, which if not done regularly, could lead to bugs or security issues being left unfixed.

Recommendation: At the very least, document the commit hash from which the code was taken from. Optimally, we recommend managing JMDNS as a dependency in Gradle.
Update: Multicast DNS will not be used in Eth2 setup and should not influence Teku's client. Nonetheless, it is part of the reusable libp2p library, which could target other applications.

QSP-25 Clone method does not copy entire state

Severity: *Informational*

Status: Fixed

File(s) affected: `src/main/java/io/libp2p/discovery/mdns/impl/ServiceInfoImpl.java`

Description: The given `clone()` implementation does not set the `_server` instance variable after a new object is created (L381).

Recommendation: There are at least two possible fixes:

- Explicitly set `_server = text` in the `clone()` method, or;
- Copy the constructor on line 64 into a new constructor, adding `server` as an additional parameter. In the newly created constructor, in addition to the previously set fields, add `_server = server`. Then, on line 381, invoke the new constructor.

Update: Multicast DNS will not be used in Eth2 setup and should not influence Teku's client. Nonetheless, it is part of the reusable libp2p library, which could target other applications. This has been changed here.

QSP-26 jvm-libp2p Project `README.md` Has No Build Information

Severity: *Informational*

Status: Fixed

Description: The `README.md` file does not provide instructions on how to build the project. When lacking documentation, external contributors and users seeking to validate and/or experiment with unstable releases may be discouraged to do so.

Recommendation: Document all build targets, and how to invoke the Gradle wrapper accordingly.
Update: Build information can be found here and here.

QSP-27 Ignored `InterruptedException`

Severity: *Informational*

Status: Fixed

File(s) affected: `src/main/java/io/libp2p/discovery/mdns/impl/JmDNSImpl.java`

Description: On Line 400, `InterruptedException` handling is simply ignored. The throwing of `InterruptedException` clears the interrupted state of the `Thread`, so if the exception is not handled properly the fact that the thread was interrupted will be lost.

Recommendation: An `InterruptedException` exception should either be rethrown - immediately or after cleaning up the method's state - or the thread should be re-interrupted by calling `Thread.interrupt()`. Any other course of action risks delaying thread shutdown and loses the information that the thread was interrupted - probably without finishing its task.
Update: Multicast DNS will not be used in Eth2 setup and should not influence Teku's client. Nonetheless, it is part of the reusable libp2p library, which could target other applications. This has been changed here.

QSP-28 Clone May Return `null`

Severity: *Informational*

Status: Fixed

File(s) affected: `src/main/java/io/libp2p/discovery/mdns/ServiceInfo.java`

Description: The catch block inside `clone()` returns `null`; however, returning `null` contravenes the method's implicit contract.

Recommendation: If object is not cloneable, do not return `null`; instead, let `CloneNotSupportedException` be thrown.
Update: Multicast DNS will not be used in Eth2 setup and should not influence Teku's client. Nonetheless, it is part of the reusable libp2p library, which could target other applications. This has been fixed here.

QSP-29 Reads Without Locks

Severity: *Undetermined*

Status: Fixed

Description: `tech.pegasys.teku.storage.store.Store` methods `equals` and `hashCode` access fields of objects without locks. The locks are needed in getters.

Recommendation: Quantstamp, based on the mutual discussion, recommends removing the methods.
Update: This has been fixed.

Adherence to Best Practices

1. There are `TODO` items in the code which should be fixed or removed. Among possibly others, these are located in:
    1. `validator/remote/src/main/java/tech/pegasys/teku/validator/remote/WebSocketBeaconChainEventAdapter.java` Line 36.
    2. `discovery.type.Hashes`
    3. `data/provider/src/main/java/ValidatorDataProvider.java`
2. `PeerCommand.java`: Line 82 declares `thrown IOException`, but it seems that this exception cannot be thrown.
3. `HostAllowListUtil.java` is parsing host records by splitting a string. A library that would be better suited must exist; if not, a regular expression would be better.
4. `HostAllowListUtil.java`: Line 25 `return` can be simplified to return the condition.
5. `GetValidators.java`: `getResultProcessor` is `private` and `final`, but `final` is not needed.
6. `BeaconRestApi.java`: Lines 86 and 87 get `app.server()`, but `server()` is annotated with `@Nullable`, so they can produce a `NullPointerException`. It would be better to test that the server is not null.
7. `package tech.pegasys.teku.storage.store.Store`: A constructor with so many parameters as on line 83 is not a best practice. The builder pattern should be used instead.
8. `rocksdb.serialization.RocksDbSerializer`: the line 37 public key serializer is not used (implemented). Consider deleting it.
9. `discovery.util.Utils`: Most of the functions are not used, consider removing.
10. `discovery.util.Utils`: Line 48: Can use a diamond operator instead of explicit listing of generics.
11. `discovery.util.RlpUtil`: `RlpDecodeException` is `RuntimeException`, so it does not need to declared on line 160.
12. `discovery.type.Hashes`: Line 42: Too broad `Exception` trap re-thrown as `RuntimeException`. This should be more specific so that the specific exception can be caught downstream.
13. `type.Hashes`: Line 34: The algorithm is suppressed and in fact, unused. The public method uses SHA256 constant, but hides it from the caller. Consider removing.
14. `task.TaskType`: Redundant semicolon on line 9.
15. `task.RecursiveLookupTasks` vs. `RecursiveLookupTask`: The method names are too similar and could be confused.
16. `storage.NodeTableImpl`: Lines 180,68: Comment prefixed with "// XXX"; not even a todo. Such comments should be addressed.
17. `storage.NodeBucketStorage`: `commit()` seems that transactions are supported, but they are not: implementations are empty. This should be removed.
18. `storage.NodeBucketStorageImpl`: Line 39: `!nodeBucketsTable.get(0).isPresent()` can be simplified into `nodeBucketsTable.get(0).isEmpty()` (avoiding negation).

19. `storage.NodeTableImpl`: Lines 53-63: This loop does not make sense. We initialize required into number of indices (constant 256), then turn it into 0 because line 54 is always true, then skip if on line 60, because required is now 0, and exit the loop. Note: the Teku confirmed on Slack that this loop should be deleted.
20. schema: `private` in constructor for enums can be deleted (`IdentitySchema`, `Protocol`, `IdentityScheme`)
21. `IdentityScheme` vs. `IdentitySchema` enums: Single character difference in the class name is easy to confuse. We suggest renaming one or both of the classes.
22. `org.ethereum.beacon.discovery.pipeline.handler.NextTaskHandler`: Line L65: Use `.isEmpty()` instead of `!.isPresent()`.
23. `org.ethereum.beacon.discovery.database.LinkedDataSource`: Line 38: Stray semicolon.
24. `org.ethereum.beacon.discovery.format.SerializerFactory`: Contains code that is commented out; delete it.
25. It seems some methods do not follow the camel case naming adopted in the majority of the methods. Example: `handleMessage_validAggregate`in `tech/pegasys/teku/networking/eth2/gossip/topics/validation/AttestationValidator.java`, `data/serializer/.../api/request/SubscribeToBeaconCommitteeRequest.java`, and `data/serializer/.../api/response/GetForkResponse.java`. We recommend refactoring any method not in camel case so that its name conforms to the convention.
26. Return on `tech/pegasys/teku/networking/eth2/gossip/topics/validation/BlockValidator.java` (L88) is unnecessary; having L91 suffices.
27. `src/main/java/io/libp2p/discovery/mdns/impl/DNSIncoming.java` defines `clone()` but doesn't implement `Cloneable`. Change the definition of `DNSIncoming` so that it implements the `Cloneable` interface.
28. In `src/main/java/com/southernstorm/noise/protocol/Noise.java` (Line 229), the method `newInstance()` is deprecated. To eliminate any associated compilation warning, add `@SuppressWarnings("deprecation")` to `throwBadTagException()`.
29. In `src/main/java/io/libp2p/discovery/mdns/impl/ServiceInfoImpl.java`, `logger` is declared, but not used. No information is logged whatsoever. We suggest adding logging information to aid debugging.
30. In `src/main/java/io/libp2p/discovery/mdns/impl/DNSIncoming.java` (Line 195), variable `source` is declared, but never used; remove it.
31. The cast in `src/main/java/io/libp2p/discovery/mdns/impl/tasks/ServiceResolver.java` (Line 55) is not safe. To eliminate any associated compilation warning, add `@SuppressWarnings("deprecation")` to the `stop()` method.
32. In `src/main/java/com/southernstorm/noise/protocol/Pattern.java` (Line 696), `FLAG_REMOTE_EPHEMERAL` is duplicated; remove the line.
33. In `src/main/java/io/libp2p/discovery/mdns/impl/DNSIncoming.java (Line 468)` the return value of `_messageInputStream.skip(len)` is not checked. However, it should be checked, and if different, some debug information should be logged accordingly.
34. In `src/main/java/io/libp2p/discovery/mdns/impl/tasks/ServiceResolver.java` (Line 48), change `1000` to `1000L`. Reason: when arithmetic is performed on integers, the result will always be an integer. You can assign that result to a long, double, or float with automatic type conversion, but having started as an int or long, the result will likely not be what you expect.
35. In `src/main/java/io/libp2p/discovery/mdns/ServiceInfo.java` (Line 230), the comment is incorrect. It should be “// clone is NOT supported”.
36. In `tech/pegasys/teku/networking/eth2/gossip/topics/validation/SignedAggregateAndProofValidator.java`, Lines 157-160`are redundant and can be removed; if the attestation was already found to be in`receivedValidAggregations` (see Lines 89-92), then it would have been ignored already.
37. Usage of raw HTTP Codes in `data/provider/src/main/java/ValidatorDataProvider.java` Lines 180, 182, 185 should use some package level or class level constant for HTTP codes.
38. Magic numbers should be avoided, and replaced with a constant.
    1. In `services/powchain/src/DepositTransactionSender.java`: Line 130: gas limit seems to be constant, as both `getGasLimit` function seems to default to `2 * 10^6 L`.
    2. In `ssz/src/main/java/backing/view/VectorViewReadImpl.java`: Line 66: magic number `16384`.
39. Class members should be ordered by scope. Some files where this could be improved are: `BeaconStateUtil.java`, `ForkChoice.java`, `PendingPool.java`, `CachingTaskQueue.java`, `StateGenerationTask.java`, `BlstSecretKey.java`, `MikuliSignature.java`, `AttestationDutyScheduler.java`, `ValidatorClientService.java`, `ScheduledDuties.java`, `ValidatorApiHandler.java`, `ForkChoiceStrategy.java`, `ProtoArrayForkChoiceStrategy.java`, `BeaconChainController.java`, `SlotProcessor.java`, and `HashTree.java`.

Test Results

Test Suite Results

Test data was obtained by executing tests in each folder of the `teku` repository and parsing the resulting output file in order to collect the summary for each test folder.

```
Tests in 'teku.acceptance-tests.acceptanceTest': 5 total, 1 ignored, 4 passed (5 m 1 s)
Tests in 'teku.bls': 534 total, 1 ignored, 533 passed (51.27 s)
Tests in 'teku.errorprone-checks': 4 total, 4 passed (19.18 s)
Tests in 'teku.ethereum.core': 140 total, 140 passed (33.05 s)
Tests in 'teku.ethereum.datastructures': 222 total, 222 passed (27.56 s)
Tests in 'teku.ethereum.statetransition': 86 total, 86 passed (31.62 s)
Tests in 'teku.events.test': 21 total, 21 passed (9.30 s)
Tests in 'teku.fuzz': 16 total, 16 passed (57.77 s)
Tests in 'teku.infrastructure.async': 75 total, 75 passed (21.42 s)
Tests in 'teku.infrastructure.unsigned': 156 total, 156 passed (1.10 s)
Tests in 'teku.networking.eth2': 256 total, 256 passed (1 m 21 s)
Tests in 'teku.networking.p2p': 102 total, 102 passed (36.83 s)
Tests in 'teku.pow': 39 total, 39 passed (6.37 s)
Tests in 'teku.protoarray': 15 total, 15 passed (50.90 s)
Tests in 'teku.services.beaconchain': 47 total, 47 passed (22.50 s)
Tests in 'teku.services.remote-validator': 22 total, 22 passed (2.33 s)
Tests in 'teku.ssz': 44 total, 44 passed (8.66 s)
Tests in 'teku.storage': 454 total, 454 passed (2 m 15 s)
Tests in 'teku.sync': 68 total, 68 passed (24.29 s)
Tests in 'teku.teku': 160 total, 160 passed (6 m 57 s)
Tests in 'teku.util': 80 total, 80 passed (14.35 s)
Tests in 'teku.validator.client': 118 total, 118 passed (47.17 s)
Tests in 'teku.validator.coordinator': 59 total, 59 passed (12.21 s)
Tests in 'teku.validator.remote': 62 total, 62 passed (12.47 s)
```

Code Coverage

Code coverage was computed by executing tests in each folder of the `teku` repository with coverage and updating a single table; it may not be accurate.

Package	Class, %	Method, %	Line, %
all classes	70.4% (**710**/1009)	61.3% (**4468**/7286)	62.1% (**16069**/25886)

Package	Class, %	Method, %	Line, %
tech.pegasys.errorpronechecks	0% (**0**/4)	0% (**0**/11)	0% (**0**/28)
tech.pegasys.teku	33.3% (**1**/3)	37.5% (**6**/16)	22.4% (**19**/85)
tech.pegasys.teku.api	0% (**0**/7)	0% (**0**/75)	0% (**0**/207)
tech.pegasys.teku.api.request	100% (**1**/1)	100% (**1**/1)	100% (**4**/4)
tech.pegasys.teku.api.response	50% (**1**/2)	40% (**2**/5)	66.7% (**14**/21)
tech.pegasys.teku.api.response.v1.node	0% (**0**/4)	0% (**0**/4)	0% (**0**/12)
tech.pegasys.teku.api.schema	52.5% (**21**/40)	45% (**59**/131)	42.8% (**252**/589)
tech.pegasys.teku.beaconrestapi	0% (**0**/5)	0% (**0**/42)	0% (**0**/155)
tech.pegasys.teku.beaconrestapi.handlers	0% (**0**/1)	0% (**0**/9)	0% (**0**/14)
tech.pegasys.teku.beaconrestapi.handlers.admin	0% (**0**/1)	0% (**0**/3)	0% (**0**/11)
tech.pegasys.teku.beaconrestapi.handlers.beacon	0% (**0**/8)	0% (**0**/26)	0% (**0**/171)
tech.pegasys.teku.beaconrestapi.handlers.network	0% (**0**/6)	0% (**0**/12)	0% (**0**/30)
tech.pegasys.teku.beaconrestapi.handlers.node	0% (**0**/5)	0% (**0**/10)	0% (**0**/27)
tech.pegasys.teku.beaconrestapi.handlers.v1.node	0% (**0**/2)	0% (**0**/5)	0% (**0**/20)
tech.pegasys.teku.beaconrestapi.handlers.validator	0% (**0**/10)	0% (**0**/39)	0% (**0**/177)
tech.pegasys.teku.beaconrestapi.schema	0% (**0**/2)	0% (**0**/7)	0% (**0**/13)
tech.pegasys.teku.bls	100% (**9**/9)	86.7% (**78**/90)	72.5% (**185**/255)
tech.pegasys.teku.bls.impl	100% (**5**/5)	92.3% (**12**/13)	85.2% (**23**/27)
tech.pegasys.teku.bls.impl.blst	100% (**9**/9)	79.7% (**59**/74)	73.9% (**255**/345)
tech.pegasys.teku.bls.impl.mikuli	100% (**12**/12)	73.2% (**104**/142)	73.9% (**339**/459)
tech.pegasys.teku.bls.impl.mikuli.hash2g2	100% (**8**/8)	89.6% (**69**/77)	97.8% (**835**/854)
tech.pegasys.teku.bls.impl.noop	0% (**0**/4)	0% (**0**/15)	0% (**0**/19)
tech.pegasys.teku.cli	100% (**2**/2)	77.8% (**14**/18)	84.9% (**124**/146)
tech.pegasys.teku.cli.converter	100% (**1**/1)	60% (**3**/5)	66.7% (**8**/12)
tech.pegasys.teku.cli.deposit	81.8% (**18**/22)	57.7% (**45**/78)	53.1% (**191**/360)
tech.pegasys.teku.cli.options	100% (**12**/12)	100% (**80**/80)	100% (**155**/155)
tech.pegasys.teku.cli.subcommand	50% (**4**/8)	21.4% (**6**/28)	14.3% (**18**/126)
tech.pegasys.teku.cli.subcommand.debug	11.1% (**1**/9)	3.8% (**1**/26)	0.6% (**1**/158)
tech.pegasys.teku.cli.util	100% (**4**/4)	100% (**16**/16)	90.9% (**70**/77)
tech.pegasys.teku.compatibility.multiclient	0% (**0**/1)	0% (**0**/6)	0% (**0**/14)
tech.pegasys.teku.compatibility.multiclient.clients	0% (**0**/1)	0% (**0**/7)	0% (**0**/30)
tech.pegasys.teku.core	100% (**17**/17)	84.8% (**167**/197)	79.4% (**695**/875)
tech.pegasys.teku.core.blockvalidator	100% (**7**/7)	100% (**22**/22)	94.8% (**91**/96)
tech.pegasys.teku.core.epoch	100% (**4**/4)	90.7% (**39**/43)	85.4% (**274**/321)
tech.pegasys.teku.core.exceptions	33.3% (**1**/3)	33.3% (**2**/6)	33.3% (**4**/12)
tech.pegasys.teku.core.lookup	100% (**4**/4)	100% (**18**/18)	100% (**37**/37)
tech.pegasys.teku.core.operationsignatureverifiers	100% (**2**/2)	100% (**7**/7)	88.2% (**30**/34)
tech.pegasys.teku.core.operationvalidators	100% (**9**/9)	89.8% (**44**/49)	96% (**119**/124)
tech.pegasys.teku.core.results	100% (**4**/4)	76.2% (**16**/21)	79.5% (**31**/39)
tech.pegasys.teku.core.signatures	71.4% (**5**/7)	68.5% (**37**/54)	68.5% (**98**/143)
tech.pegasys.teku.core.signatures.record	0% (**0**/4)	0% (**0**/25)	0% (**0**/71)
tech.pegasys.teku.core.stategenerator	90.9% (**10**/11)	91.4% (**74**/81)	87.9% (**340**/387)
tech.pegasys.teku.data	100% (**1**/1)	75% (**3**/4)	87.5% (**7**/8)
tech.pegasys.teku.data.recorder	**0% (0/1)**	**0% (0/7)**	**0% (0/18)**
tech.pegasys.teku.datastructures.attestation	100% (**1**/1)	94.4% (**17**/18)	94.6% (**35**/37)

Package	Class, %	Method, %	Line, %
tech.pegasys.teku.datastructures.blocks	100% (**11**/11)	86.1% (**99**/115)	84.7% (**272**/321)
tech.pegasys.teku.datastructures.forkchoice	66.7% (**4**/6)	57.1% (**32**/56)	70.9% (**90**/127)
tech.pegasys.teku.datastructures.hashtree	100% (**3**/3)	96.9% (**31**/32)	95.6% (**87**/91)
tech.pegasys.teku.datastructures.hashtree.traversal	100% (**3**/3)	91.7% (**11**/12)	96.7% (**29**/30)
tech.pegasys.teku.datastructures.networking.libp2p.rpc	87.5% (**7**/8)	44.6% (**33**/74)	49.5% (**94**/190)
tech.pegasys.teku.datastructures.operations	100% (**13**/13)	78.7% (**111**/141)	74.4% (**334**/449)
tech.pegasys.teku.datastructures.state	94.7% (**18**/19)	74.3% (**176**/237)	74.6% (**547**/733)
tech.pegasys.teku.datastructures.util	100% (**18**/18)	85.6% (**237**/277)	86.9% (**1102**/1268)
tech.pegasys.teku.datastructures.validator	33.3% (**1**/3)	25% (**3**/12)	27.3% (**6**/22)
tech.pegasys.teku.ethtests	0% (**0**/1)	0% (**0**/8)	0% (**0**/37)
tech.pegasys.teku.ethtests.finder	0% (**0**/6)	0% (**0**/42)	0% (**0**/82)
tech.pegasys.teku.events	0% (**0**/8)	0% (**0**/50)	0% (**0**/150)
tech.pegasys.teku.fuzz	100% (**1**/1)	100% (**17**/17)	86% (**92**/107)
tech.pegasys.teku.fuzz.input	100% (**7**/7)	68.4% (**39**/57)	64.7% (**90**/139)
tech.pegasys.teku.infrastructure.async	85.7% (**12**/14)	82.2% (**111**/135)	71.9% (**274**/381)
tech.pegasys.teku.infrastructure.unsigned	100% (**1**/1)	100% (**37**/37)	98.6% (**73**/74)
tech.pegasys.teku.logging	64.3% (**9**/14)	30.5% (**32**/105)	19% (**68**/358)
tech.pegasys.teku.metrics	70% (**7**/10)	48.9% (**22**/45)	48% (**61**/127)
tech.pegasys.teku.network.p2p	0% (**0**/2)	0% (**0**/8)	0% (**0**/42)
tech.pegasys.teku.network.p2p.jvmlibp2p	0% (**0**/1)	0% (**0**/5)	0% (**0**/8)
tech.pegasys.teku.network.p2p.peer	100% (**2**/2)	50% (**9**/18)	54.5% (**24**/44)
tech.pegasys.teku.networking.eth2	0% (**0**/8)	0% (**0**/92)	0% (**0**/344)
tech.pegasys.teku.networking.eth2.gossip	33.3% (**2**/6)	20% (**4**/20)	33.8% (**25**/74)
tech.pegasys.teku.networking.eth2.gossip.encoding	100% (**5**/5)	85.7% (**12**/14)	81.8% (**27**/33)
tech.pegasys.teku.networking.eth2.gossip.events	100% (**1**/1)	75% (**3**/4)	81.8% (**9**/11)
tech.pegasys.teku.networking.eth2.gossip.subnets	33.3% (**2**/6)	36.6% (**15**/41)	53.1% (**76**/143)
tech.pegasys.teku.networking.eth2.gossip.topics	68.8% (**11**/16)	68% (**51**/75)	65.1% (**112**/172)
tech.pegasys.teku.networking.eth2.gossip.topics.validation	90% (**9**/10)	75.8% (**50**/66)	71.3% (**258**/362)
tech.pegasys.teku.networking.eth2.mock	0% (**0**/1)	0% (**0**/2)	0% (**0**/2)
tech.pegasys.teku.networking.eth2.peers	60% (**6**/10)	20.9% (**29**/139)	27.6% (**134**/485)
tech.pegasys.teku.networking.eth2.rpc	100% (**2**/2)	78.6% (**11**/14)	95.8% (**69**/72)
tech.pegasys.teku.networking.eth2.rpc.beaconchain	100% (**1**/1)	73.3% (**11**/15)	88.9% (**32**/36)
tech.pegasys.teku.networking.eth2.rpc.beaconchain.methods	66.7% (**8**/12)	32% (**16**/50)	25.3% (**56**/221)
tech.pegasys.teku.networking.eth2.rpc.core	56% (**14**/25)	52.9% (**64**/121)	53.6% (**229**/427)
tech.pegasys.teku.networking.eth2.rpc.core.encodings	100% (**9**/9)	90.3% (**28**/31)	72.4% (**118**/163)
tech.pegasys.teku.networking.eth2.rpc.core.encodings.compression.exceptions	0% (**0**/4)	0% (**0**/5)	0% (**0**/10)
tech.pegasys.teku.networking.eth2.rpc.core.encodings.compression.noop	100% (**3**/3)	70% (**7**/10)	66.7% (**16**/24)
tech.pegasys.teku.networking.eth2.rpc.core.encodings.compression.snappy	100% (**7**/7)	90% (**27**/30)	69.1% (**159**/230)
tech.pegasys.teku.networking.eth2.rpc.core.encodings.ssz	100% (**3**/3)	100% (**14**/14)	86.2% (**25**/29)
tech.pegasys.teku.networking.p2p	100% (**1**/1)	66.7% (**14**/21)	82.9% (**58**/70)
tech.pegasys.teku.networking.p2p.connection	100% (**6**/6)	98% (**50**/51)	97.3% (**177**/182)
tech.pegasys.teku.networking.p2p.discovery	**100% (1/1)**	**75% (6/8)**	**56% (14/25)**
tech.pegasys.teku.networking.p2p.discovery.discv5	0% (**0**/2)	0% (**0**/19)	0% (**0**/55)

Package	Class, %	Method, %	Line, %
tech.pegasys.teku.networking.p2p.discovery.noop	100% (**1**/1)	37.5% (**3**/8)	37.5% (**3**/8)
tech.pegasys.teku.networking.p2p.libp2p	77.8% (**7**/9)	26.7% (**24**/90)	31.1% (**85**/273)
tech.pegasys.teku.networking.p2p.libp2p.gossip	0% (**0**/3)	0% (**0**/15)	0% (**0**/58)
tech.pegasys.teku.networking.p2p.libp2p.rpc	33.3% (**1**/3)	25% (**7**/28)	25.2% (**27**/107)
tech.pegasys.teku.networking.p2p.mock	33.3% (**1**/3)	20.8% (**5**/24)	30% (**9**/30)
tech.pegasys.teku.networking.p2p.network	83.3% (**5**/6)	28.6% (**16**/56)	47% (**54**/115)
tech.pegasys.teku.networking.p2p.peer	80% (**4**/5)	38.5% (**10**/26)	40% (**20**/50)
tech.pegasys.teku.networking.p2p.rpc	50% (**1**/2)	50% (**1**/2)	66.7% (**2**/3)
tech.pegasys.teku.pow	69.2% (**9**/13)	37.2% (**54**/145)	39.7% (**205**/516)
tech.pegasys.teku.pow.api	33.3% (**1**/3)	11.1% (**1**/9)	11.1% (**2**/18)
tech.pegasys.teku.pow.contract	70% (**7**/10)	21.9% (**7**/32)	9.8% (**8**/82)
tech.pegasys.teku.pow.event	100% (**3**/3)	74.1% (**20**/27)	73.2% (**60**/82)
tech.pegasys.teku.pow.exception	0% (**0**/1)	0% (**0**/2)	0% (**0**/2)
tech.pegasys.teku.protoarray	88.9% (**8**/9)	80.4% (**82**/102)	79.5% (**356**/448)
tech.pegasys.teku.provider	100% (**16**/16)	88.6% (**31**/35)	89.4% (**59**/66)
tech.pegasys.teku.service.serviceutils	40% (**2**/5)	30% (**6**/20)	27.5% (**11**/40)
tech.pegasys.teku.services	0% (**0**/1)	0% (**0**/4)	0% (**0**/18)
tech.pegasys.teku.services.beaconchain	25% (**1**/4)	24% (**18**/75)	21% (**85**/405)
tech.pegasys.teku.services.chainstorage	0% (**0**/1)	0% (**0**/5)	0% (**0**/28)
tech.pegasys.teku.services.powchain	0% (**0**/3)	0% (**0**/16)	0% (**0**/76)
tech.pegasys.teku.services.remotevalidator	50% (**3**/6)	57.1% (**20**/35)	62.9% (**78**/124)
tech.pegasys.teku.services.timer	0% (**0**/2)	0% (**0**/6)	0% (**0**/29)
tech.pegasys.teku.ssz.SSZTypes	100% (**15**/15)	83.6% (**102**/122)	82.7% (**248**/300)
tech.pegasys.teku.ssz.backing	70% (**7**/10)	69.2% (**9**/13)	71.9% (**23**/32)
tech.pegasys.teku.ssz.backing.cache	75% (**3**/4)	53.1% (**17**/32)	61.2% (**49**/80)
tech.pegasys.teku.ssz.backing.tree	100% (**9**/9)	87.2% (**41**/47)	83.9% (**156**/186)
tech.pegasys.teku.ssz.backing.type	100% (**13**/13)	86.4% (**51**/59)	88.3% (**106**/120)
tech.pegasys.teku.ssz.backing.view	94.7% (**18**/19)	88.4% (**107**/121)	93% (**321**/345)
tech.pegasys.teku.ssz.sos	100% (**3**/3)	96.2% (**25**/26)	90.2% (**138**/153)
tech.pegasys.teku.statetransition	100% (**3**/3)	77.1% (**27**/35)	73.9% (**105**/142)
tech.pegasys.teku.statetransition.attestation	100% (**8**/8)	85.7% (**42**/49)	91% (**201**/221)
tech.pegasys.teku.statetransition.blockimport	100% (**1**/1)	41.7% (**5**/12)	67.2% (**45**/67)
tech.pegasys.teku.statetransition.events.attestation	50% (**2**/4)	33.3% (**6**/18)	40% (**16**/40)
tech.pegasys.teku.statetransition.events.block	100% (**2**/2)	62.5% (**5**/8)	54.5% (**12**/22)
tech.pegasys.teku.statetransition.forkchoice	66.7% (**2**/3)	60.5% (**23**/38)	66.2% (**86**/130)
tech.pegasys.teku.statetransition.genesis	0% (**0**/1)	0% (**0**/8)	0% (**0**/32)
tech.pegasys.teku.statetransition.util	100% (**3**/3)	86.3% (**44**/51)	89.1% (**147**/165)
tech.pegasys.teku.storage.api	71.4% (**5**/7)	44.8% (**13**/29)	53.7% (**22**/41)
tech.pegasys.teku.storage.api.schema	100% (**1**/1)	100% (**4**/4)	100% (**7**/7)
tech.pegasys.teku.storage.client	87.5% (**7**/8)	44.6% (**62**/139)	50% (**206**/412)
tech.pegasys.teku.storage.events	100% (**4**/4)	97.7% (**42**/43)	99.2% (**119**/120)
tech.pegasys.teku.storage.server	44.4% (**4**/9)	37.1% (**23**/62)	24.3% (**50**/206)
tech.pegasys.teku.storage.server.metadata	100% (**1**/1)	83.3% (**5**/6)	78.9% (**15**/19)
tech.pegasys.teku.storage.server.network	0% (**0**/1)	0% (**0**/5)	0% (**0**/28)
tech.pegasys.teku.storage.server.noop	0% (**0**/1)	0% (**0**/18)	0% (**0**/18)
tech.pegasys.teku.storage.server.rocksdb	100% (**4**/4)	93.1% (**54**/58)	93.4% (**225**/241)
tech.pegasys.teku.storage.server.rocksdb.core	**100% (10/10)**	**84.9% (90/106)**	**80.6% (377/468)**
tech.pegasys.teku.storage.server.rocksdb.dataaccess	100% (**7**/7)	89.8% (**106**/118)	90.4% (**207**/229)

Package	Class, %	Method, %	Line, %
tech.pegasys.teku.storage.server.rocksdb.schema	100% (**6**/6)	100% (**17**/17)	95.8% (**91**/95)
tech.pegasys.teku.storage.server.rocksdb.serialization	100% (**8**/8)	100% (**34**/34)	100% (**103**/103)
tech.pegasys.teku.storage.server.state	75% (**3**/4)	38.5% (**5**/13)	50% (**27**/54)
tech.pegasys.teku.storage.storageSystem	100% (**6**/6)	84.2% (**48**/57)	84.6% (**170**/201)
tech.pegasys.teku.storage.store	92.3% (**12**/13)	90.1% (**137**/152)	90.2% (**568**/630)
tech.pegasys.teku.sync	80% (**16**/20)	52.8% (**84**/159)	54.1% (**350**/647)
tech.pegasys.teku.sync.util	0% (**0**/1)	0% (**0**/6)	0% (**0**/6)
tech.pegasys.teku.test.acceptance	0% (**0**/3)	0% (**0**/15)	0% (**0**/52)
tech.pegasys.teku.test.acceptance.dsl	0% (**0**/9)	0% (**0**/82)	0% (**0**/277)
tech.pegasys.teku.test.acceptance.dsl.tools	0% (**0**/3)	0% (**0**/19)	0% (**0**/54)
tech.pegasys.teku.util	100% (**1**/1)	100% (**3**/3)	100% (**10**/10)
tech.pegasys.teku.util.bytes	100% (**2**/2)	71.4% (**5**/7)	80% (**12**/15)
tech.pegasys.teku.util.cache	66.7% (**2**/3)	47.4% (**9**/19)	61.1% (**22**/36)
tech.pegasys.teku.util.cli	100% (**4**/4)	72.2% (**13**/18)	30.2% (**38**/126)
tech.pegasys.teku.util.collections	100% (**2**/2)	66.7% (**4**/6)	66.7% (**4**/6)
tech.pegasys.teku.util.config	100% (**12**/12)	78.6% (**165**/210)	84.8% (**540**/637)
tech.pegasys.teku.util.crypto	100% (**1**/1)	60% (**3**/5)	60% (**3**/5)
tech.pegasys.teku.util.events	100% (**3**/3)	89.5% (**17**/19)	87.9% (**51**/58)
tech.pegasys.teku.util.file	100% (**1**/1)	50% (**2**/4)	20% (**6**/30)
tech.pegasys.teku.util.hashtree	100% (**3**/3)	71.9% (**23**/32)	61% (**86**/141)
tech.pegasys.teku.util.iostreams	0% (**0**/3)	0% (**0**/32)	0% (**0**/79)
tech.pegasys.teku.util.json	0% (**0**/9)	0% (**0**/17)	0% (**0**/25)
tech.pegasys.teku.util.message	100% (**1**/1)	66.7% (**2**/3)	66.7% (**2**/3)
tech.pegasys.teku.util.resource	100% (**5**/5)	81.8% (**9**/11)	82.4% (**28**/34)
tech.pegasys.teku.util.time	66.7% (**2**/3)	75% (**9**/12)	76.5% (**13**/17)
tech.pegasys.teku.validator.api	75% (**3**/4)	69.2% (**18**/26)	56.8% (**42**/74)
tech.pegasys.teku.validator.client	72.7% (**8**/11)	56.5% (**48**/85)	54.1% (**151**/279)
tech.pegasys.teku.validator.client.duties	77.8% (**7**/9)	67.6% (**50**/74)	76.6% (**160**/209)
tech.pegasys.teku.validator.client.loader	100% (**5**/5)	100% (**22**/22)	89.2% (**99**/111)
tech.pegasys.teku.validator.client.metrics	100% (**3**/3)	95.5% (**21**/22)	95.7% (**67**/70)
tech.pegasys.teku.validator.client.signer	33.3% (**1**/3)	16.7% (**3**/18)	16% (**8**/50)
tech.pegasys.teku.validator.coordinator	62.5% (**5**/8)	60.5% (**46**/76)	62.2% (**214**/344)
tech.pegasys.teku.validator.eventadapter	0% (**0**/1)	0% (**0**/8)	0% (**0**/17)
tech.pegasys.teku.validator.remote	**50% (2/4)**	**62.8% (27/43)**	**57.1% (80/140)**
tech.pegasys.teku.validator.remote.apiclient	100% (**2**/2)	100% (**23**/23)	91.7% (**88**/96)

Changelog

- 2020-09-29 - Initial report (teku, a3f6ed9; signers, 1069ade; jvm-libp2p, 3fd9dd9; discovery, 840e90b)
- 2020-10-21 - Initial update

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.