Multichain Client Library

Source Code Review

coinspect

flare

Multichain Client Library

Source Code Review

Version: v240220 Prepared for: Flare November 2023

Security Assessment

Executive Summary

In **June 2022**, **Flare** engaged Coinspect to perform a source code review of its **Multichain Client Library**. The objective of the project was to evaluate the overall security posture of the library.

✓ Solved	⚠ Caution Advised	✗ Resolution Pending
High 1	High 0	High 0
Medium 1	Medium 0	Medium 0
Low 1	Low 0	Low 0
No Risk 0	No Risk 0	No Risk 0
Total **3**	Total **0**	Total **0**

Coinspect identified one high-risk issue related to the parsing of Algorand transactions, along with medium and low-risk issues within UTXO chains.

In addition to these specific issues, Coinspect also notes a broader concern regarding the library's abstraction level. Acting merely as a thin wrapper over JSON RPC methods of various blockchains, each with potentially different security assumptions, the library's design makes it susceptible to user errors. To mitigate this, Coinspect recommends a more focused effort to standardize responses across different blockchains whenever possible. Clear documentation of any limitations, particularly when standardization is not feasible, would greatly reduce the likelihood of user mistakes.

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.

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
MCC-1	Attacker can create Algorand transactions with fake details	High
MCC-2	UTXO chains return transactions not yet included in a block	Medium
MCC-3	UTXO chains use wrong data to report status	Low

Assessment and Scope

The audit started on June 20 and was conducted on the mcc-audit branch of the git repository at https://github.com/flare-foundation/multi-chain-client/tree/mcc-audit as of commit `fe0b7cbf493cac6937635aea09a38b9b37825954` of Jun 20.

The assessment considered the codebase a library to be used by unknown clients, although special attention was paid to possible interactions with Flare's Attestation Client.

The audited files have the following hash:

```
da9e1635a0490475f81dff9862773ce14f891a4fb91ca9c186dd272496f89d8b
./src/utils/retry.ts
202cd6e2b07c87969474dc5d31aea8da248620b69b0346dfaf677db8ee214777
./src/utils/typeReflection.ts
3e2606d116b250c81b2ed4c59cfc5c19e8f57e7618bfbf1782ce636aa36fab34
./src/utils/constants.ts
f219acd5c6c3efde6b155e31bbe77a42d502d3e4b1ad8f91c724934fe1dfa9e4
./src/utils/utxoUtils.ts
6145423af75ec26258c2dd98b74f0154d7e4f540aa1de1760b8a97bf555a9315
./src/utils/xrpUtils.ts
65ab01128e5ecd3bd6e471775d1425c85a1f4b701299a5f6835539533a8947c5
./src/utils/trace.ts
e505bd2a55d92dc4569a8cbc2c4c8c85a724c10ef7ebfd5e0aa12bf38ac3cf6f
./src/utils/algoUtils.ts
54be778f511600818226066b6f7699d9b6508e8244bccd162ba38e9fc9904e2c
./src/utils/managed.ts
b4985c395cb7fe6ec9e651467d95e7fb5e1e3248fefa4a62c511c1a10942c151
./src/utils/utils.ts
2ef86b108d29d44000df6a2a8714c289cd2902da5e8d45f4ff780eb48034f0a2
./src/utils/errors.ts
39acd5d7aec72f37dcf13dda376f382e8e2f029b87bdf6baad5a96be8386dcf7
./src/utils/strackTrace.ts
e59d86a9c5da68be5903329dd337517e725f6527ff83a0d3309512a968f1d2a8
./src/types.ts
868086e3a137efebef103641a39ec91de2e11d7fc4227f49953ad84bb5649c74
./src/types/genericMccTypes.ts
c6a6ee79c99979710647e29bd80b877c852fa132fe4d0ae7f0fd1cbf3b6c59d4
./src/types/utxoTypes.ts
c3467a4ea450ee429517ad38fb0c551e30056689a88ae3f6c988886e2cf24b41
./src/types/axiosRateLimitTypes.ts
4aaea9fca1a36b826abaa6a774a38cc441cd7f823542566d4e95b8970b46ba84
./src/types/xrpTypes.ts
3ae3fc7b6c23b1a2bd1abec2382798d72aaa6c0c8d7d4d7118e74d5ca4fe76ad
./src/types/algoTypes.ts
5fc5b9f935609be9500e5da8f141b0971c6e300f3bdd5dccbe8031fa702eaaa9
./src/types/attestationTypes.ts
73a89afdf58be83486a508b7574f7b8de5c520bd3222424024d480fed29ce9d1
./src/types/dogeTypes.ts
```

```
b6b3675bda27b4fc0966a149c18f0c1297e0981cd5e09ec78af420148204bae1
./src/chain-clients/LtcRpcImplementation.ts
591de157a54cba01ce33c4832178e48b8274225d3d1f9d44886164f51a0ce959
./src/chain-clients/UtxoCore.ts
fb404bba7e64a59fc3d7827407383818b037fc5c4c6417f3ba0ea2a2be8bc042
./src/chain-clients/AlgoRpcImplementation.ts
40062a36bfb2d653444076ef1b29671829f38892065f94af4b5d294e9d9035b3
./src/chain-clients/BtcRpcImplementation.ts
546a03acd5221dc112cc30e40d0671560d707dbfcd1e83abc81a5cbc30fb4255
./src/chain-clients/XrpRpcImplementation.ts
62fe3cba95d60cb5e57d177af78356fd0480e97526ca735a1b19fdbd34beb4eb
./src/chain-clients/DogeRpcImplementation.ts
9d122801e897a84f088bd2b6ee367c4c947301f35a808e09641a14c80e25d865
./src/index.ts
f4218e4664cbfb950e7d65b492851bd403ebd23a72dc6c84902e0a94757d3db1
./src/base-objects/transactions/LtcTransaction.ts
a17d2c0ddc85e18ae4f0b441562e8eac601462bb018034e420f5a6f190698346
./src/base-objects/transactions/UtxoTransaction.ts
6a6ea716bf25956f76df9df9413316fe91038f2dafcf81f8ec9f2f738b2083d3
./src/base-objects/transactions/XrpTransaction.ts
4e5e1c6c68d337a016f6f812513bc7a68e1b39ef90e5dcc2c2937e8d40904a04
./src/base-objects/transactions/BtcTransaction.ts
81e160d908734970bbfb60d5801631a423655584b8e9169b8722cc95888d4d6a
./src/base-objects/transactions/DogeTransaction.ts
29b9eab7314c6bf81e49440f5f1dc0df96f4236e5e8d72e66a8230d3e67a5b64
./src/base-objects/transactions/AlgoTransaction.ts
2a05cafcea1667428a5b900edc48f25af2f9641d0c3bdc18e8e95143d379639d
./src/base-objects/BlockBase.ts
1d374b5dbeb263c593e76850b33dc562331b6410f98807d79a6ffe486606e0c1
./src/base-objects/status/XrpStatus.ts
2866ba2abba0b653b3e0089d3ac7acfb1ea2fb2408622257c0e4cc0e7cf6b683
./src/base-objects/status/AlgoStatus.ts
21c7dbb06e9c675a314ccc36b4355bd01d6e8a224fa2e795ca1eecce70534ed4
./src/base-objects/status/UtxoStatus.ts
b7f80981df637fbc9b55f47eb810aba0588c0069adaa5eb80fa84006d3059fa5
./src/base-objects/TransactionBase.ts
c4829e108d884062a407af6ff87e36260d0210de7b809450a4b5577e06d3ed6b
./src/base-objects/StatusBase.ts
2a0be9fc0b4a7acda9a22f245a64077a4ace39f48e17fda28ef44a82b816ff7a
./src/base-objects/blocks/DogeBlock.ts
b05357f4366465880fd6f9aee93b6113f68559b7cdfd69dcae7dbd24af96808d
./src/base-objects/blocks/LtcBlock.ts
e2270e6bd2eaec2692b77525b2f2c37d4c820ebf274bc08c536eceba5277dc39
./src/base-objects/blocks/XrpBlock.ts
66024e2f6618bd9d8d4a70a64500c76161cff4912f9e0c0296aa59283fa6f223
./src/base-objects/blocks/BtcBlock.ts
2f53715434252df7cdca1fdce3e97de014e44c13ae8a20fde78de727e33f8bd9
./src/base-objects/blocks/AlgoBlock.ts
2a024f1854c6d8ebd56d89b329089beb416a97dda902f5df2007d71fdf2cbabb
./src/base-objects/blocks/LiteBlock.ts
84f2e45847fba2cb395e8901fbf2345739675055f01ebf03336e9804293849f4
./src/base-objects/blocks/UtxoBlock.ts
044c47a84fa7f7014f59ffc4069b842814117b98337646459b444962d49798e6
./src/axios-rate-limiter/axios-rate-limit.d.ts
43289e8f2580ab832d0f89b90f18bc57dd2de18d86feff87fac76d3954ab7a5b
./src/global-settings/globalSettings.ts
```

Fixes Review

Flare shared their fixes in the tag mcc-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. The only issue not completely addressed was **MCC-3**, but its impact is considerably low and unlikely to affect operations.

Detailed Findings

MCC-1

Attacker can create Algorand transactions with fake details

Status
Solved

Resolution
Fixed

Risk
High

Impact
High

Likelihood
High

Location

`src/base-objects/transactions/AlgoTransaction.ts`

Description

The library's vulnerability lies in its susceptibility to incorrectly parsing Algorand transactions, which could be exploited by attackers to execute malicious transactions. Additionally, innocent users may unintentionally encounter parsing errors, potentially resulting in unintended consequences. The severity of these issues varies depending on the library's usage, with one possible consequence being the unauthorized transfer of funds to another account without the ability to inform the State Connector.

Coinspect found three scenarios where Algorand's transactions are not interpreted correctly:

- When a transaction is sent by the clawback address, the library will interpret the source address as the `clawback` instead of the actual source, the asnd.
- When a transaction uses AssetCloseTo, the library will interpret the receiver address as only the `arcv` instead of both the `arcv` and the address specified on `AssetCloseTo`. This also leads to incorrect amounts being reported by the library.
- When a transaction uses CloseRemainderTo, a similar scenario occurs, but the transaction will be of type pay. This makes it a valid native transaction according to the library and it may have a valid payment reference. **This may be used by an attacker to move funds undetected by the Attestation Client**.

It's important to note that the operations triggering these issues do not necessarily require deliberate manipulation and may occur during the regular operation of a user's account.

The specific functions contributing to this problem are the getters `sourceAddresses`, `receingAddresses`, and `spentAmounts`. These functions consider only the `rcv`, `arcv`, `amt`, and `aamnt`t fields, potentially overlooking other relevant fields such as asnd.

Coinspect has developed a test case that demonstrates the inaccuracies in reporting when a `clawback` address is utilized. Further details can be found in the **Appendix**.

Recommendation

Parse these special fields correctly.

Status

The Multichain Client Library now takes into account the extra fields. The implementation added some extra getters. It is up to clients to actually use them correctly.

MCC-2

UTXO chains return transactions not yet included in a block

Status

Solved

Resolution

Fixed

Risk

Medium

Impact

Medium

Likelihood

Medium

Location

```
/src/chain-clients/UtxoCore.ts
```

Description

The library's behavior includes returning UTXO-chain transactions that reside solely in the mempool, awaiting inclusion in a block. This could potentially mislead clients into considering transactions that will never be confirmed.

This behavior is unexpected for users of the library, as it lacks documentation and diverges from the approach taken for transactions from other chains. For instance, while Algorand throws an exception in similar scenarios, Ripple only returns transactions already included in a block. To address this issue, users of the library must be aware of the need to query `tx.data.confirmations > 0` to ensure that transactions are included in the mainchain.

The root of the problem lies in the usage of the `getrawtransaction` method without specifying a block hash, a practice that results in the retrieval of transactions from the mempool.

```
    let verbose = true; // by default getting transaction is in verbose
mode
    let unTxId = unPrefix0x(txId);
    let res = await this.client.post("", {
         jsonrpc: "1.0",
         id: "rpc",
         method: "getrawtransaction",
         params: [unTxId, verbose],
    });
```

Users of the library that have implemented custom confirmation requirements will not be affected by this problem.

Recommendation

Make the user explicitly state that they are requesting either a finalized transaction, a transaction included in at least a block or a mempool transaction.

Status

A check has been added so transactions without at least one confirmation are not returned.

MCC-3

UTXO chains use wrong data to report status

Status
Solved

Resolution
Fixed

Risk
Low

Impact
Low

Likelihood
Medium

Location

```
src/base-objects/status/UtxoStatus.ts
```

Description

UTXO chains report wrong network data from the node, leading a client blind to problems with their node or its synced status. As with all components in the library, the impact of this is very dependent on how the client code uses the affected functions UTXO chains use the `networkactive` key from the `getnetworkinfo` RPC endpoint to report both if it a node is synced and if it is healthy:

```
public get isHealthy(): boolean {
   return this.data.networkactive;
}

public get isSynced(): boolean {
   return this.data.networkactive;
}
```

This field has no relationship to neither of health or sync status. `networkactive` will only return whether the p2p network is enabled on the node, a boolean flag that is by default true and can be changed at will by calling `setnetworkactive`.

Recommendation

Use `verificationinprogress` or `initialblockdownload` to know whether a node is synced.

For `isHealthy`, the answer is more nuanced, as it depends on what the definition of healthy is for the library. This should be documented, but a mixture of the response codes and the warnings fields present in `getnetworkinfo` should help.

Status

The `isSynced` endpoint now uses `initialblockdownload`, as mentioned in the recommendation.

The `isHealthy` has not changed.

Appendix

Testcases

MCC-1

```
it("Should correctly interpret asset sender", function() {
        let txToModify = block.data.block.txns[0];
        // when an asnd is present, this will actually be the
        // person to loose tokens, _not_ the snd!
        txToModify.txn.asnd =
algosdk.decodeAddress("QC7XT7QU7X6IHNRJZBR67RBMKCAPH67PCSX4LYH4QKVSQ7DQZ32P
G5HSVQ").publicKey;
        // the sender will actually be the clawback address
        txToModify.txn.snd =
algosdk.decodeAddress("EW64GC6F24M7NDSC5R3ES4YUVE3ZXXNMARJHDCCCLIHZU6TBEOC7
XRSBG4").publicKey;
        // in the example, the clawback is reclaiming tokens back to itself
        txToModify.txn.arcv =
algosdk.decodeAddress("EW64GC6F24M7NDSC5R3ES4YUVE3ZXXNMARJHDCCCLIHZU6TBEOC7
XRSBG4").publicKey;
        block.transactionObjects = [];
        block.data.block.txns = [txToModify];
        block.processTransactions();
        expect(block.transactions.length).to.eq(1);
        let transaction = block.transactions[0];

        expect(transaction.sourceAddresses.length).to.eq(1);
        expect(transaction.receivingAddresses.length).to.eq(1);

expect(transaction.sourceAddresses[0]).to.eq("QC7XT7QU7X6IHNRJZBR67RBMKCAPH
67PCSX4LYH4QKVSQ7DQZ32PG5HSVQ");

expect(transaction.receivingAddresses[0]).to.eq("EW64GC6F24M7NDSC5R3ES4YUVE
3ZXXNMARJHDCCCLIHZU6TBEOC7XRSBG4");
})
```

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.