ABDK CONSULTING

SMART CONTRACT AUDIT

CMTA

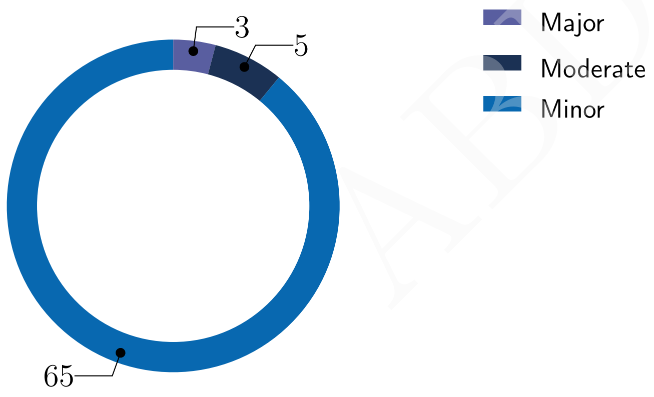
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
10th September 2021

We've been asked to review the CMTAT smart contracts in a github repo. We found 3 major issues and a few less important ones. Major issues were fixed in a subsequent commit.



Findings

ID	Severity	Category	Status
CVF-1	Minor	Procedural	Opened
CVF-2	Minor	Procedural	Opened
CVF-3	Minor	Procedural	Opened
CVF-4	Minor	Bad naming	Opened
CVF-5	Minor	Readability	Opened
CVF-6	Moderate	Suboptimal	Opened
CVF-7	Minor	Suboptimal	Opened
CVF-8	Minor	Suboptimal	Opened
CVF-9	Minor	Suboptimal	Opened
CVF-10	Minor	Suboptimal	Opened
CVF-11	Moderate	Flaw	Opened
CVF-12	Minor	Unclear behavior	Opened
CVF-13	Moderate	Suboptimal	Opened
CVF-14	Minor	Suboptimal	Opened
CVF-15	Minor	Bad naming	Opened
CVF-16	Minor	Suboptimal	Opened
CVF-17	Minor	Flaw	Opened
CVF-18	Minor	Suboptimal	Opened
CVF-19	Minor	Suboptimal	Opened
CVF-20	Minor	Suboptimal	Opened
CVF-21	Minor	Bad datatype	Opened
CVF-22	Minor	Suboptimal	Opened
CVF-23	Minor	Suboptimal	Opened
CVF-24	Minor	Flaw	Opened
CVF-25	Minor	Flaw	Opened
CVF-26	Minor	Flaw	Opened
CVF-27	Minor	Procedural	Opened

ID	Severity	Category	Status
CVF-28	Moderate	Flaw	Opened
CVF-29	Minor	Flaw	Opened
CVF-30	Minor	Flaw	Opened
CVF-31	Minor	Flaw	Opened
CVF-32	Minor	Procedural	Opened
CVF-33	Minor	Flaw	Opened
CVF-34	Minor	Procedural	Opened
CVF-35	Minor	Bad naming	Opened
CVF-36	Minor	Procedural	Opened
CVF-37	Minor	Readability	Opened
CVF-38	Minor	Procedural	Opened
CVF-39	Minor	Procedural	Opened
CVF-40	Minor	Procedural	Opened
CVF-41	Minor	Procedural	Opened
CVF-42	Minor	Suboptimal	Opened
CVF-43	Minor	Procedural	Opened
CVF-44	Minor	Procedural	Opened
CVF-45	Minor	Procedural	Opened
CVF-46	Minor	Procedural	Opened
CVF-47	Minor	Procedural	Opened
CVF-48	Minor	Procedural	Opened
CVF-49	Minor	Procedural	Opened
CVF-50	Minor	Documentation	Opened
CVF-51	Minor	Procedural	Opened
CVF-52	Minor	Procedural	Opened
CVF-53	Minor	Procedural	Opened
CVF-54	Minor	Procedural	Opened
CVF-55	Minor	Procedural	Opened
CVF-56	Moderate	Unclear behaviour	Opened
CVF-57	Minor	Procedural	Opened

ID	Severity	Category	Status
CVF-58	Minor	Procedural	Opened
CVF-59	Major	Flaw	Fixed
CVF-60	Minor	Procedural	Opened
CVF-61	Minor	Procedural	Opened
CVF-62	Minor	Procedural	Opened
CVF-63	Minor	Documentation	Opened
CVF-64	Major	Flaw	Fixed
CVF-65	Major	Suboptimal	Fixed
CVF-66	Minor	Documentation	Opened
CVF-67	Minor	Suboptimal	Opened
CVF-68	Minor	Suboptimal	Opened
CVF-69	Minor	Suboptimal	Opened
CVF-70	Minor	Documentation	Opened
CVF-71	Minor	Bad naming	Opened
CVF-72	Minor	Documentation	Opened
CVF-73	Minor	Suboptimal	Opened

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	September 07, 2021	D. Khovratovich	Initial Draft
0.2	September 08, 2021	D. Khovratovich	Minor revision
1.0	September 09, 2021	D. Khovratovich	Release
1.1	September 10, 2021	D. Khovratovich	Fixes applied

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations. We have reviewed the contract in the CMTAT repository, commit c3afd7:

- interfaces/IRule.sol;
- interfaces/IRuleEngine.sol;
- modules/AuthorizationModule.sol;
- modules/BaseModule.sol;
- modules/BurnModule.sol;
- modules/EnforcementModule.sol;
- modules/MetaTxModule.sol;
- modules/MintModule.sol;
- modules/PauseModule.sol;
- modules/SnapshotModule.sol;
- modules/ValidationModule.sol;
- CMTAT.sol.

The fixes were applied at commit d3dd4a.

2.1 About ABDK

ABDK Consulting, established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function. The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

3 Detailed Results

3.1 CVF-1

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** SnapshotModule.sol

Description Should be "0̂.8.0" unless there is something special about this particular version. Also relevant fot the next files: ValidationModule.sol, CMTAT.sol, PauseModule.sol, AuthorizationModule.sol, MintModule.sol, EnforcementModule.sol, MetaTxModule.sol, BurnModule.sol, BurnModule.sol, BaseModule.sol, IRuleEngine.sol, IRule.sol.

Listing 1:

```
1   solidity ^0.8.2;
```

3.2 CVF-2

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** SnapshotModule.sol

Recommendation The version of the imported library should be provided.

Listing 2:

```
3   "../../openzeppelin-contracts-upgradeable/contracts/utils/ContextUpgradeable.sol";
    "../../openzeppelin-contracts-upgradeable/contracts/proxy/utils/Initializable.sol";
    "../../openzeppelin-contracts-upgradeable/contracts/token/ERC20/ERC20Upgradeable.sol";
    "../../openzeppelin-contracts-upgradeable/contracts/utils/ArraysUpgradeable.sol";
```

3.3 CVF-3

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** SnapshotModule.sol

Recommendation It is common practice to provide a short function description and parameter meaning in the comments preceding the function.

Listing 3:

```
14  contract SnapshotModule is Initializable, ContextUpgradeable, ERC20Upgradeable {
```

3.4 CVF-4

- **Severity** Minor
- **Category** Bad naming
- **Status** Opened
- **Source** SnapshotModule.sol

Description The word “snapshoter” (12K results in Google) sounds odd.
Recommendation “Snapshotter” (100K results in Google) or even “snapshooter” (69K results in Google) sounds much better.

Listing 4:

```
bytes32 public constant SNAPSHOTER_ROLE = keccak256("
  ↪ SNAPSHOTER_ROLE");
```

3.5 CVF-5

- **Severity** Minor
- **Category** Readability
- **Status** Opened
- **Source** SnapshotModule.sol

Description This variable is not initialized.
Recommendation Consider explicitly initializing to 0 for readability.

Listing 5:

```
uint256 private _currentSnapshot;
```

3.6 CVF-6

- **Severity** Moderate
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Description Using an unordered list of scheduled snapshots and removing already created snapshots from it is suboptimal and have several important drawbacks: 1. The ID of a scheduled snapshot is unknown before the snapshot is currently created. This limits possibilities of scheduling snapshots from smart contracts. 2. Schedule times for already created snapshots are not available on-chain.
Recommendation We recommend using an ordered array of scheduled snapshots and don't remove already created snapshots from it, so the snapshot ID is the index in this array. When snapshot is scheduled, its time should be greater that the currency block timestamp and shouldn't be less than time of the latest scheduled snapshot (if any). When snapshot is rescheduled, its new scheduled time shouldn't be less than the time of the previous scheduled snapshot (if any) and shouldn't be greater than the time of the next scheduled snapshot (if any). Only the latest scheduled snapshot could be unscheduled. Such approach would make it possible to use binary search to find the current snapshot index. It also would make the snapshot ID known when snapshot is just scheduled, and would make it possible to know on-chain the scheduled times of already created snapshots. It would also allow scheduling several snapshots at the same time and usint snapshot IDs instead of times to identify the scheduled snapshots.

Listing 6:

```
uint256[] private _scheduledSnapshots;
```

3.7 CVF-7

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Description This function always returns its argument.
Recommendation Consider removing the returned value or returning the index of the just scheduled snapshot.

Listing 7:

```
function _scheduleSnapshot (uint256 time) internal returns (
    ↪ uint256) {
```

3.8 CVF-8

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Recommendation Here two passes over the array are made while just one suffices.

Listing 8:

```
(bool foundNew, ) = _findScheduledSnapshotIndex(newTime);

(bool foundOld, uint256 index) = _findScheduledSnapshotIndex(oldTime);
```

3.9 CVF-9

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Description This function always returns its second argument which is clearly redundant.
Recommendation Consider returning nothing.

Listing 9:

```
return newTime;
```

3.10 CVF-10

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Description This function always returns its argument which is clearly redundant.
Recommendation Consider returning nothing.

Listing 10:

```
function _unscheduleSnapshot (uint256 time) internal returns (uint256) {

  return time;
```

3.11 CVF-11

- **Severity** Moderate
- **Category** Flaw
- **Status** Opened
- **Source** SnapshotModule.sol

Description This method returns also uncleared snapshots from the past.

Listing 11:

```
return _scheduledSnapshots;
```

3.12 CVF-12

- **Severity** Minor
- **Category** Unclear behavior
- **Status** Opened
- **Source** SnapshotModule.sol

Description Is it desired behaviour that the current balance is returned for an invalid snapshot time?

Listing 12:

```
return snapshotted ? value : balanceOf(owner);

return snapshotted ? value : totalSupply();
```

3.13 CVF-13

- **Severity** Moderate
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Description This call reads the entire snapshot array, which is a significant overhead over each transfer.
Recommendation Consider saving snapshots into a more efficient structure, e.g. a list.

Listing 13:

```
_setCurrentSnapshot();
```

3.14 CVF-14

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Description The most often used branch is the last one.
Recommendation Consider refactoring like this: if (from != address (0)) { _updateAccountSnapshot(from); if (to != address (0)) { _updateAccountSnapshot(to); } else { _updateTotalSupplySnapshot(); } } else { _updateAccountSnapshot(to); _updateTotalSupplySnapshot(); }

Listing 14:

```
112 } else {
```

3.15 CVF-15

- **Severity** Minor
- **Category** Bad naming
- **Status** Opened
- **Source** SnapshotModule.sol

Description Semantics of returned values is unclear.
Recommendation Consider adding comments and descriptive names.

Listing 15:

```
119 function _valueAt(uint256 time, Snapshots storage snapshots)
      ↪ private view returns (bool, uint256) {
```

3.16 CVF-16

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Description This function is redundant as it just returns the value of a storage variable.
Recommendation Consider removing this function.

Listing 16:

```
167 function _getCurrentSnapshot() internal view virtual returns (
      ↪ uint256) {
```

3.17 CVF-17

- **Severity** Minor
- **Category** Flaw
- **Status** Opened
- **Source** SnapshotModule.sol

Description These loops do linear searches through the array whose length is unlimited. This makes the contract vulnerable to DoS attacks by scheduling a large number of snapshots.
Recommendation Consider using more efficient approaches.

Listing 17:

```
for (uint256 i=0; i<_scheduledSnapshots.length; i++) {

for (uint256 i=0; i<_scheduledSnapshots.length; i++) {

while (i < _scheduledSnapshots.length) {
```

3.18 CVF-18

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Description The "_scheduledSnapshots.length" value is read on every iteration.
Recommendation Consider caching it in a local variable and decrementing the cashed value when a scheduled snapshot was removed.

Listing 18:

```
while (i < _scheduledSnapshots.length) {
```

3.19 CVF-19

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Description The element to be moved to this position will be once again read from storage at the next iteration.
Recommendation Consider caching it in the memory to avoid extra storage reads and writes.

Listing 19:

```
_removeScheduledItem(i);
```

3.20 CVF-20

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** SnapshotModule.sol

Description This line executes even if index is the last element.

Listing 20:

```
_scheduledSnapshots[index] = _scheduledSnapshots[
  ↪ _scheduledSnapshots.length -1];
```

3.21 CVF-21

- **Severity** Minor
- **Category** Bad datatype
- **Status** Opened
- **Source** ValidationModule.sol

Recommendation The parameter type should be "IRuleEngine".

Listing 21:

```
event RuleEngineSet (address indexed newRuleEngine);
```

3.22 CVF-22

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** ValidationModule.sol

Recommendation This module should use "IRule" instead of "IRuleEngine" as it doesn't need to know that the rule used is actually a composition of several nested rules.

Listing 22:

```
IRuleEngine public ruleEngine;
```

3.23 CVF-23

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** ValidationModule.sol

Description The module doesn't allow changing the rule engine after the deployment.
Recommendation Consider implementing such ability.

Listing 23:

```
IRuleEngine public ruleEngine;
```

3.24 CVF-24

- **Severity** Minor
- **Category** Flaw
- **Status** Opened
- **Source** ValidationModule.sol

Description This will revert in case the rule engine is not set.
Recommendation Consider returning true (allow by default) or false (deny by default) but don't revert in such a case.

Listing 24:

```
return ruleEngine.validateTransfer(from, to, amount);
```

3.25 CVF-25

- **Severity** Minor
- **Category** Flaw
- **Status** Opened
- **Source** ValidationModule.sol

Description This will revert in case the rule engine is not set.
Recommendation Consider returning a special restriction message "No rules set" in such as case when the special restriction code was provided.

Listing 25:

```
return ruleEngine.messageForTransferRestriction(restrictionCode)
    ↪ ;
```

3.26 CVF-26

- **Severity** Minor
- **Category** Flaw
- **Status** Opened
- **Source** ValidationModule.sol

Description This will revert in case the rule engine is not set.
Recommendation Consider returning a special "No rules set" restriction code in such a case.

Listing 26:

```
return ruleEngine.detectTransferRestriction(from, to, amount);
```

3.27 CVF-27

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Description Defining restriction code constants in several placed is error-prone.
Recommendation Consider defining all the codes (and probably all the messages) in one place.

Listing 27:

```
uint8 constant TRANSFER_OK = 0;
```

3.28 CVF-28

- **Severity** Moderate
- **Category** Flaw
- **Status** Opened
- **Source** CMTAT.sol

Recommendation There should be a call to the "__Validation_init_unchained" function somewhere after this call.

Listing 28:

```
__Context_init_unchained();
```

3.29 CVF-29

- **Severity** Minor
- **Category** Flaw
- **Status** Opened
- **Source** CMTAT.sol

Recommendation There should be a call to the "__ERC165_init_unchained" function before this call.

Listing 29:

```
__AccessControl_init_unchained();
```

3.30 CVF-30

- **Severity** Minor
- **Category** Flaw
- **Status** Opened
- **Source** CMTAT.sol

Recommendation This should be done before the "__Base_init_unchained" call as Base module inherits from ERC20 module.

Listing 30:

```
__ERC20_init_unchained(name, symbol);
```

3.31 CVF-31

- **Severity** Minor
- **Category** Flaw
- **Status** Opened
- **Source** CMTAT.sol

Recommendation There should be a call to the "__ERC2771Context_init_unchained" function before this call.

Listing 31:

```
39 __MetaTx_init_unchained(forwarder);
```

3.32 CVF-32

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Recommendation This event should be emitted inside the "_mint" function.

Listing 32:

```
64 emit Mint(to, amount);
```

3.33 CVF-33

- **Severity** Minor
- **Category** Flaw
- **Status** Opened
- **Source** CMTAT.sol

Description This would emit the "Approval" event (which is not desired) but will not emit the "Spend" event (which is actually desired).
Recommendation Consider emitting the "Spend" event and some how preventing emitting the "Approval" event.

Listing 33:

```
82 _approve(account, _msgSender(), currentAllowance - amount);
```

3.34 CVF-34

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Recommendation This event should be emitted inside the "_burn" function.

Listing 34:

```
85 emit Burn(account, amount);
```

3.35 CVF-35

- **Severity** Minor
- **Category** Bad naming
- **Status** Opened
- **Source** CMTAT.sol

Description With multiple inheritance, it is unclear what base contract is referred as “super” here.
Recommendation Consider specifying explicitly like this: return BaseModule.decimals();

Listing 35:

```
139 return super.decimals();

143 return super.transferFrom(sender, recipient, amount);

217 super._beforeTokenTransfer(from, to, amount);

228 return super._msgSender();

232 return super._msgData();
```

3.36 CVF-36

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Recommendation This logic should be moved to the “PauseModule” contract.

Listing 36:

```
154 if (paused()) {
      return TRANSFER_REJECTED_PAUSED;
    }

174 } else if (restrictionCode == TRANSFER_REJECTED_PAUSED) {
      return TEXT_TRANSFER_REJECTED_PAUSED;
```

3.37 CVF-37

- **Severity** Minor
- **Category** Readability
- **Status** Opened
- **Source** CMTAT.sol

Recommendation Should be "} else" for readability.

Listing 37:

```
156 }

159 }

162 }
```

3.38 CVF-38

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Recommendation This logic should be moved to the "EnforcementModule" contact.

Listing 38:

```
157 if (frozen(from)) {
      return TRANSFER_REJECTED_FROZEN;
    }

176 } else if (restrictionCode == TRANSFER_REJECTED_FROZEN) {
      return TEXT_TRANSFER_REJECTED_FROZEN;
```

3.39 CVF-39

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Recommendation This logic should be moved to the "ValidationModule".

Listing 39:

```
160 if (address(ruleEngine) != address(0)) {
      return _detectTransferRestriction(from, to, amount);
    }

178 } else if (address(ruleEngine) != address(0)) {
      return _messageForTransferRestriction(restrictionCode);
```

3.40 CVF-40

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Recommendation These functions should log some event.

Listing 40:

```
function setTokenId (string memory tokenId_) public onlyRole(
   ↪ DEFAULT_ADMIN_ROLE) {

function setTerms (string memory terms_) public onlyRole(
   ↪ DEFAULT_ADMIN_ROLE) {

function setTrustedForwarder(address trustedForwarder_) public
   ↪ onlyRole(DEFAULT_ADMIN_ROLE) {
```

3.41 CVF-41

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Recommendation Due to various misuse cases, it is best practice to use selfdestruct only when multiple short-lived contracts are created.

Listing 41:

```
function kill() public onlyRole(DEFAULT_ADMIN_ROLE) {
```

3.42 CVF-42

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** CMTAT.sol

Recommendation These checks should be the first to save gas.

Listing 42:

```
require(!paused(), "CMTAT: token transfer while paused");
require(!frozen(from), "CMTAT: token transfer while frozen");
```

3.43 CVF-43

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Recommendation This check should be done in the “PauseModule” smart contract.

Listing 43:

```
require(!paused(), "CMTAT: token transfer while paused");
```

3.44 CVF-44

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Recommendation This check should be done in the “EnforcementModule” smart contract.

Listing 44:

```
require(!frozen(from), "CMTAT: token transfer while frozen");
```

3.45 CVF-45

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** CMTAT.sol

Recommendation This code should be moved to the “ValidationModule” smart contract.

Listing 45:

```
if (address(ruleEngine) != address(0)) {
  require(_validateTransfer(from, to, amount), "CMTAT: transfer
     ↪ rejected by validation module");
}
```

3.46 CVF-46

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** PauseModule.sol

Recommendation The version of the imported library should be provided.

Listing 46:

```
"../../openzeppelin-contracts-upgradeable/contracts/security/
  ↪ PausableUpgradeable.sol";
"../../openzeppelin-contracts-upgradeable/contracts/proxy/utils/
  ↪ Initializable.sol";
```

3.47 CVF-47

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** PauseModule.sol

Recommendation This contract should define the "__Pause_init" and "__Pause_init_unchained" functions.

Listing 47:

```
contract PauseModule is Initializable, PausableUpgradeable {
```

3.48 CVF-48

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** PauseModule.sol

Description Defining constants for different transfer rejection reasons in different contracts is error-prone.
Recommendation Consider using a single enum or define these constants in a single file.

Listing 48:

```
uint8 internal constant TRANSFER_REJECTED_PAUSED = 1;
```

3.49 CVF-49

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** AuthorizationModule.sol

Recommendation The version of the imported library should be provided.

Listing 49:

```
"../../openzeppelin-contracts-upgradeable/contracts/access/
    ↪ AccessControlUpgradeable.sol";
```

3.50 CVF-50

- **Severity** Minor
- **Category** Documentation
- **Status** Opened
- **Source** AuthorizationModule.sol

Recommendation It is a good practice to put a comment into an empty block to explain why it is empty.

Listing 50:

```
contract AuthorizationModule is AccessControlUpgradeable {
```

3.51 CVF-51

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** AuthorizationModule.sol

Recommendation This contract should define the "__Authorization_init" and "__Uathorization_init_unchained" functions.

Listing 51:

```
contract AuthorizationModule is AccessControlUpgradeable {
```

3.52 CVF-52

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** MintModule.sol

Description This module doesn't actually implement minting functionality.
Recommendation Consider moving the minting functionality to this module.

Listing 52:

```
contract MintModule {
```

3.53 CVF-53

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** EnforcementModule.sol

Recommendation The version of the imported library should be provided.

Listing 53:

```
"../../openzeppelin-contracts-upgradeable/contracts/utils/
    ↪ ContextUpgradeable.sol";
"../../openzeppelin-contracts-upgradeable/contracts/proxy/utils/
    ↪ Initializable.sol";
"../../openzeppelin-contracts-upgradeable/contracts/token/ERC20/
    ↪ ERC20Upgradeable.sol";
```

3.54 CVF-54

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** EnforcementModule.sol

Recommendation These events should probably also have the “reason” parameter.

Listing 54:

```
event Freeze (address indexed enforcer, address indexed owner);

event Unfreeze (address indexed enforcer, address indexed owner)
    ↪ ;
```

3.55 CVF-55

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** EnforcementModule.sol

Description Defining constants for different transfer rejection reasons in different contracts is error-prone.
Recommendation Consider using a single enum or define these constants in a single file.

Listing 55:

```
31 uint8 internal constant TRANSFER_REJECTED_FROZEN = 2;
```

3.56 CVF-56

- **Severity** Moderate
- **Status** Opened
- **Category** Unclear behaviour
- **Source** EnforcementModule.sol

Description The message is exactly the same as in the "PauseModule" contract. Also, the message is misleading.
Recommendation Consider changing to something like "Account frozen" or "Transfers for the account frozen".

Listing 56:

```
32 string internal constant TEXT_TRANSFER_REJECTED_FROZEN = "All
   ↪ transfers paused";
```

3.57 CVF-57

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** MetaTxModule.sol

Recommendation The version of the imported library should be provided.

Listing 57:

```
3 "../../openzeppelin-contracts-upgradeable/contracts/metatx/
   ↪ ERC2771ContextUpgradeable.sol";
```

3.58 CVF-58

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** MetaTxModule.sol

Description This code relies on how the base contract is implemented.
Recommendation Consider calling the "__ERC2771Context_init" function instead.

Listing 58:

```
13 __Context_init_unchained();
```

3.59 CVF-59

- **Severity** Major
- **Category** Flaw
- **Status** Fixed
- **Source** MetaTxModule.sol

Recommendation An unchained initializer is not supposed to call the base contract initializer.

Listing 59:

```
18 __ERC2771Context_init_unchained(forwarder);
```

3.60 CVF-60

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** BurnModule.sol

Recommendation The version of the imported library should be provided.

Listing 60:

```
3 "../../openzeppelin-contracts-upgradeable/contracts/proxy/utils/
  ↪ Initializable.sol";
```

3.61 CVF-61

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** BurnModule.sol

Description This module doesn't actually implement burning. Also, in inherits from the "Initializable" interface but doesn't have any initializer functions.
Recommendation Consider moving the burning logic into this module and creating an initializer function, probably empty.

Listing 61:

```
5 contract BurnModule is Initializable {
```

3.62 CVF-62

- **Severity** Minor
- **Category** Procedural
- **Status** Opened
- **Source** BaseModule.sol

Recommendation The version of the imported library should be provided.

Listing 62:

```
4 "../../openzeppelin-contracts-upgradeable/contracts/proxy/utils/
    ↪ Initializable.sol";
  "../../openzeppelin-contracts-upgradeable/contracts/token/ERC20/
    ↪ ERC20Upgradeable.sol";
```

3.63 CVF-63

- **Severity** Minor
- **Category** Documentation
- **Status** Opened
- **Source** BaseModule.sol

Description This requirement is not present in the module documentation

Listing 63:

```
63 * - `sender` and `recipient` cannot be the zero address.
```

3.64 CVF-64

- **Severity** Major
- **Category** Flaw
- **Status** Fixed
- **Source** BaseModule.sol

Description The returned value is ignored.
Recommendation Consider emitting the "Spend" event only in case the returned value is true. The current code relies on the knowledge of how the base contract is implemented.

Listing 64:

```
69 super.transferFrom(sender, recipient, amount);
```

3.65 CVF-65

- **Severity** Major
- **Category** Suboptimal
- **Status** Fixed
- **Source** BaseModule.sol

Recommendation It would be better to call "super.approve" here. The current code relies on the knowledge of how the base contract is implemented.

Listing 65:

```
84 _approve(_msgSender(), spender, amount);
```

3.66 CVF-66

- **Severity** Minor
- **Category** Documentation
- **Status** Opened
- **Source** IRuleEngine.sol

Recommendation It is common practice to provide a short function description and parameter meaning in the comments preceding the function.

Listing 66:

```
5 IRuleEngine {
```

3.67 CVF-67

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** IRuleEngine.sol

Description This interface is redundant. It basically defines a composite rule, i.e. a rule composed from other rules, but for those tokens that use such composite rule, it's composite nature doesn't not make any difference and is basically an implementation details.
Recommendation Consider removing this interface and using IRule interface instead in the CMTAT.sol contract. One could always implement a composite rule contract under the "IRule" interface.

Listing 67:

```
IRuleEngine {
```

3.68 CVF-68

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** IRuleEngine.sol

Description Setting all rules at once effectively limits the maximum number of rules, as size of a transaction is limited by block gas limit.
Recommendation Consider implementing an ability to populates the set of rules in several transactions. Also, consider implementing an ability to selectively remove rules.

Listing 68:

```
function setRules(IRule[] calldata rules_) external;
```

3.69 CVF-69

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** IRuleEngine.sol

Description These functions are very similar to the functions defined in the "IRule" interface.
Recommendation Consider inheriting the "IRuleEngine" interface from the "IRule" interface.

Listing 69:

```
function validateTransfer(

function detectTransferRestriction (

function messageForTransferRestriction (uint8 _restrictionCode)
  ↪ external view returns (string memory);
```

3.70 CVF-70

- **Severity** Minor
- **Category** Documentation
- **Status** Opened
- **Source** IRule.sol

Recommendation It is common practice to provide a short function description and parameter meaning in the comments preceding the function.

Listing 70:

```
IRule {
```

3.71 CVF-71

- **Severity** Minor
- **Category** Bad naming
- **Status** Opened
- **Source** IRule.sol

Recommendation The interface name is too generic. The interface is all about transfers, consider reflecting this in the name.

Listing 71:

```
IRule {
```

3.72 CVF-72

- **Severity** Minor
- **Category** Documentation
- **Status** Opened
- **Source** IRule.sol

Description The semantics of these functions is unclear from their signatures.
Recommendation Consider adding documentation comments.

Listing 72:

```
function detectTransferRestriction(

function canReturnTransferRestrictionCode(uint8 _restrictionCode
    ↪ ) external view returns (bool);

function messageForTransferRestriction(uint8 _restrictionCode)
    ↪ external view returns (string memory);
```

3.73 CVF-73

- **Severity** Minor
- **Category** Suboptimal
- **Status** Opened
- **Source** IRule.sol

Description Types narrower than 256 bits don't save gas when used with function arguments or return values, however they limit the range of possible values.
Recommendation Consider using the "uint256" type instead of "uint8".

Listing 73:

```
external view returns (uint8);

function canReturnTransferRestrictionCode(uint8 _restrictionCode
    ↪ ) external view returns (bool);

function messageForTransferRestriction(uint8 _restrictionCode)
    ↪ external view returns (string memory);
```