Test

Table of Contents

Guideline

It is important that the tests can easily be improved and understood by others.
For each test file, the list of tests must be present

How write a test ?

The test must be follow the pattern AAA for the documentation and the structure
First, read this excellent document by Microsoft.
Here a little resume :

Term	Definition
Arrange	Arrange your objects, create and set them up as necessary.
Arrange - Assert	If you create assertion to check your arrange
Act	The function which is tested (and only this function !!!!)
Assert	All check to verify the result obtained by the call of the function(s) in the Act part.

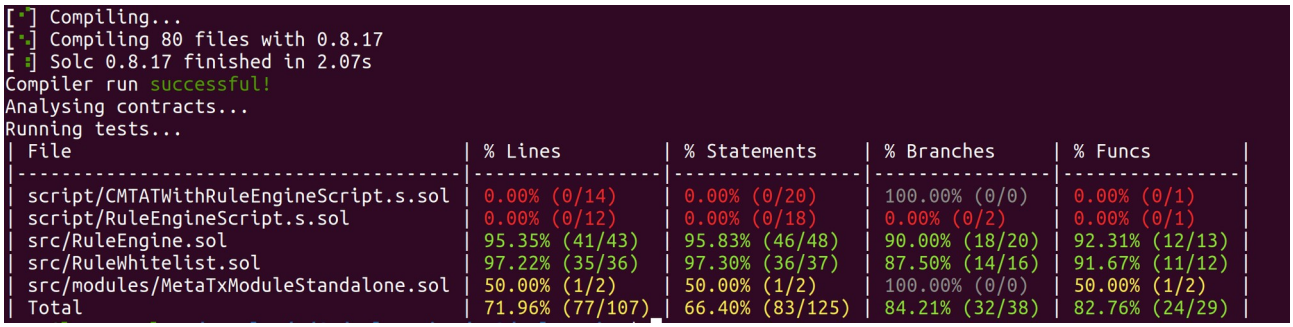
New test file
- Create a new tab with a new Id [A,B, C…..]
- Create a new tab for in the section checklist

New test
For each new test : add an entry after the previous ones in the corresponding table
Example : you create a new test called “testCanTransferIsTrue” in the file RuleWhitelist.t.sol. You add then an entry in the corresponding table. After that, add the test in the checklist too.

Below is an example of an entry in the table

id	Test function	Target function	Expected result	Actual result	conclusion	Improvement
[...previous test]						
25	testCanTransferIsTrue	The function which is tested	What is the result supposed to be returned by the function ? ??	[As expected] or[Not as exepected + the result]	[Ok, Not Ok]	Possible improvement for the test

Coverage



```
[⠒] Compiling...
[⠒] Compiling 80 files with 0.8.17
[⠃] Solc 0.8.17 finished in 2.07s
Compiler run successful!
Analysing contracts...
Running tests...
```

File	% Lines	% Statements	% Branches	% Funcs
script/CMTATWithRuleEngineScript.s.sol	0.00% (0/14)	0.00% (0/20)	100.00% (0/0)	0.00% (0/1)
script/RuleEngineScript.s.sol	0.00% (0/12)	0.00% (0/18)	0.00% (0/2)	0.00% (0/1)
src/RuleEngine.sol	95.35% (41/43)	95.83% (46/48)	90.00% (18/20)	92.31% (12/13)
src/RuleWhitelist.sol	97.22% (35/36)	97.30% (36/37)	87.50% (14/16)	91.67% (11/12)
src/modules/MetaTxModuleStandalone.sol	50.00% (1/2)	50.00% (1/2)	100.00% (0/0)	50.00% (1/2)
Total	71.96% (77/107)	66.40% (83/125)	84.21% (32/38)	82.76% (24/29)

Remark:

- MetaTxModuleStandalone is not tested (_msgSender() & _msgData()).

- The scripts are not tested.

Checklist

The checklist allows you to quickly check that all the functions are tested as well as to find the corresponding test

RuleWhiteList

File : **RuleWhiteList.sol**

Functions	Test id
addAddressesToTheWhitelist	A2/2, A2/4, A2/6, A5/2
removeAddressesFromTheWhitelist	A3/2, A3/4, A5/4
addAddressToTheWhitelist	A2/1, A2/3 (0x0 address), A2/5 (duplicate) A5/1
removeAddressFromTheWhitelist	A3/1, A3/3, A5/3
numberWhitelistedAddress	A1/8
addressIsWhitelisted	A1/1, A1/2, A1/3
isTransferValid	A1/6, A1/7
detectTransferRestriction	A1/9, A1/10, A1/11
canReturnTransferRestrictionCode	A1/4
messageForTransferRestriction	A1/5

File: OpenZeppelin library

Functions	Test id
grantRole	A6/1, A6/3
revokeRole	A6/2, A6/4

RuleEngine

File : **RuleEngine.sol**

Functions	Test id
setRules	B1/1, B1/13, B1/14, B1/15, B4/1
clearRules	B1/2, B4/2
addRule	B1/3, B1/10, B1/16, B1/17, B4/3
removeRule	B1/4, B1/5, B1/6, B1/12, B4/4
ruleLength	B1/7
rule	B1/8
rules	B1/9
detectTransferRestriction	B2/1, B2/2, B2/3
validateTransfer	B2/8

messageForTransferRestriction	B2/4, B2/5, B2/6
getRuleIndex	B1/18

CMTAT

File : **CMTAT.sol**
Here, we test only the integration with our own RuleEngine and Whitelist

Functions	Test id
mint	C9
transfer	C1, C2, C3, C4
detectTransferRestriction	C5, C6, c7, C8
messageForTransferRestriction	C5, C6, C7, C8

Test list

Test A

Target File : **RuleWhitelist.sol**

Test A1

File **: Rulewhite.t.sol**

id	Test function	Target function	Expected result	Actual result	conclusion	Improvement
1	testReturnFalseIfAddressNotWhitelisted	addressIsWhitelisted	For all addresses the functions return false	As expected	Ok	
2	testAddressIsIndicatedAsWhitelisted	addressIsWhitelisted	For the address added in the whitelist the function return true	As expected	Ok	
3	testAddressesIsIndicatedAsWhitelisted	addressIsWhitelisted	For all addresses in the whitelist the function return true	As expected	Ok	
4	testCanReturnTransferRestrictionCode	canReturnTransferRestrictionCode	Return true if the code exists, false otherwise	As expected	Ok	
5	testReturnTheRightMessageForAGivenCode	messageForTransferRestriction	For the different codes, the corresponding message is returned.	As expected	Ok	
6	testIsTransferValid	isTransferValid	Returns true for the different tested transfer	As expected	Ok	
7	testTransferDectedAsInvalid	isTransferValid	Returns false for the different tested transfer	As expected	Ok	
8	testNumberWhitelistedAdd	numberWhitelistedAdd	The number of address in the	As expected	Ok	

	ress	ress	whitelist is update after adding/deleting addresses.			
9	testDetectTransferRestrictionFrom	detectTransferRestriction	Detect that the from address is not whitelisted and returns the corresponding code.	As expected	Ok	
10	testDetectTransferRestrictionTo	detectTransferRestriction	Detect that the to address is not whitelisted and returns the corresponding code.	As expected	Ok	
11	testDetectTransferRestrictionOk	detectTransferRestriction	Detect that the transfer respects the rules and returns the corresponding code	As expected	As expected	

Test A2

File : RuleWhitelistAddTest.t.sol

id	Test function	Target function	Expected result	Actual result	conclusion	Improvement
1	testAddAddressToTheWhitelist	addAddressToTheWhitelist	The call of the function addressIsWhitelisted returns true for the added addresses + numberWhitelistsAddress return the number of addresses update	As expected	Ok	Warning : the test depends of two others functions addressIsWhitelisted. and numberWhitelistsAddress
2	testAddAddressToTheWhitelist	addAddressesToTheWhitelist	The call of the function addressIsWhitelisted returns true for the added addressses + numberWhitelistsAddress return the number of addresses update	As expected	Ok	Warning : the test depends of two others functions addressIsWhitelisted. and numberWhitelistsAddress
3	testCanAddAddressZeroToTheWhitelist	addAddressToTheWhitelist	The address 0x0 is added to the whitelist	As expected	Ok	
4	testCanAddAddressesZeroToTheWhitelist	addAddressesToTheWhitelist	All addresses are in the whitelist	As expected	Ok	
5	testAddAddressTwiceToTheWhitelist	addAddressToTheWhitelist	The transaction is reverted, the target address is still in the whitelist	As expected	Ok	
6	testAddAddressesTwiceToTheWhitelist	addAddressesToTheWhitelist	The new address is added, no change for the duplicate addresses	As expected	Ok	

Test A3

File : RuleWhitelistRemove.t.sol

id	Test function	Target function	Expected result	Actual result	conclusion	Improvement
1	testRemoveAddressesFromTheWhitelist	removeAddressFromTheWhitelist	The call of the function addressIsWhitelisted returns false for the removed addresses	As expected	Ok	
2	testRemoveAddressesFromTheWhitelist	removeAddressesFromTheWhitelist	The call of the function addressIsWhitelisted returns false for the removed addresses	As expected	Ok	
3	testRemoveAddressNotPresentFromTheWhitelist	removeAddressFromTheWhitelist	The transaction is reverted, no change in the whitelist	As expected	Ok	
4	testRemoveAddressesNotPresentFromTheWhitelist	removeAddressesFromTheWhitelist	The addresses present in the whitelist are correctly removed, no change for the address which was not present.	As expected	Ok	

Test A5

File RuleWhitelistAccessControl.t.sol

id	Test function	Target function	Expected result	Actual result	conclusion	Improvement
1	testCannotAttackerAddAddressToTheWhitelist	addAddressToTheWhitelist	The call is reverted because the sender has no role on the contract	As expected	Ok	
2	testCannotAttackerAddAddressesToTheWhitelist	addAddressesToTheWhitelist	The call is reverted because the sender has no role on the contract	As expected	Ok	
3	testCannotAttackerRemoveAddressFromTheWhitelist	removeAddressFromTheWhitelist	The call is reverted because the sender has no role on the contract	As expected	Ok	
4	testCannotAttackerRemoveAddressesFromTheWhitelist	removeAddressesFromTheWhitelist	The call is reverted because the sender has no role on the contract	As expected	Ok	

Test A6

File: RuleWhitelistAccessControlOZ.t.sol

id	Test function	Target function	Expected result	Actual result	conclusion	Improvement
1	testCanGrantRoleAsAdmin	grantRole	The target address has the attributed role	As expected	Ok	

2	testRevokeRoleAsAdmin	revokeRole	The target address has no longer the revoked role.	As expected	Ok	
3	testCannotGrantFromNonAdmin	grantRole	The call is reverted because the sender has not the right role	As expected	Ok	
4	testCannotRevokeFromNonAdmin	revokeRole	The call is reverted because the sender has not the right role	As expected	Ok	

Test B

Test B1

File: **RuleEngine.t.sol**

id	Test function	Target function	Expected result	Actual result	conclusion	Improvement
1	testCanSetRules	setRules	The rules are present in the list of rules.	As expected	Ok	
2	testCanClearRules	clearRules	No more rules are in the list of rules, this one is empty	As expected	Ok	
3	testCanAddRule	addRule	The target rule is present in the list of rules	As expected	Ok	
4	testCanRemoveNonExistantRule	removeRule	Noting changes	As expected	Ok	
5	testCanRemoveLatestRule	removeRule	The target rule is no longer in the list, the others rules are still present	As expected	Ok	
6	testCanRemoveRule	removeRule	The target rule is no longer in the list, the others rules are still present	As expected	Ok	
7	testRuleLength	ruleLength	Check the length of the rule before and after have added rules.	As expected	Ok	
8	testGetRule	rule	The function returns the right rule specified by its index.	As expected	Ok	
9	testGetRules	rules	The function returns all the rules	As expected	Ok	
10	testCannotAddRuleZeroAddress	addRule	The address 0 can not be add to the list, the transaction is reverted	As expected	OK	

12	testCanRemoveFirstRule	removeRule	The target rule, which is the first rule in the array, is correctly removed.	As expected	OK	
13	testCannotSetEmptyRulesT1	setRules	With a empty array in parameter, the transaction is reverted	As expected	OK	
14	testCannotSetEmptyRulesT2	setRules	The array in argument contains the zero address as a rule. The transaction is reverted.	As expected	OK	
15	testCannotSetRuleIfARuleIsAlreadyPresent	setRules	One of the rule is already present, the transaction is reverted.	As expected	OK	
16	testCannotAddARuleAlreadyPresent	addRule	The transaction is reverted.	As expected	OK	
17	testCanAddARuleAfterThisRuleWasRemoved	addRule	The target rule is present in the list of rules	As expected	OK	
18	testCanGetRuleIndex	getRuleIndex	The function return the right index for a given rule	As expected	OK	

Test B2

File: RuleEngineRestriction.t.sol

id	Test function	Target function	Expected result	Actual result	conclusion	Improvement
1	testCanDetectTransferRestrictionOK	detectTransferRestriction	The function return 0 because the transfer is valid	As expected	Ok	
2	testCanDetectTransferRestrictionWithFrom	detectTransferRestriction	Detect that the from address is not whitelisted and returns the corresponding code..	As expected	Ok	
3	testCanDetectTransferRestrictionWithTo	detectTransferRestriction	Detect that the to address is not whitelisted and returns the corresponding code	As expected	Ok	
4	testMessageForTransferRestrictionWithValidRC	messageForTransferRestriction	The message corresponding to the code is returned	As expected	Ok	
5	testMessageForTransferRestrictionNoRule	messageForTransferRestriction	The function detects that the code is unknown and returns the corresponding message.	As expected	Ok	
6	testMessageForTransferRestrictionWithUnknownRestrictionCode	messageForTransferRestriction	The function detects that the code is unknown and returns the corresponding message.	As expected	Ok	
7	testValidateTransferOK	validateTransfer	The transfer is valid, the function return true	As expected	Ok	
8	testValidateTransferRestricted	validateTransfer	The transfer is invalid, the function return false	As expected	Ok	

Test B4

File RuleEngineAccessControl.t.sol

id	Test function	Target function	Expected result	Actual result	conclusion	Improvement
1	testCannnotAttackerSetRules	setRules	The call is reverted because the sender has no role on the contract	As expected	Ok	
2	testCannnotAttackerClearRules	clearRules	The call is reverted because the sender has no role on the contract	As expected	Ok	
3	testCannotAttackerAddRule	addRule	The call is reverted because the sender has no role on the contract	As expected	Ok	
4	testCannotAttackerRemoveRule	removeRule	The call is reverted because the sender has no role on the contract	As expected	Ok	

Test B5

File: RuleEngineAccessControlOZ.t.sol

id	Test function	Target function	Expected result	Actual result	conclusion	Improvement
1	testCanGrantRoleAsAdmin	grantRole	The target address has the attributed role	As expected	Ok	

2	testRevokeRoleAsAdmin	revokeRole	The target address has no longer the revoked role.	As expected	Ok	
3	testCannotGrantFromNonAdmin	grantRole	The call is reverted because the sender has not the right role	As expected	Ok	
4	testCannotRevokeFromNonAdmin	revokeRole	The call is reverted because the sender has not the right role	As expected	Ok	

Test C

File : **CMTAT.t.sol**
Remark : we can consider this test as an integration test.

ID	Test function	Target function	Expected result	Actual result	conclusion	Improvement
1	testCannotTransferWithoutAddressWhitelisted	transfer	Revert because addresses are not whitelisted	As expected	Ok	
2	testCannotTransferWithoutFromAddressWhitelisted	transfer	Revert because address From is not whitelisted	As expected	Ok	
3	testCannotTransferWithoutToAddressWhitelisted	transfer	Revert because address To is not whitelisted	As expected	Ok	
4	testCanMakeATransferr	transfer	Successful transfer of tokens because FROM and TO are whitelisted	As expected	Ok	
5	testDetectAndMessageWithFromNotWhitelisted	F1) DetectTransferRestriction F2) messageForTransferRestriction	F1 returns the corresponding code when the address FROM is not whitelisted F2 returns the message corresponding to the code returned by F1.	As expected	Ok	
6	testDetectAndMessageWithToNotWhitelisted	F1) DetectTransferRestriction F2) messageForTransferRestriction	F1 returns the corresponding code when the address TO is not whitelisted F2 returns the message corresponding to the code returned by F1.	As expected	Ok	

7	testDetectAndMessageWithFromAndToNotWhited	F1) DetectTransferRestriction F2) messageForTransferRestriction	Same result that for the test *testCannotTranferIfToIsNotWhitelisted* but this time the address TO is not whitelisted too	As expected	Ok	
8	testDetectAndMessageWithAValidTransfer	F1) DetectTransferRestriction F2) messageForTransferRestriction	From and To are whitelisted, the transfer is valid. F1 returns the corresponding code with this case F2 returns the message corresponding to the code returned by F1.	As expected	Ok	
9	testCanMint	mint	The tokens were minted and the owner is the target address.	As expected	Ok	