CMTAT v3.1.0 - Audit Agent Report - Comment

This tab summarizes the finding reported by the Nethermind Audit Agent on CMTAT codebase, commit 5148513f...89477867 (before release v3.1.0)

N°, Title and Severity columns come from the agent, while the others are the answers by CMTA maintainers.

N°	Title	Severity (Agent)	Validity [Valid, Invalid, Partial]	Design choice	Comment
1	Partial-freeze not enforced on transfer path allows frozen tokens to be spent (breaks frozen ≤ balance invariant)	High	Invalid	-	Check made by calling the previous base module, function `_checkTransferred` `CMTATBaseCommon._checkTransferred(spender, from, to, value);` See 0_CMTATBaseCommon.sol#L148
2	Unprotected initialize() allows front-running of proxy initialization	High	Invalid	-	Proxy must always be initialized in the same transaction as deployment.
3	Missing spender validation in transfer check function	Medium	Valid	☑	The check must be done in the next base module CMTATBaseCommon does not include ValidationModule.
4	Missing Contract Validation for RuleEngine Address	Medium	Valid	☑	- Hard to check - Contract operators are supposed to know what they do
5	Transfers ignore pause/deactivation when using CMTATBaseCommon (transfers remain possible after deactivateContract)	Low	Valid	☑	Same as #3
6	Transfers to frozen recipients are possible in CMTATBaseCommon.transfer(), violating the freeze invariant for receivers	Low	Valid	☑	Same as #3
7	Reentrancy window between unfreeze and balance update allows freezing based on stale balance, causing frozen > balance after completion	Low	Invalid	-	The call to the snapshot engine is performed after the update hook by using the old values (balance, totalSupply).
8	canTransfer/canTransferFrom can return true even when the ERC-20 transfer would revert (insufficient balance when no tokens are frozen)	Low	Valid	☑	This is intended. `canTransfer` and `canTransferFrom` does not check the balance and are only present for compliance purpose check. We may eventually want to change this in the future. See issues/336
9	SnapshotEngine hook is bypassed in *update (direct call to CMTATBaseCommon.*update)	Low	Valid	☑	Intended. There is no intermediate `_update` override
10	RuleEngine spender is hardcoded to address(0) for minter-initiated transfers (and for mint/burn), weakening compliance checks that depend on spender identity	Low	Invalid	-	There is no spender with burn/mint and regular transfer operations. Only `transferFrom` has a spender.

N°	Title	Severity (Agent)	Validity [Valid, Invalid, Partial]	Design choice	Comment
11	Forced transfers still enforced by standard validation (freeze/allowlist/pause), defeating enforcement intent	Low	Invalid	-	The module calls directly the ERC-20 OpenZeppelin internal function, bypassing `_checkTransferred`.
12	Inconsistent deactivation handling: canTransfer() can return true while ERC-1404 detectTransferRestriction() reports deactivated	Low	Invalid	-	We don't check deactivated() because the contract must be in the pause state to be deactivated. The function will return false if the contract is deactivated in both case.
13	Approve function not protected by pause modifier, allowing allowance changes when contract is paused	Low	Valid	☑	We don't forbid approval if the contract is in the pause state. We may want to change this in the future. See issues/335
14	ERC2771 forwarder is set via constructor in upgradeable deployments, leaving proxy storage uninitialized and breaking _msgSender semantics	Low	Invalid	-	The forwarder is stored in an immutable variable, which means that it is value is stored directly in the contract bytecode rather in the storage.