FYEO

Flare

Repo: https://github.com/mboben/avalanchego

Security Review Update: October 28, 2025

Reviewer: balthasar@gofyeo.com

Security Code Review Flare

New security issues, 1

After the development team implemented the latest updates, FYEO conducted a review of the modifications. The primary goal of this evaluation was to ensure the continued robustness of the program's security features, safeguarding user funds and maintaining the overall integrity of the program.

General Updates:

The update primarily adds dynamic fee and validator-fee configuration data into multiple genesis files, introducing detailed parameters for fee weighting, capacity, refill rates, minimum prices, and hardcoded constants used for excess conversion. A local genesis file's minimum price was increased as well. Across networking code, the upgrade-time reference was switched to a different upgrade marker, and additional safeguards were introduced so a node running a partial sync will not attempt to create certain types of transactions. The peer/gossip subsystem was adjusted so that nodes flagged for partial sync stop participating in some gossip activities; some mempool-related restrictions were removed or reworked and an unused verifier field was taken out from the network structure. It also enables Etna activation and the related EVM upgrades (the Avalanche EVM upgrades) to be brought into the Flare Network.

A small utility was added to aggregate ownership information from two separate sources into a single collection, and several client/context flows were simplified to use a single chain client instead of separate clients. The wallet subsystem was refactored: the wallet type was converted to a concrete pointer-based form, constructors and signing wiring were changed to accept explicit connection parameters and keychains, and new helpers were added to create a chain-only wallet and fetch chain state. Example programs were updated to use the new wallet creation and context patterns and to call issuance methods directly on the consolidated wallet/context rather than going through the previous layered accessors. Overall, the changes add richer fee tuning, tighten behavior under partial sync, and simplify and clarify wallet and ownership-fetching code paths.

INCONSISTENT PROTECTION MAY PERMIT INVALID TX PROPAGATION

Finding ID: FYEO-FLARE-01
Severity: **Low**
Status: **Open**

Description

In `manager.go` the code now returns `ErrImportTxWhilePartialSyncing` if an `ImportTx` is attempted while `PartialSyncPrimaryNetwork` is enabled.

However, the network layer has had earlier guards removed that previously prevented the node from pushing/pulling gossip and from accepting RPC-issued transactions into the mempool when running partial sync. This creates an inconsistent state, allowing propagation of transactions that the node itself cannot validate and should not forward.

Proof of Issue

File name: vms/platformvm/block/executor/manager.go
Line number: ~128

```
+   // If partial sync is enabled, this node isn't guaranteed to have the full
+   // UTXO set from shared memory. To avoid issuing invalid transactions,
+   // issuance of an ImportTx during this state is completely disallowed.
+   if m.txExecutorBackend.Config.PartialSyncPrimaryNetwork {
+       if _, isImportTx := tx.Unsigned.(*txs.ImportTx); isImportTx {
+           return ErrImportTxWhilePartialSyncing
+       }
+   }
```

File name: vms/platformvm/network/network.go
Line number: ~193

```
func (n *Network) PushGossip(ctx context.Context) {
-   // TODO: Even though the node is running partial sync, we should support
-   // issuing transactions from the RPC.
-   if n.partialSyncPrimaryNetwork {
-       return

func (n *Network) IssueTxFromRPC(tx *txs.Tx) error {
-   // If we are partially syncing the Primary Network, we should not be
-   // maintaining the transaction mempool locally.
-   //
-   // TODO: We should still push the transaction to some peers when partial
-   // syncing.
-   if n.partialSyncPrimaryNetwork {
-       return errMempoolDisabledWithPartialSync
-   }
```

Severity and Impact Summary

Partial-sync nodes may accept or forward transactions that they are not capable of fully validating. `ImportTx`s rely on shared memory/UTXO state. This could result in propagation of invalid transactions to peers and a possible attack vector where adversaries flood partial-sync nodes with transactions that will be gossiped but later rejected.

Recommendation

Make the behavior consistent at the network/mempool layer before allowing any txs from RPC or gossip while `PartialSyncPrimaryNetwork` is enabled.

Commit Hash Reference:

For transparency and reference, the security review was conducted on the specific commit hash for the Flare repository. The commit hash for the reviewed versions is as follows:

f8fd751ab551466c79b3a199efdfafd98d9cce1a

Conclusion:

In conclusion, the security aspects of the Flare program remain robust and unaffected by the recent updates. Users can confidently interact with the protocol, assured that their funds are well-protected. The commitment to security exhibited by the development team is commendable, and we appreciate the ongoing efforts to prioritize the safeguarding of user assets.