OpenZeppelin | security

Ava Labs - Validator Manager Contracts Audit

Ava Labs.

October 30, 2024

Table of Contents

Summary

Type	DeFi
Timeline	From 2024-09-23 To 2024-10-11
Languages	Solidity

Total Issues	10 (9 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	2 (2 resolved)
Low Severity Issues	2 (1 resolved)
Notes & Additional Information	6 (6 resolved)

Scope

We audited the ava-labs/teleporter repository at commit 80c4230.

In scope were the following files:

```
contracts/validator-manager
├── interfaces
│   ├── IERC20Mintable.sol
│   ├── IERC20TokenStakingManager.sol
│   ├── INativeTokenStakingManager.sol
│   ├── IPoAValidatorManager.sol
│   ├── IPoSValidatorManager.sol
│   ├── IRewardCalculator.sol
│   └── IValidatorManager.sol
├── ERC20TokenStakingManager.sol
├── ExampleRewardCalculator.sol
├── NativeTokenStakingManager.sol
├── PoAValidatorManager.sol
├── PoSValidatorManager.sol
├── ValidatorManager.sol
└── ValidatorMessages.sol
```

System Overview

The Validator Manager system consists of smart contracts that manage the registration and termination of validators and their delegators, as outlined in ACP-77. Although this audit does not cover the full scope of ACP-77, the proposal significantly reduces the costs and resource requirements for subnet validators. Following ACP-77, subnet validators will operate independently of Primary Network Validators and will no longer need to validate the primary network. ACP-77 defines the mechanisms for creating subnets, transitioning them to an open proof-of-stake consensus model, and establishing communication between these subnets and the P-chain, which serves as the source of truth for subnet validators. The in-scope smart contracts parse and send messages to the P-chain using the Avalanche Warp Messaging (AWM). In addition, they handle reward accrual and disbursement, locking and unlocking of validator and delegator funds, and transitioning of validators and delegators.

When a subnet is created, the creator is registered as the "owner" of the subnet, and the subnet is considered to have a proof-of-authority consensus mechanism. At any point in the future, the owner can transition the subnet to proof-of-stake, allowing other users to stake tokens and begin creating blocks on the subnet. Validators must submit a BLS public key which will be used for attesting block validity on the subnet. Validators earn rewards from their activities. Delegators may stake tokens on some validator and earn rewards, paying a small percentage to the validator as a fee.

The staked token of the subnet may either be an ERC-20 token or that subnet's native token. The Validator Manager contract has the capability of minting either of these tokens to reward validators and delegators. These tokens, being the stake, must be "locked" into the Validator Manager when initializing the registration of a validator or delegator and are only unlocked when the registration is complete.

To finalize their registration, validators must wait for the warp messages to be passed to the P-chain and back. Once the initial registration message reaches the P-chain and is approved by the P-chain validators, the subnet validator becomes responsible for producing blocks. Once anyone broadcasts the P-chain's "response" message back to the subnet, the validator's status is changed to `Active` after which they begin accruing rewards. If the P-chain never responds and the registration expiry period elapses, the validator may cancel their registration. A similar process is used for ending the validator, as well as for starting and ending delegators. However, only registering a validator has an expiry period associated with it.

Furthermore, there is a "churn" mechanism that measures how much staked assets have changed on the subnet. It also prevents changes to the validator set that would cause the churn rate to be exceeded within a single churn period. This prevents hostile takeovers or chaotic, quick changes to the subnets' validator sets.

ACP-77 also discusses a "continuous fee" associated with being a subnet validator. This fee is charged on the P-chain. When first registering as a validator, some amount of AVAX is escrowed for fee payments for at least two weeks. The fee varies but increases with the number of subnet validators. This fee accounts for the continuous load placed on the Avalanche network due to subnets and forces subnets to be sunset if they run out of funds. This is crucial for clearing out inactive subnets and preventing spam on the overall network. Note that this continuous fee is not part of the audit scope.

Trust Assumptions and Privileged Roles

- We assume that the warp messages function as intended and that the off-chain relayers are properly incentivized to deliver these warp messages to the destination chain.
- We assume that P-chain validators will only sign warp messages that are valid and behave as stated for the purpose of this audit.
- We assume that BLS key ownership is verified for all validators by checking the signature in `RegisterSubnetValidatorTx`.
- We assume that the continuous fee mechanism will be set appropriately to remove stale validators while avoiding being set so high that it may be abused to kick out old validators.
- We assume that the owner of the `PoAValidatorManager` contract, who holds the privileged role of managing the validator set by adding, removing, and adjusting validator weights, is acting honestly and in the best interest of the network.

Medium Severity

M-01 Tokens With Low Decimals Are Incompatible

The `valueToWeight` and `weightToValue` functions convert between an amount of tokens (either ERC-20 or native) and a "weight" for the purposes of recording validator stakes. They perform this conversion by multiplying or dividing by `1e12`. However, this conversion results in two complications.

First, amounts that are locked may not be fully credited to the validator or delegator due to a rounding error. Amounts of tokens up to `1e12 - 1` units may be lost and left in the contract as dust. Second, in the case of an ERC-20 staking asset, if the `decimals` are close enough to 12, this could result in low precision when computing the relative weights of validators and delegators. If the decimals are too low, all weights for validators or delegators could be severely skewed or rounded down to `0`.

Consider informing the subnet creators and validators about the aforementioned complications present in the conversion between value and weight. Alternatively, consider implementing a customizable conversion factor, or a conversion factor derived from reading the `decimals` of the `_token` contract.

Update: *Resolved in pull request #621. The Ava Labs team stated:*

> *We made this conversion factor configurable at the contract level. In addition, we added a note to the documentation regarding the possibility of dust amounts being locked in the contract due to integer division.*

M-02 Churn Tracker Can Be Abused to Stall Validator Manager

To prevent large deviations in stake weight in a short amount of time, the Validator Manager contract enforces churn limitations. Only a certain configurable percentage of the total weight is allowed to be added or removed in a configurable period of time. As churn is always additive whenever the weight is being added or removed, a malicious actor could consume the allowed weight changes in a certain churn period by repeatedly adding and removing themselves as a validator or delegator. This could prevent other users from registering as a validator or delegator and from exiting their validator or delegator position.

While the `PoSValidatorManagerStorage` contract already configures a minimum stake duration, this storage variable can still be smaller than the configured churn period. In addition, there is no minimum duration enforced for Delegators.

In order to prevent this type of attack, consider enforcing a minimum staking duration, for both delegators and validators, of at least one single churn period.

***Update:** Resolved in pull request #617 and pull request #625. The Ava Labs team stated:*

> *This issue was addressed across two PRs. The first added the requirement that the minimum staking duration must be longer than the churn period, and is enforced for common-cast delegations where the delegation is ended prior to the staker. However, in order to allow validators to end at any time after their own minimum stake duration has elapsed, the minimum stake duration cannot always be enforced for delegations, which are implicitly ended if the validator they delegated to is ended. The second properly handles an additional edge case where the validator ends prior to the delegator ending. In this case, it still may have been possible for delegated stake to be added, removed, and re-added in a single churn period. PR #617 made it such that delegated stake can never be returned within the same churn period.*

Low Severity

L-01 Unused Parameters in `calculateReward`

Within the `PoSValidatorManager`, whenever `calculateReward` is called, the `initialSupply` and `endSupply` parameters are hard-coded to 0. Furthermore, within the `ExampleRewardCalculator` contract, these parameters are unused. This fact may not be obvious for future developers or integrators. In addition, since the 0 values of these parameters are hard-coded, the current version of `PoSValidatorManager` would require an update to allow for a meaningful use of these parameters. As the code currently exists, these additional parameters add unneeded complexity and gas consumption.

To prevent the aforementioned parameters from being forgotten about, consider removing them from the current iteration of `PoSValidatorManager` and adding an issue to the project issues backlog regarding implementing them in the future. Alternatively, consider configuring these parameters within the initialization of `PoSValidatorManager`. This can be in the form of storing two constant values or storing a target address which can be called to get their values. This will allow `PoSValidatorManager` to remain extensible and not change the interface of the reward calculator. Moreover, it will keep the existence of these parameters obvious for future developers or deployers.

Update: *Resolved in pull request #612. The Ava Labs team stated:*

> *We removed the two parameters from the interface for the current implementation. This interface could be modified if required for future iterations.*

L-02 Griefing Attack From Identical Validator Registration

When registering a Validator, the `validationID` is supposed to be a unique identifier for it. The `validationID` is the hash of all the registration parameters. Once a validator is registered, the `nodeID` associated with it cannot be used again. Since the `validationID` hash does not incorporate `msg.sender`, it is possible for a malicious entity to front-run validator initialization calls using the same parameters and generate the same `validationID`.

While it should be trivially easy for the victim to generate a new `nodeID`, this may not always be the case. There may be many reasons for the victim to be unable to generate a new `nodeID` quickly, such as hardware or software restrictions. Additionally, pre-computing the `validatorID` may be desirable in some cases, and in such cases, the delegators may be tricked into beginning delegation on a validator that is owned by the attacker. It should also be noted that during this time, even if the victim generates a new `nodeID`, the attacker's validator may be earning rewards that otherwise would have gone to the victim.

Consider including the "owner" `address` in the hash which determines the `validationID` to avoid hash collisions and make `validationID`s truly unique. Alternatively, consider warning users about the possibility of this edge case.

Update: *Acknowledged, not resolved. The Ava Labs team stated:*

> *This issue has been called out in the security considerations of the ACP-77 specification here. We do not think this issue is related to validation ID uniqueness, but rather to node ID uniqueness. Even if the "owner" address was added to the `validationID` pre-image, it is still the case that a given node ID can only have a single active validation per L1 at a time, and there is no "proof of possession" of a given node ID required to*

> *register a validator. As mentioned in the issue, generating a new node ID is trivial, and in order to carry out a node ID griefing attack to prevent someone from becoming a validator, an attacker would have to continually front-run any new node IDs they generate, which would require continually spending transaction fees. We have decided to accept this very minimal risk instead of taking on the additional complexity that proving the possession of a node ID would require in both the implementation and the user flows in adding a validator.*

Notes & Additional Information

N-01 Unused Error

In `ValidatorMessages.sol`, the `InvalidSignatureLength` error is unused.

To improve the overall clarity and readability of the codebase, consider either using or removing any unused errors.

Update: *Resolved at commit 3b06b23. The Ava Labs team stated:*

> *This error was renamed to `InvalidBLSPublicKey` and is properly used in the linked commit.*

N-02 Duplicate Imports

Throughout the codebase, multiple instances of duplicate imports were identified:

- Both `import {Validator} from "./interfaces/IValidatorManager.sol";` and `import { Validator, ValidatorStatus, ValidatorRegistrationInput } from "./interfaces/IValidatorManager.sol";` import duplicated alias `Validator` in `PoSValidatorManager.sol`.
- Import `import { IValidatorManager, ValidatorManagerSettings, ValidatorChurnPeriod, ValidatorStatus, Validator, ValidatorChurnPeriod, SubnetConversionData, InitialValidator, ValidatorRegistrationInput } from "./interfaces/`

`IValidatorManager.sol";` imports `ValidatorChurnPeriod` twice in `ValidatorManager.sol`.

Consider removing duplicate imports to improve the overall clarity and readability of the codebase.

Update: *Resolved in pull request #608.*

N-03 Incorrect Documentation

Throughout the codebase, multiple instances of inaccurate or misleading documentation were identified:

- The NatSpec of the `initialize` function in `NativeTokenStakingManager` mentions that it is initializing the ERC-20 token staking manager, whereas it should be the native token staking manager.
- The NatSpec of the `unpackSubnetValidatorRegistrationMessage` function in `ValidatorMessages` says "to do", whereas it should say "due to".

Consider correcting the documentation to align with the code's behavior. This will help improve the clarity and readability of the codebase.

Update: *Resolved in pull request #616.*

N-04 Unaddressed TODO Comments

Having unaddressed TODO comments in the codebase can affect code clarity and maintainability. A TODO comment was identified in line 367 of `ValidatorManager.sol`.

Consider removing all instances of TODO comments and instead tracking them in the issues backlog. Alternatively, consider linking each inline TODO comment to a corresponding issues backlog entry.

Update: *Resolved in pull request #614.*

N-05 Variable Initialized With Its Default Value

Within `ValidatorMessages.sol`, the `i` variable is being initialized with its default value.

To avoid wasting gas, consider not initializing variables with their default values.

Update: *Resolved in [pull request #609](#) and [pull request #622](#).*

N-06 Magic Numbers

Throughout the codebase, multiple instances of unexplained literal values being used were identified. As an example, the `10000` literal number in `ExampleRewardCalculator.sol`.

Consider defining and using `constant` variables instead of using literals to improve the readability of the codebase.

Update: *Resolved in [pull request #610](#). The Ava Labs team stated:*

> *A constant was added for the `BIPS_CONVERSION_FACTOR` of 10,000.*

Conclusion

The Validator Manager system provides a robust framework for managing validators and delegators within an Avalanche subnet, as outlined in ACP-77. By leveraging Avalanche Warp Messaging (AWM) for communication with the P-chain, the system ensures a seamless flow of messages required for validator registration, reward management, and state transitions. In addition, it supports both ERC-20 and native tokens as staking assets, making it adaptable to various subnet configurations.

Overall, the codebase exhibited strong architectural design and implementation, with a well-thought-out approach to the challenges posed by validator management in a multi-subnet environment. The audit process was smooth, with the Ava Labs team demonstrating excellent communication and responsiveness. Their insights and guidance significantly aided the audit, especially in understanding the intricacies of external dependencies and the broader Avalanche system. The Validator Manager system is well-positioned to enable moving validator management from the P-chain to the subnets.