

Attestation Client Source Code Review



Attestation Client Source Code Review

Version: v240220

Prepared for: Flare

November 2023

Security Assessment

Executive Summary

Summary of Findings

- Findings where caution is advised

- Solved issues & recommendations

Assessment and Scope

Fixes review




Detailed Findings

Disclaimer

Executive Summary

In June 2022, Flare engaged Coinspect to perform a source code review of its Attestation Client. The objective of the project was to evaluate the security of the application, which is a critical off-chain component of its attestation ecosystem.

The **attestation client** is a crucial dependency of Flare's State Connector contract and uses Flare's Multichain Client Library as a component. Because of the interlink between these systems, Coinspect analyzed both the code itself and design issues arising from the interaction between them and the underlying chains. Design and operational risks that are not directly related to the codebase but should be taken into account are reflected in the **Assessment and Scope** section of this report.

 Solved	 Caution Advised	 Resolution Pending
High 3	High 0	High 0
Medium 1	Medium 3	Medium 0
Low 2	Low 0	Low 0
No Risk 0	No Risk 0	No Risk 0
Total 6	Total 3	Total 0

ATC-1 represents the risks associated with the current storage of secrets. **ATC-2** describes a vulnerability present specifically for the XRPL blockchain where unfinalized transactions might be considered for attestation. Both **ATC-3** and **ATC-4** represent

ways in which the network may halt, be it because providers will crash or will not agree on the votes. **ATC-5** is related to inherited risks of using some blockchain transactions which are malleable. **ATC-6** talks about the difficulty of users to make correct decisions on some scenarios. **ATC-7** informs the risks of not having integration tests in a system which heavily depends on the interaction with other external systems. Lastly, **ATC-8** describes a scenario in which providers will consider old blocks for a round.

Summary of Findings

Findings where caution is advised

These issues have been addressed, but their risks have not been fully mitigated. Any future changes to the codebase should be carefully evaluated to avoid exacerbating these issues or increasing their probability.

Findings with a risk of `None` pose no threat, but document an implicit assumption which must be taken into account. Once acknowledged, these are considered solved.

Id	Title	Risk
ATC-5	Transaction malleability issues make it easier to scam users	Medium
ATC-7	Lack of integration tests	Medium
ATC-9	Insufficient or incomplete test cases	Medium

Solved issues & recommendations

These issues have been fully fixed or represent recommendations that could improve the long-term security posture of the project.

Id	Title	Risk
ATC-1	Secrets not stored securely	High
ATC-2	Non final XRPL transactions are attested	High
ATC-3	Attestation client will crash due to unbound growth of a cache	High

ATC-6	Users can lose money resubmitting rejected transactions	Medium
ATC-4	Unexistence of upperBoundProof block may lead vote splitting	Low
ATC-8	Attesters will vote on old requests when restarted	Low

Assessment and Scope

The audit started on June 23 and was conducted on the as-audit branch of the git repository at <https://github.com/flare-foundation/attestation-client/> as of commit 9973c18be55f2352498f479590b935b41ba4ec89 of Jun 23.

The assessment focused on:

- The custom merkle set created by Flare
- The attestation process itself: its voting process and the transactions included in the merkle set root
- Interactions between the client and the blockchain nodes that may cause problems such as denial of service or incorrect information
- Interactions between the client and the Multichain Client Library - Blocks and transactions ingestion by in the indexer
- Verification operations on valid and confirmed transactions

The version reviewed by Coinspect does not seem to be production ready: there are multiple comments marked with **TODO** and some functionally appears to be unimplemented. For example, at the moment, absolutely all XRPL transactions are considered a valid trustline issuance transaction: the verification is not yet implemented and simply returns `Valid` for all passed transactions.

Operational risks arise from the trust placed in providers: an attacker could try to impersonate Flare to try to get providers to update to a malicious version of the program. Coinspect shared ideas with the Flare team on how to reduce operational risks regarding software updates, like setting up a mechanism which involves a cryptographically signed update or even publishing a known hash in a smart contract to announce updates.

Another risk is the compromise of the passwords and specially the secret keys handled by the attestation providers. This risk is specially discussed in **ATC-01**, which talks about the insecure handling of secrets in the project. Many bridges get compromised directly through their validators, which makes ensuring a minimum safety for these is of the utmost importance.

Even without malicious actors, at the moment there are no incentives for providers to behave according to the expected rules. The lack of an incentive mechanism was a well-known fact for Flare at this point in the project, but nevertheless it is worth noting that many of the intended security properties of the system rely upon its

design. Being an attestation provider is an expensive operation and incentives should take the cost of behaving honestly into account.

For the purpose of this audit, the majority of attestation providers were assumed to be trusted: attacks that an important percentage of providers to collude were not considered. In general, a majority of attestation providers being malicious means the system will not hold.

Even a big minority of compromised attestation providers colluding could halt the network, as it is a requirement that 50% of the network reaches consensus on the merkle root. If a big minority does not collaborate, the rest of the network must agree on the merkle root, which may be difficult due to different confirmation times and setups for each different attestation provider.

Risks associated with a halting of the network due to underlying chains node's behaving inconsistently were acknowledged by Flare and they have stated that they will implement a system to temporarily drop support for the affected blockchain. It is worth noting though that as of the audited commit, if nodes from a particular blockchain start to return inconsistent responses or answer intermittently, it is likely the whole attestation system will halt as well.

Coinspect informed Flare of some concerns regarding the user experience of the project. This risk is reflected particularly in **ATC-6**. Flare is aware of this risk and plans to have in place a system of discovery for attestation providers such that users can always find at least one helpful provider. Flare also expects wallet integrations on different chains to provide abstraction for end users of the system.

The quality of the codebase can be improved, especially considering how critical it is for the security of the attestation system. Specially, the usage of callbacks to handle control flow could be improved. As it stands right now, callbacks are set in different places for different objects, which makes creating a mental flow of how each component interacts with the other difficult. Improving this would make finding bugs easier.

Lastly, it is worth noting that this audit heavily prioritized some critical paths of the codebase, as the time assigned was not sufficient to perform a thorough analysis of each aspect of the program. Coinspect recommends to continue auditing the attestation client as well as other components of the system. Flare has stated that they plan to do so and that they do not consider the system as it stands today as production ready.

Fixes review

Flare shared their fixes in the tag `as-audit-0-fix`, plus a pull request showing the diff and a document outlining their reasoning on every change. Overall, the fixes were correct and addressed the vulnerabilities reported. Two issues (**ATC-4** and **ATC-8**) had their severities reduced after clarifications made by the Flare team.

Detailed Findings

ATC-1

Secrets not stored securely

Status

Solved



Resolution

Fixed

Risk

High



Impact
High

Likelihood
High

Location

`/test/chains-config.json`

Description

Secrets were found in the public Github repository. These gave access to internal Flare services. Furthermore, the system supports storing secrets in the filesystem and configuration files, which is not secure.

Coinspect alerted the Flare team immediately upon discovery of the leak and recommended the secrets to be rotated immediately.

Recommendation

Do not let users hardcode secrets in configuration files, server files or environment variables. Support several cloud providers. Recommend attesters to use a hardware device with the keys, so they can't be extracted. Note that not even an actual HSM will protect against compromise of the server and signing arbitrary transactions.

Attesters should run the exposed proof HTTP API independently and separately from the systems handling a private key. This would make it less likely for critical keys to be exposed.

Status

Flare responded quickly once the leak was discovered and fixed it. They have stated that they are in the process of building recommendations for attesters on how to protect their secrets.

ATC-2

Non final XRPL transactions are attested



Description

XRPL finality is not correctly handled, potentially leading to wrong transactions being attested or the wrong merkle root being calculated. The XRPL confirmation height is defined as 1 block, but XRPL transactions are confirmed when their block (or ledger) is validated, not on a fixed amount of confirmations on top.

To ingest blocks from the XRPL blockchain, the Attestation Client uses the `getBlock` method of the XRPL RPC via the Multichain Client Library, which will use the `ledger` JSON RPC call. Then, it uses the confirmation block number in the settings of the Attestation Client to determine if a transaction is final or not.

```
// MCC client should support hash queries
let newPromise = this.client.getBlock(blockHashOrNumber);
if (typeof blockHashOrNumber === "number") {
  let block = await newPromise;
}
```

```
if (!block) return null;
let blockHash = block.blockHash; // TODO
this.blockCache.set(blockHash, new Promise as Promise<any>); // TODO
type
} else {
  this.blockCache.set(blockHashOrNumber, new Promise as Promise<any>);
// TODO type
}
this.checkAndCleanup();
return new Promise as Promise<any>; // TODO type
}
```

But XRPL may not be final even if returned by the ledger method. As the [XRPL documentation states](#), the ledger RPC returns a validated field that indicates when that particular block is finalized. This is not taken into account by the attestation client, which should be responsible for accepting only finalized transactions.

Recommendation

Check the `validated` field of incoming blocks to make sure its transactions are confirmed. Alternatively, use `ledger_closed` instead of `ledger` in the MCC.



Status

The issue was fixed by:

1. Adding a `isValid()` method to the Multichain Client Library, which returns the validated field of blocks in Ripple.
2. Waiting until that method returns true to continue processing in the Attestation Client.

ATC-3

Attestation client will crash due to unbound growth of a cache

Status Solved	Risk High
	
Resolution Fixed	Impact High Likelihood High
Location <code>lib/caching/CachedMccClient.ts:144</code>	

Description

The Attestation Client will eventually crash due to a bug that prevents the blocks and transactions cache from being cleaned up.

The main growth of the cache is due to its usage when receiving a new confirmed block on the DOGE blockchain, where each new transaction in the DOGE block is added to the `CachedMccClient`. This causes the cache to grow fairly quickly. The `cleanup()` function should remove blocks and transactions from the corresponding caches when the `blockCleanupQueue` and `transactionCleanupQueue` exceed specified limits as shown in the following code snippet:

```
private cleanup() {  
  if (this.blockCleanupQueue.size >= this.settings.blockCacheSize +
```

```
this.settings.cleanupChunkSize) {
    while (this.blockCleanupQueue.size > this.settings.blockCacheSize)
    {
        this.blockCache.delete(this.blockCleanupQueue.shift());
    }
}
if (this.transactionCleanupQueue.size >=
this.settings.transactionCacheSize +
this.settings.cleanupChunkSize) {
    while (this.transactionCleanupQueue.size >
this.settings.transactionCacheSize) {

this.transactionCache.delete(this.transactionCleanupQueue.shift());
    }
}
}
```

Recommendation

However, Coinspect noticed that both `blockCleanupQueue` and `transactionCleanupQueue` are not being filled with the items to be deleted.


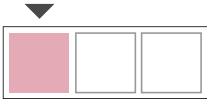
Hence, the actual caches will never be cleaned up.

Status

Fixed by following the recommendation: now the queues are correctly filled, so the system will remove them in time.

ATC-4

Unexistence of upperBoundProof block may lead vote splitting

Status Solved	Risk Low
	
Resolution Fixed	Impact Medium
	Likelihood Low
Location Attestation Client	

Description

The Synchronized query window system designed to synchronize the root hash is susceptible to race conditions that would make the attesters vote incorrectly.

If the indexer does not know the confirmation block provided in the attestation request, the client waits 30 seconds until a retry looking for the block happens. If the commit phase ends before that, i.e. it has passed the 60 seconds mark of the voting window, then the attestation will be ignored by that particular attester.

Users submitting attestation requests for blocks that just appeared on each underlying blockchain will trigger this issue frequently, forcing the attesters to vote different hashes in a pseudo random fashion.

Recommendation

Add a validation for the confirmation block timestamp. Let's say that the confirmation block timestamp in seconds is T and the timestamp of the Flare block where the attestation request event was emitted in seconds is R , then add a validation for

$$T < R - S$$

The current assumption on the code would suggest that $S = 30$ seconds is enough, but a higher value is recommended because multiple blockchains accept blocks in the future.

Status

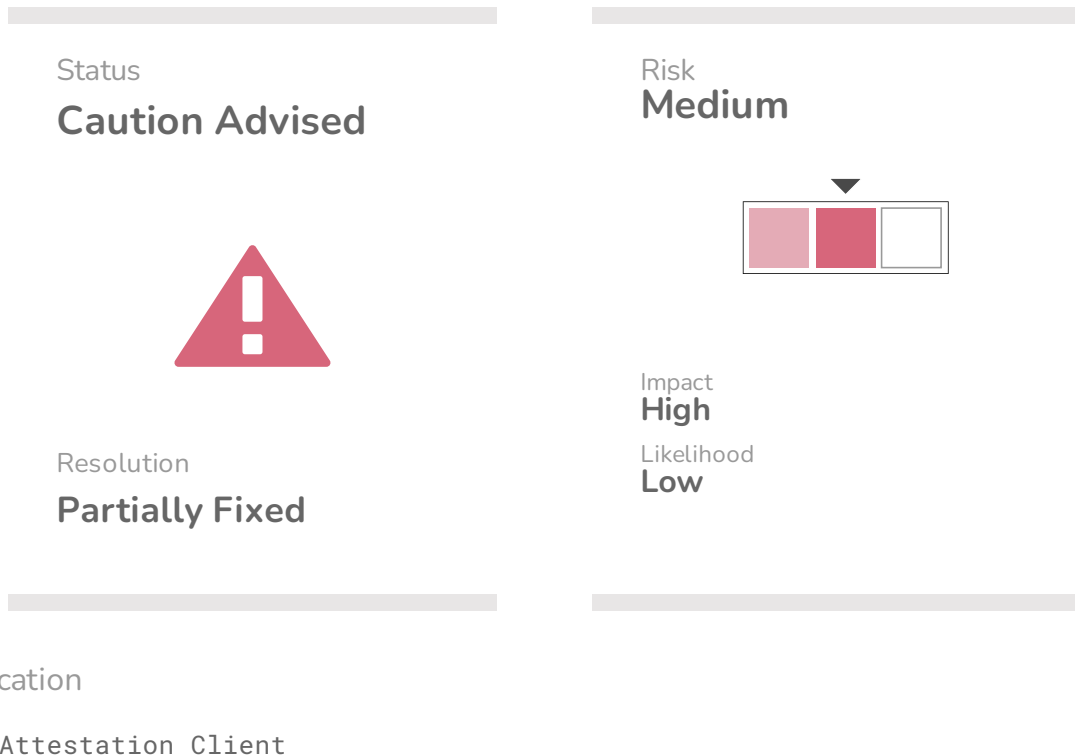
Coinspect originally considered this issue a High risk issue, but has updated the report to be more accurate after the Flare team provided [research](#) and rationale supporting their conclusion that their system is robust enough, at least for Bitcoin blocks. The small likelihood of this issue triggering is considered protection enough by Flare.

It is nevertheless worth keeping in mind that the probability of the issue happening increments with the amount of supported blockchains. Furthermore, it should be noted that the system implicitly depends on the speed on which supported blockchains relay their nodes in their network.

Flare has independently found an issue in this component related to selfish miners mining old forks. The fix for that particular problem is out of scope for this audit.

ATC-5

Transaction malleability issues make it easier to scam users



Description

An attacker looking to scam users can change the hash of a victim's transaction on some blockchains, making it hard for them to attest for their transaction.

Bitcoin non-segwit transactions and XRLP multisig transactions are malleable, which make it possible for attackers to replace transactions.

To exploit this, an attacker has to wait until a user sends them a deposit on a vulnerable underlying chain, such as non-segwit Bitcoin. Once the user performs the deposit, and before the transaction is confirmed, the attacker will replace the transaction for one identical to the one sent by the user, but with a different hash. The replacement mechanism varies by chain, but in Bitcoin could be done by allocating a bigger fee to miners.

When the user then tries to attest for their transaction, the attestation client will, rightfully, reject it, as the transaction is nowhere to be found on the blockchain. The only way for a user to prevent this is for them to be aware of these issues in the underlying chain, monitor the blockchain for new transactions that originate from their account or UTXO's, and submit an attestation with the new hash before the transaction is deemed stale (by the lower boundary in the search for transactions, which is dependent on the round id).

This is not an issue directly related to the Attestation Client, but because the system trusts the security of the underlying chains, it directly affects it. The likelihood is low because most sources of malleability, at least in Bitcoin, have been deemed non standard and will not be relayed by nodes before their inclusion. Nevertheless, there are no consensus rules that force this to happen.

Recommendation

The Attestation Client may not be able to directly solve the problem, but it can mitigate it. Some strategies to do so are:

- Clearly inform users about this issue in underlying chains such as Bitcoin and XRPL.
- Never accept malleable transactions. This makes it clear cut for users of the systems that vulnerable transactions are not accepted and less likely for them to make them send funds to a scammer. In Bitcoin, this would mean attesting only for segwit transactions. In XLRP, this would mean not accepting multisig transactions.

As Flare plans to have integration with wallets which abstract details from users, wallets should only send non-malleable transactions.

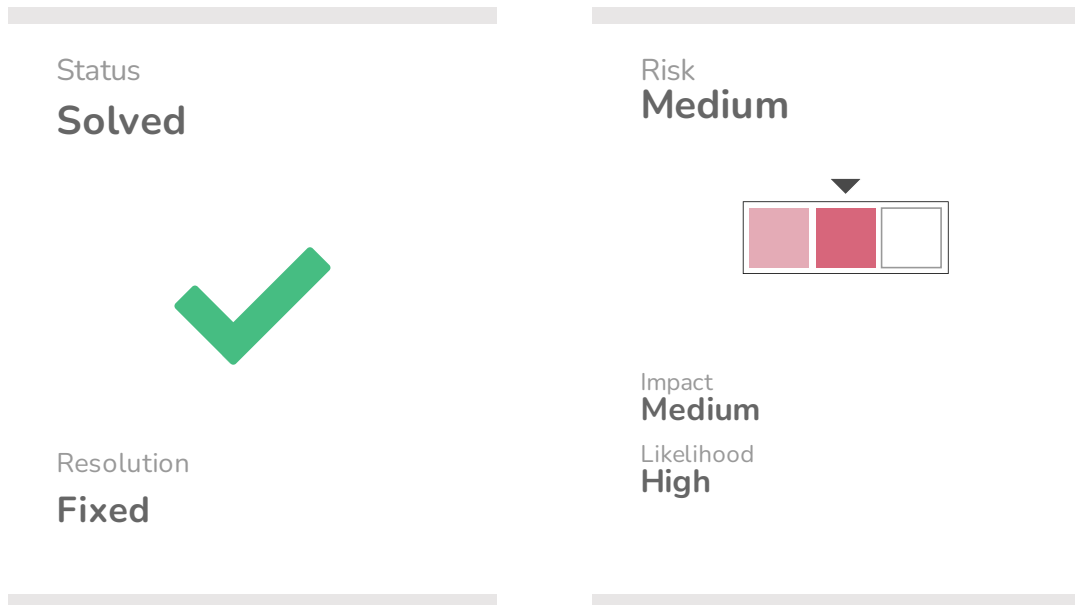
A service could also be provided which makes it easy for users to find their new transaction hash in case they do not use wallets approved by Flare.

Status

Flare will address this only by warning users about the risks involved.

ATC-6

Users can lose money resubmitting rejected transactions



Description

Users have no way to discern between a problem with their underlying transaction, censorship, attestation providers being at full capacity for their attestation or just their request being too late. This leads to them making the wrong choices, such as retrying an attestation that will fail again, making them waste gas and time.

This is due to the attestation client not announcing why a transaction request was not included in the merkle root for that round.

When a transaction is not included in the merkle trie, the user is left with the option of going through several attestation providers before deciding they are not being censored. Even then, the user cannot know if they were too late into the round or surpassed their capacity, or if there is a problem with the underlying transaction (for example, an attacker changing their transaction ID as reported in ATC-5).

Recommendation

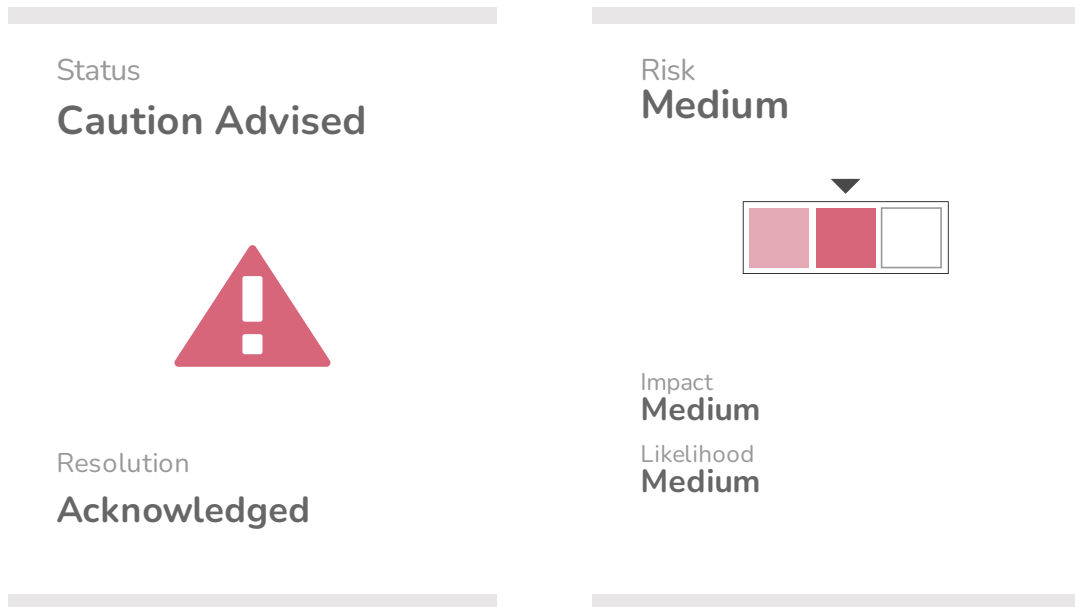
Consider implementing a mechanism to inform users when an attestation request cannot be processed. Attester could provide this information via an API, like they do for proofs.

Status

The recommendation was followed and now there is an endpoint to check the status of attestations.

ATC-7

Lack of integration tests



Description

Lack of integration tests makes it hard to find bugs and vulnerabilities quickly and reliably. As the project depends on the correct interaction and information retrieval from several underlying chains, and the consensus across multiple attestation clients, not performing proper integration testing could conceal bugs related, but not limited to:

- the actual view of the blockchain by nodes, which could be different from other attestation clients,
- the timing of attestation requests submitted by users,
- the coordination of voting windows

Given that the correct operation of the Attestation Client depends on the interaction with multiple blockchain nodes and a minimum expected synchronization, integration tests are heavily recommended.

Recommendation

Consider creating an integration test suite which would ingest data (blocks, transactions, addresses, etc) from production/testnet blockchain portions. To perform tests on real, controlled blockchains would allow developers to test the system under realistic conditions while expecting the same results along several executions. Once integration tests on a limited environment pass, evaluate switching to a testnet environment.


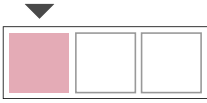
Integration tests should include, but not be limited to, the correct ingestion of blocks and transactions as well as the processing of many different attestation requests through multiple attestation rounds by multiple attestation clients.

Status

The issue was acknowledged and more tests will be added in the future.

ATC-8

Attesters will vote on old requests when restarted

Status Solved	Risk Low
	
Resolution Fixed	Impact Low
	Likelihood Low
Location <code>lib/attester/AttesterClient.ts</code>	

Description

On startup, attesters will choose events from an old block to attest transactions, leading to wrong votes.

When initializing, the attestation client uses the method `getBlockForTime` to choose the block they should start from. This method uses the timestamp from the `roundId` to choose a block, but decrements happen in steps of 10 blocks. This can cause blocks from a previous `roundId` to be collected and their events considered.

```
async getBlockForTime(time: number) {  
  let blockNumber = await  
  this.attesterWeb3.web3Functions.getBlockNumber();  
}
```



```
while (true) {
  let block = await
this.attesterWeb3.web3Functions.getBlock(blockNumber);
  if (block.timestamp < time) {
    this.logger.debug2(`start block number ${blockNumber} time
${secToHHMMSS(block.timestamp)}`);
    return blockNumber;
  }
  blockNumber -= 10;
}
}
```

Recommendation

Use the first block from the current round, instead of returning a block that is potentially 10 behind the current round.

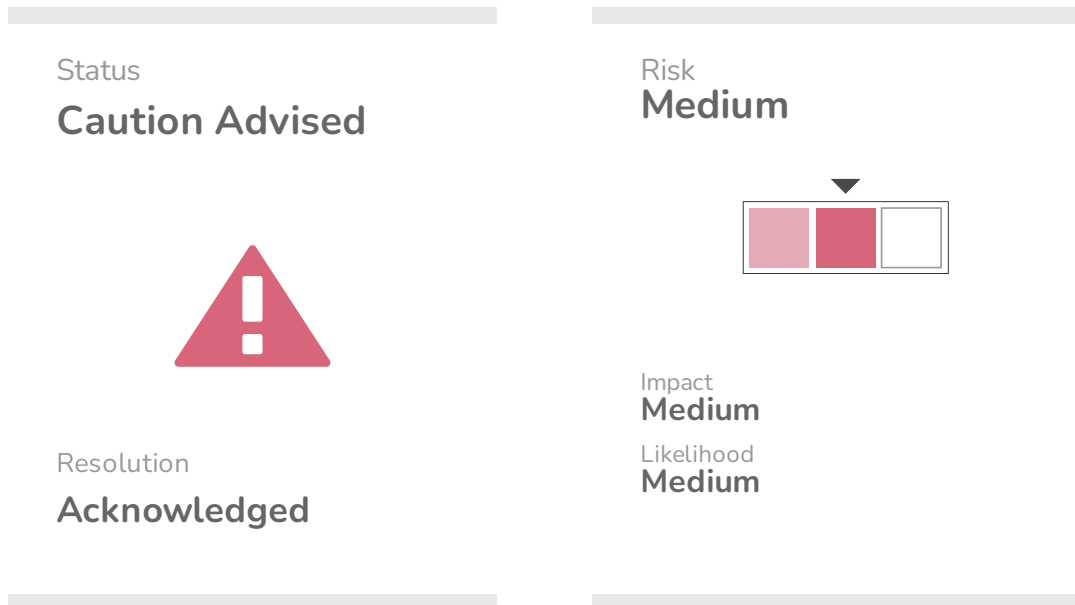
Status

Even though the starting block may be stale, Flare pointed out these would not be considered for voting purposes due to a protection in the atteststate method.

This issue was originally considered a Low risk issue by Coinspect, but was updated to Info to reflect that there is no risk in practice.

ATC-9

Insufficient or incomplete test cases



Description

Insufficient testing added to errors identified in current test cases hinders the detection of unexpected outcomes or behaviors. Given the criticality of the Attestation Client, the current test suite could be improved to cover more functionality and possible execution paths in its test cases.

As an example, in `test/verification/verification.test.ts:41` the number of confirmations required is preset for all the test cases independently of the underlying chain being tested:

```
let NUMBER_OF_CONFIRMATIONS = 6;
```

Furthermore, tests in the above mentioned file only test for *happy paths*, ignoring the possibility of an invalid or malformed Attestation Request for instance.\)

On the other hand, the number of tests for the blocks Indexer might not be enough to cover its entire functionality or might not test unexpected conditions such as

the absence of specific blocks.

Finally, consultants identified a minor error in `test/IndexedQueryManager.test.ts:91`, consisting in a typo. Should have been `randomTransaction2` instead of `randomTransaction1` twice.

Recommendation

Make sure tests are performed using configurations similar to the ones expected for a productive environment (e.g., the correct number of confirmations expected).

Consider increasing the number of test cases to cover unexpected or unhappy paths, as well as performing code coverage tests.

Status

The issue was acknowledged and more tests will be added in the future.

Disclaimer

The information presented in this document is provided “as is” and without warranty. Source code reviews are a “point in time” analysis, and as such, it is possible that something in the code could have changed since the tasks reflected in this report were executed. This report should not be considered a perfect representation of the risks threatening the analyzed system.