Report

v. 1.0

Customer

CMTA

Smart Contract Audit

CMTAT and RuleEngine

7th July 2023

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	4.07.23	A. Zveryanskaya	Initial Draft
0.2	7.07.23	A. Zveryanskaya	Minor revision
1.0	7.07.23	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) 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.

Established by Lenz & Staehelin, Swissquote Bank Ltd, and Temenos AG, with the support of EPFL, the CMTA is a not-for-profit, non-governmental association, capable of assuming an independent role as standard-setter for the use of blockchain technology in capital markets.

3 Project scope

We were asked to review:

- CMTAT v.1.0 and 2.2 comparison
- CMTAT patch v.1.0
- RuleEngine Original code

And corresponding fixes:

- CMTAT v. 2.3.0
- CMTAT patch (no modification on the code)
- RuleEngine v. 1.0.2

CMTAT patch files:

/	
	CMTAT.sol
modules/	
	OnlyDelegateCallModule.sol

RuleEngine files:

/		
CodeList.sol	RuleEngine.sol	RuleWhiteList.sol

CMTAT compared files:

/		
CMTAT.sol		
interfaces/		
IDebtGlobal.sol	IERC1404.sol	IERC1404Wrapper.sol
IRule.sol	IRuleEngine.sol	
modules/internal/		
EnforcementModule Internal.sol	ValidationModule Internal.sol	
modules/security/		
AuthorizationModule.sol	OnlyDelegate CallModule.sol	
modules/wrapper/mandatory/		
BaseModule.sol	BurnModule.sol	ERC20BaseModule.sol
EnforcementModule.sol	MintModule.sol	PauseModule.sol
modules/wrapper/optional/DebtModule/		
CreditEvents.sol	MetaTxModule.sol	
ValidationModule.sol		
contracts/modules/		
BaseModule.sol		

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code 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 as well as 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 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 if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure 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.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Minor issues** contain code style, best practices and other recommendations.

5 Our findings

We found 1 critical, 14 major, and a few less important issues. All identified Critical and issues have been fixed.

Issues	Active	Fixed
Severity		
Critical	**0**	**1**
Major	**4**	**10**
Moderate	**1**	**5**
Minor	**13**	**18**

Fixed 34 out of 52 issues

6 Critical Issues

CVF-1. FIXED

- **Category** Flaw
- **Source** ValidationModule.sol

Description Here true is returned even if transfers are paused or the origin address is frozen.

Recommendation Consider returning false in such a case.

Client Comment *The control was made in CMTAT.sol. We have moved this inside the ValidationModule.*

```
119 +return true;
```

7 Major Issues

CVF-2. FIXED

- **Category** Unclear behavior
- **Source** CMTAT.sol

Description When this argument is true, no other argument is used.

Recommendation Consider implementing two versions of this contract: one for standalone deployment, and another for proxy-deployment. These two contracts may share the same base class. Such refactoring could simplify the code.

Client Comment *We have created two versions of the contracts as recommended by the audit.*

```
53 +bool deployedWithProxyIrrevocable_,
```

CVF-3. FIXED

- **Category** Unclear behavior
- **Source** ValidationModule.sol

Description This function returns an empty string on unknown restriction code in case the rule engine is not set.

Recommendation Consider reverting in such a case.

Client Comment *Revert for an external function is not really useful. We have decided to return a defined error message.*

```
93 +function messageForTransferRestriction(
94 +    uint8 restrictionCode
```

CVF-4. FIXED

- **Category** Suboptimal
- **Source** CreditEvents.sol

Description These events are emitted even if corresponding properties didn't change.

Recommendation Consider emitting event only for those properties that actually changed.

Client Comment *We have decided to add only a notice in order to simplify the function since comparing string is not very efficient.*

```
50 +emit FlagDefaultSet(flagDefault_);
51 +emit FlagRedeemedSet(flagRedeemed_);
52 +emit RatingSet(rating_, rating_);
```

CVF-5. FIXED

- **Category** Unclear behavior
- **Source** BurnModule.sol

Recommendation This function should accept the burning reason as an argument.

Client Comment *We have added a reason argument in the function and the event as recommended.*

```
42 +function forceBurn(
```

CVF-6. INFO

- **Category** Unclear behavior
- **Source** ValidationModuleInternal.sol

Description Rule engine is an implementation detail of the module. In the future, the module could take other approach and get rid of rule engine.

Recommendation Consider just allowing all the transfers in case rule engine is not set.

Client Comment *If we do it, we have to define an error message and the code inside the internal module, which we do not want to do.*

```
45 +@dev before making a call to this function, you have to check if a
      ↪ ruleEngine is set.
```

```
56 +@dev before making a call to this function, you have to check if a
      ↪ ruleEngine is set.
```

```
65 +@dev before making a call to this function, you have to check if a
      ↪ ruleEngine is set.
```

CVF-7. INFO

- **Category** Procedural
- **Source** IRuleEngine.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions. This also would make it harder to use this interface in other projects.

Recommendation Consider specifying as "^0.8.0". Also relevant for: IRule.sol, IERC1404Wrapper.sol, IERC1404.sol, IDebtGlobal.sol.

Client Comment *For interface, it is indeed relevant to set the version to ^0.8.0.*

```
3 +pragma solidity ^0.8.17;
```

CVF-8. INFO

- **Category** Suboptimal
- **Source** IRuleEngine.sol

Description Setting all the rules at once could consume lots of gas, especially when only a few rules actually changed.

Recommendation Consider adding function to manipulate set of rules one by one, such as add one rule, remove one rule, change rule's position in the list, etc.

Client Comment *We think that it is to the issuer to decide if he wants these features. We have a better implementation in the RuleEngine project. We prefer to have a minimalist interface here.*

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

CVF-9. INFO

- **Category** Suboptimal
- **Source** IRule.sol

Description This function seems cumbersome and inefficient.

Recommendation Consider replacing it with a functions that returns are bitmask of all valid restriction codes. Such bitmask would fit into a single 256-bit word.

Client Comment *Bitmask is not really readable for non-technical people. We track this as a possible improvement in the future: issues/161*

```
11  +function canReturnTransferRestrictionCode(
```

CVF-10. FIXED

- **Category** Suboptimal
- **Source** RuleEngine.sol

Description Linear search is inefficient.

Recommendation Consider accepting an additional argument with the rule index hint.

Client Comment *Add an additional argument with the rule index hint and create a public function getRuleIndex.*

```
71  71  for (uint256 i = 0; i < _rules.length; ) {
```

CVF-11. FIXED

- **Category** Procedural
- **Source** RuleEngine.sol

Description Implementing an ability to destroy a contract is a bad practice, as it cause more risks than benefits.

Recommendation Consider removing this function.

Client Comment *The function has been removed as recommended. The ability to destroy the contract was not really a requirement.*

```
146 146 function kill() public onlyRole(DEFAULT_ADMIN_ROLE) {
```

CVF-12. FIXED

- **Category** Suboptimal
- **Source** RuleWhiteList.sol

Description Here the same storage slot is updated multiple times in a loop.

Recommendation Consider updating a local variable instead and writing into the storage once after the loop.

Client Comment *Use a local variable as recommended.*

```
40 40 ++numAddressesWhitelisted;
```

```
58 58 --numAddressesWhitelisted;
```

CVF-13. FIXED

- **Category** Suboptimal
- **Source** RuleWhiteList.sol

Description This condition is always true due to the ”require” statement above.

Recommendation Consider removing the conditional statement.

Client Comment *The conditional statement has been removed as recommended.*

```
77 77 if (!whitelist[_newWhitelistAddress]) {
```

CVF-14. FIXED

- **Category** Unclear behavior
- **Source** RuleWhiteList.sol

Description This condition is always true due to the ”require” statement above.

Recommendation Consider removing the conditional statement.

Client Comment *The conditional statement has been removed as recommended.*

```
95  95  if (whitelist[_removeWhitelistAddress]) {
```

CVF-15. FIXED

- **Category** Procedural
- **Source** RuleWhiteList.sol

Description Implementing an ability to destroy a contract is a bad practice, as it cause more risks than benefits.

Recommendation Consider removing this function.

Client Comment *The function has been removed as recommended. The ability to destroy the contract was not really a requirement.*

```
175 175 function kill() public onlyRole(DEFAULT_ADMIN_ROLE) {
```

8 Moderate Issues

CVF-16. FIXED

- **Category** Suboptimal
- **Source** AuthorizationModule.sol

Description Granting all roles to admin looks cumbersome, doesn't scale and doesn't make much sense.

Recommendation A proper way to make admin a superuser with all the roles is to override the "hasRole" function to return true regardless of the role in case the account is an admin.

Client Comment *We have overridden the function hasRole() as recommended.*

```
54 +// EnforcementModule
55 +_grantRole(ENFORCER_ROLE, admin);
56 +// MintModule
57 +_grantRole(MINTER_ROLE, admin);
58 +// BurnModule
59 +_grantRole(BURNER_ROLE, admin);
60 +// PauseModule
61 +_grantRole(PAUSER_ROLE, admin);
62 +// SnapshotModule
63 +_grantRole(SNAPSHOOTER_ROLE, admin);
64 +// DebtModule
65 +_grantRole(DEBT_ROLE, admin);
66 +// CreditEvents
67 +_grantRole(DEBT_CREDIT_EVENT_ROLE, admin);
```

CVF-17. FIXED

- **Category** Suboptimal
- **Source** AuthorizationModule.sol

Description This approach is cumbersome and doesn't scale.

Recommendation A usual and safe way for transferring adminship is: 1. The old admin grant admin role to the new admin 2. The new admin revokes admin role from the old admin.

Client Comment *With the fix of the CVF-10, we can use now the function grantRole() & renounceRole() from the OpenZeppelinLlibrary*

```
72 +The newAdmin will have the same roles as the current admin.
```

```
74 +By transfering his rights, the former admin loses them all.
```

CVF-18. INFO

- **Category** Suboptimal
- **Source** CMTAT.sol

Description Even with "onlyDelegateCall" modifier, the ability to destroy a smart contract is still a bad practice. While ether is rescued from the contract's balance, other assets are not. Also, any assets sent to the contract's address after destruction will be lost.

Recommendation Consider removing this function.

Client Comment *At the moment, it is a legal requirement. We keep it to be compliant with the specifications for the CMTAT 1.0. But we know that the possibility to kill a contract is deprecated.*

```
213 +function kill() public onlyRole(DEFAULT_ADMIN_ROLE)
       ↪ onlyDelegateCall  {
```

CVF-19. FIXED

- **Category** Flaw
- **Source** RuleEngine.sol

Description This function doesn't check for duplicate rules.

Recommendation Consider adding such a check. An efficient way to do this would be to require the "rules_" array to be sorted. This would also simplify the zero rule test.

Client Comment *We use a hashmap to check for duplicate rules and the transaction is reverted in case of such one.*

```
26  26 function setRules(
```

CVF-20. FIXED

- **Category** Unclear behavior
- **Source** RuleEngine.sol

Description This function allows adding a duplicate rule.

Recommendation Consider forbidding duplicate rules.

Client Comment *We use a hashmap to check for duplicate rules and the transaction is reverted in case of such one.*

```
55  55 function addRule(IRule rule_) public onlyRole(RULE_ENGINE_ROLE) {
```

CVF-21. FIXED

- **Category** Unclear behavior
- **Source** RuleEngine.sol

Description This function removes only the first occurrence of a rule.

Recommendation Consider removing all occurrences.

Client Comment *No longer needed since we forbid duplicate rules (CVF-7, CVF-8).*

```
70  70  function removeRule(IRule rule_) public onlyRole(RULE_ENGINE_ROLE) {
```

9 Minor Issues

CVF-22. INFO

- **Category** Procedural
- **Source** ValidationModule.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.

Recommendation Consider specifying as "^0.8.0". Also relevant for: CMTAT.sol, MetaTxModule.sol, DebtBaseModule.sol, CreditEvents.sol, ERC20BaseModule.sol, PauseModule.sol, MintModule.sol, EnforcementModule.sol, BurnModule.sol, BaseModule.sol, OnlyDelegateCallModule.sol, AuthorizationModule.sol, ValidationModuleInternal.sol, EnforcementModuleInternal.sol.

Client Comment *We do not want to offer the possibility to deploy the contract with a version interior to the version 0.8.17, which is a version used for the tests and the development.*

```
3 +pragma solidity ^0.8.17;
```

CVF-23. FIXED

- **Category** Unclear behavior
- **Source** ValidationModule.sol

Description This event if emitted even if nothing actually changed.

Client Comment *We have added a check (require) before emitting the event to compare the old and new value of the ruleEngine.*

```
63 +emit RuleEngineSet(ruleEngine_);
```

CVF-24. FIXED

- **Category** Readability
- **Source** ValidationModule.sol

Recommendation Should be "else return".

Client Comment *We have moved the return statement inside the else branch as recommended.*

```
85 +return uint8(REJECTED_CODE.TRANSFER_OK);
```

CVF-25. INFO

- **Category** Suboptimal
- **Source** MetaTxModule.sol

Description These overrides looks redundant, as they don't change the behavior of the functions.

Recommendation Consider removing them.

Client Comment *If we remove them, in CMTAT we need to override the functions _msgSender() & _msgData() from the module ERC2771ContextUpgradeable instead of MetaTxModule. We prefer to avoid a situation where a wrapper module is "bypassed".*

```
23 +function _msgSender()
```

```
33 +function _msgData()
```

CVF-26. FIXED

- **Category** Bad naming
- **Source** DebtBaseModule.sol

Recommendation Events are usually named via nouns, such as "InterestRate", "ParValue", etc.

Client Comment *We have renamed the different events as recommended.*

```
19 +event InterestRateSet(uint256 indexed newInterestRate);
20 +event ParValueSet(uint256 indexed newParValue);
21 +event GuarantorSet(string indexed newGuarantorIndexed, string
      ↪ newGuarantor);
22 +event BondHolderSet(
```

```
26 +event MaturityDateSet(
```

```
30 +event InterestScheduleFormatSet(
```

```
34 +event InterestPaymentDateSet(
```

```
38 +event DayCountConventionSet(
```

```
42 +event BusinessDayConventionSet(
```

```
46 +event PublicHolidaysCalendarSet(
```

```
50 +event IssuanceDateSet(
```

```
54 +event CouponFrequencySet(
```

CVF-27. FIXED

- **Category** Suboptimal
- **Source** DebtBaseModule.sol

Description Indexing numerical event parameters doesn't make much sense, as indexes could only be used for exact values searches, but not for range searches.

Recommendation Consider not indexing numeric parameters.

Client Comment *We have removed the indexation of numeric parameters as recommended.*

```
19 +event InterestRateSet(uint256 indexed newInterestRate);
20 +event ParValueSet(uint256 indexed newParValue);
```

CVF-28. INFO

- **Category** Unclear behavior
- **Source** DebtBaseModule.sol

Description These events are logged even if nothing actually changed.

Client Comment *Compare string is not very efficient in Solidity. We decide to only perform a check (require) for other types (e.g. uint256) and add a notice for string event.*

```
134 +emit InterestRateSet(interestRate_);
```

```
139 +emit ParValueSet(parValue_);
```

```
144 +emit GuarantorSet(guarantor_, guarantor_);
```

```
151 +emit BondHolderSet(bondHolder_, bondHolder_);
```

```
158 +emit MaturityDateSet(maturityDate_, maturityDate_);
```

```
165 +emit InterestScheduleFormatSet(
166 +    interestScheduleFormat_,
167 +    interestScheduleFormat_
168 +);
```

```
175 +emit InterestPaymentDateSet(interestPaymentDate_,
    ↪ interestPaymentDate_);
```

```
182 +emit DayCountConventionSet(dayCountConvention_, dayCountConvention_
    ↪ );
```

```
189 +emit BusinessDayConventionSet(
190 +    businessDayConvention_,
191 +    businessDayConvention_
192 +);
```

```
199 +emit PublicHolidaysCalendarSet(
200 +    publicHolidayCalendar_,
201 +    publicHolidayCalendar_
202 +);
```

```
209 +emit IssuanceDateSet(issuanceDate_, issuanceDate_);
```

```
216 +emit CouponFrequencySet(couponFrequency_, couponFrequency_);
```

CVF-29. FIXED

- **Category** Bad naming
- **Source** CreditEvents.sol

Recommendation Events are usually named via nouns, such as ”DefaultFlag”, ”RedeemedFlag”, or ”Rating”.

Client Comment *We have renamed the different events as recommended.*

```
19 +event FlagDefaultSet(bool indexed newFlagDefault);
20 +event FlagRedeemedSet(bool indexed newFlagRedeemed);
21 +event RatingSet(string indexed newRatingIndexed, string newRating);
```

CVF-30. FIXED

- **Category** Procedural
- **Source** EnforcementModule.sol

Description It is unclear from this message what address is frozen: origin or destination.

Recommendation Consider having two different messages for these two cases.

Client Comment *We have defined two different messages as recommended by the audit.*

```
20 +"The address is frozen";
```

CVF-31. FIXED

- **Category** Suboptimal
- **Source** BaseModule.sol

Recommendation The ”../modules/” part in the path is redundant.

Client Comment *Remove the redundant part as recommended.*

```
8 +import "../../../modules/security/OnlyDelegateCallModule.sol";
```

CVF-32. FIXED

- **Category** Bad naming
- **Source** BaseModule.sol

Recommendation Events are usually named via flags, such as ”Term”, ”TokenId”, ”Information”, etc.

Client Comment *Rename the different events as recommended.*

```
13 +event TermSet(string indexed newTermIndexed, string newTerm);
14 +event TokenIdSet(string indexed newTokenIdIndexed, string
      ↪ newTokenId);
15 +event InformationSet(
```

```
19 +event FlagSet(uint256 indexed newFlag);
```

CVF-33. FIXED

- **Category** Unclear behavior
- **Source** BaseModule.sol

Description These events are emitted even if nothing actually changed.

Client Comment *Compare string is not very efficient in Solidity. We decide to only perform a check (require) for other types (e.g. uint256) and add a notice for string event.*

```
73 +emit TokenIdSet(tokenId_, tokenId_);
```

```
80 +emit TermSet(terms_, terms_);
```

```
87 +emit InformationSet(information_, information_);
```

```
92 +emit FlagSet(flag_);
```

CVF-34. INFO

- **Category** Suboptimal
- **Source** BaseModule.sol

Description Using selfdestruct nowadays is considered as a bad practice. While it could save ether, it doesn't safe other assets at the contract's balance.

Recommendation Consider removing thins function.

Client Comment *At the moment, it is a legal requirement. We can not remove it.*

```
96 +@notice destroys the contract and send the remaining ethers in the
      ↪ contract to the sender
97 +Warning: the operation is irreversible, be careful
```

CVF-35. FIXED

- **Category** Bad naming
- **Source** ValidationModuleInternal.sol

Recommendation Events are usually named via nouns, such as "RuleEngine".

Client Comment *We have renamed the different events as recommended.*

```
21 +event RuleEngineSet(IRuleEngine indexed newRuleEngine);
```

CVF-36. INFO

- **Category** Suboptimal
- **Source** EnforcementModuleInternal.sol

Description Indexing reasons does make sense if there is a well-known list of possible reasons.

Recommendation Consider defining a enum for possible freeze reasons and indexing a value of this enum. Also there could be an unindexed string field with additional details.

Client Comment *Since the project will be used by different organization, it is possible that each organization will have its own list.*

```
24 +string indexed reasonIndexed,
```

```
34 +string indexed reasonIndexed,
```

CVF-37. FIXED

- **Category** Bad naming
- **Source** IRuleEngine.sol

Description The name sounds odd.

Recommendation Consider renaming to ”rulesCount”.

Client Comment *The function was renamed as recommended.*

```
17 +function ruleLength() external view returns (uint256);
```

CVF-38. INFO

- **Category** Procedural
- **Source** IERC1404.sol

Recommendation This interface should extend the ”IERC20” interface.

Client Comment *This change requires too many modifications to do in the architecture due to linearization error.*

```
5 +interface IERC1404 {
```

CVF-39. FIXED

- **Category** Suboptimal
- **Source** IERC1404.sol

Recommendation Consider defining named constants for valid restriction codes, either inside this interface or somewhere else.

Client Comment *We have defined a list of valid restriction code in ERC1404Wrapper.*

```
14 +) external view returns (uint8);
```

CVF-40. INFO

- **Category** Procedural
- **Source** IDebtGlobal.sol

Description This interface consists only of structs.

Recommendation Consider moving the structs definitions onto the top level and removing this interface.

Client Comment *We keep the current structure. With an interface, it is easier for another organization to create its own version of the DebtModule.*

```
5  +interface IDebtGlobal {
```

CVF-41. INFO

- **Category** Documentation
- **Source** IDebtGlobal.sol

Description The number format of this value is unclear.

Recommendation Consider documenting.

Client Comment *We do not have a specific format at the moment, the documentation is available in the CMTA specifications.*

```
7  +uint256 interestRate;
```

CVF-42. INFO

- **Category** Unclear behavior
- **Source** IDebtGlobal.sol

Description The format of these strings is unclear.

Recommendation Consider using structs of primitive types instead, or at least explaining the strings format in a documentation comment.

Client Comment *We do not have a specific format at the moment, the documentation is available in the CMTA specifications.*

```
11 +string maturityDate;
12 +string interestScheduleFormat;
13 +string interestPaymentDate;
```

```
17 +string issuanceDate;
18 +string couponFrequency;
```

```
24 +string rating;
```

CVF-43. INFO

- **Category** Suboptimal
- **Source** IDebtGlobal.sol

Recommendation Consider using a bit map for flags, that would allow adding more flags in the future without changing the structure.

Client Comment *It could be interesting, but it is not easy to use for non-technical person. We track this as a possible improvement in the future: issues/161.*

```
22 +bool flagDefault;
23 +bool flagRedeemed;
```

CVF-44. INFO

- **Category** Procedural
- **Source** OnlyDelegateCallModule.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.

Recommendation Consider specifying as "^0.8.0".

Client Comment *Keep to have the same notation as the rest of the code.*

```
1 +pragma solidity ^0.8.2;
```

CVF-45. FIXED

- **Category** Procedural
- **Source** CodeList.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.

Recommendation Consider specifying as "^0.8.0". Also relevant for: RuleEngine.sol, RuleWhiteList.sol.

Client Comment *The contract codeList has been removed (CVF-11). For others contracts, we change to a floating pragma, but w do not want to offer the possibility to deploy the contract with a version interior to the version 0.8.17, which is a version used for the tests and the development (same principle as for the CMTAT).*

```
3 3 pragma solidity 0.8.17;
```

CVF-46. FIXED

- **Category** Procedural
- **Source** CodeList.sol

Recommendation This contract could be turned into a library. Alternatively, as it only contains constants, the constants could be moved to the top level and the contract could be removed.

Client Comment *The constants were moved inside the contract whitelist, a file was added in the documentation to summarize all the used codes.*

```
5  5  abstract contract CodeList {
```

CVF-47. INFO

- **Category** Procedural
- **Source** CodeList.sol

Description There is no access modifier for these constants, so internal access will be used by default.

Recommendation Consider explicitly specifying an access level.

Client Comment *The access level was set to public*

```
7  7  uint8 constant CODE_ADDRESS_FROM_NOT_WHITELISTED = 20;
8  8  uint8 constant CODE_ADDRESS_TO_NOT_WHITELISTED = 30;
9  9  uint8 constant NO_ERROR = 0;
```

CVF-48. FIXED

- **Category** Unclear behavior
- **Source** RuleEngine.sol

Description These functions should emit some events.

Client Comment *Emit event as recommended, except for the function kill which will be removed. Moreover, it is no useful to store event in a destroyed smart contract.*

```
26  26  function setRules(
```

```
46  46  function clearRules() public onlyRole(RULE_ENGINE_ROLE) {
```

```
55  55  function addRule(IRule rule_) public onlyRole(RULE_ENGINE_ROLE) {
```

```
70  70  function removeRule(IRule rule_) public onlyRole(RULE_ENGINE_ROLE) {
```

```
146  146  function kill() public onlyRole(DEFAULT_ADMIN_ROLE) {
```

CVF-49. INFO

- **Category** Suboptimal
- **Source** RuleEngine.sol

Recommendation Without this check, the ”clearRules” function wouldn’t be necessary.

Client Comment *Yes, we want to have a separate function to clear all the rules, to avoid human errors.*

```
29 29  require(rules_.length != 0, "The array is empty");
```

CVF-50. FIXED

- **Category** Readability
- **Source** RuleWhiteList.sol

Recommendation Should be ”else if”.

Client Comment *The function has been modified as recommended.*

```
138 138  if (!whitelist[_to]) {
```

CVF-51. FIXED

- **Category** Readability
- **Source** RuleWhiteList.sol

Recommendation Should be ”else return”.

Client Comment *The function has been modified as recommended.*

```
142 142  return NO_ERROR;
```

CVF-52. FIXED

- **Category** Suboptimal
- **Source** RuleWhiteList.sol

Recommendation This could be simplified as: return _restrictionCode == CODE_ADDRESS_FROM_NOT_WHITELISTED || _restrictionCode == CODE_ADDRESS_TO_NOT_WHITELISTED;

Client Comment *The function has been simplified as recommended.*

```
148 148 if (
149 149     _restrictionCode == CODE_ADDRESS_FROM_NOT_WHITELISTED ||
150 150     _restrictionCode == CODE_ADDRESS_TO_NOT_WHITELISTED
151 151 ) {
152 152     return true;
153 153 }
154 154 return false;
```

ABDK
Consulting

About us

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.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting