SMART CONTRACT AUDIT

ZOKYO.

May 19th, 2022 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

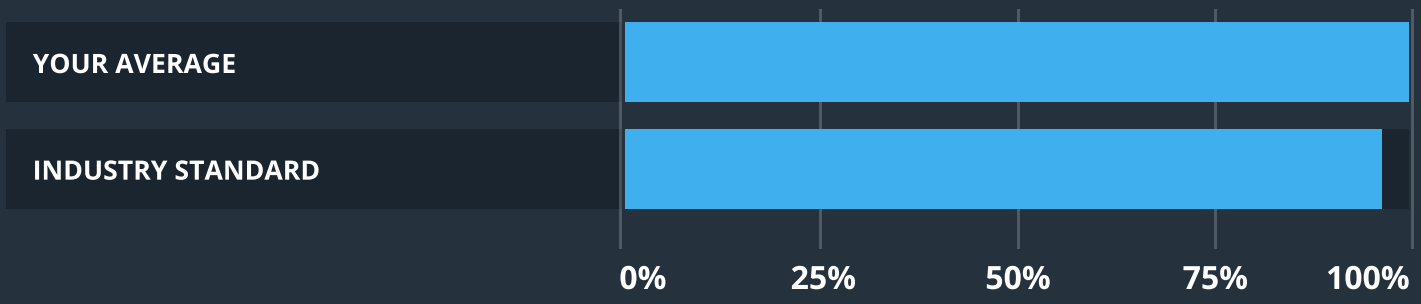
TECHNICAL SUMMARY

This document outlines the overall security of the Symbiosis smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Symbiosis smart contract codebase for quality, security, and correctness.

Contract Status

Testable Code



The testable code is 98%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a security of the contract we at Zokyo recommend that the Symbiosis team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Symbiosis repository.
Repository:
https://github.com/symbiosis-finance/contracts-audit-with-tests
Initial commit
3bc13a5622e1072b5c05db6f83f35893c15fef4
Last audited commit
f6245cea9a6941c853b7128b3703c7891cd9bdb4
Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- MetaRouteStructs.sol
- MetaRouterGateway.sol
- MetaRouter.sol
- Portal.sol
- RelayRecipientUpgradeable.sol
- SyntERC20.sol
- SyntFabric.sol
- Synthesis.sol
- Timelock.sol
- BridgeV2.sol
- AdminableUpgradeable.sol
- Wrapper.sol
- MulticallRouter.sol

Throughout the review process, care was taken to ensure that the contract:

- Implements and adheres to existing standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of resources, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Symbiosis smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

The audit has included updates of the MetaRouter to the version 3 (with all supportive changes within the repo) together with several modified contracts (together with the latest renamings of the external calls). The audit has shown the unclear logic during some utility contracts initialization, few validations missed and unclear external call functionality added. Nevertheless, after the conversation with the Symbiosis team all unclear functionality was verified and issues were successfully resolved. Although, auditor's team has verified the consistency of the codebase and test coverage after the applied changes. Thus, there were no malicious changes discovered and the functionality prooved to be working as intended after the regression checks.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

CRITICAL | RESOLVED

Incorrect usage of Initializable

RelayRecipientUpgradeable.sol: __RelayRecipient_init().
AdminableUpgradeable.sol: __Adminable_init().
Contracts inherit Initializable.sol, which is necessary for keeping that Proxy contract are initialized only once. The current version of @openzeppelin/contracts-upgradeable is 4.4.2 In this version Initializable has two modifiers: initializer and onlyInitializing. For internal initialize functions(such as __RelayRecipient_init()), a modifier “onlyInitializing” should be used, while modifier “initializer” should be used only on the top function call. Due to this, contracts which inherit RelayRecipientUpgradeable.sol or AdminableUpgradeable.sol cannot be initialized.

Recommendation:

Use the modifier “onlyInitializing” for internal initializing functions.

Post-audit:

Issue was verified with Symbiosis team and resolved by bumping OZ to 4.4.1. It was verified that the issue doesn’t affect the ability of the contract to be initialized. Although modifiers in internal functions were updated to drop support of @openzeppelin/contracts-upgradeable below v4.4.1.

HIGH | VERIFIED

Target addresses are not validated

MulticallRouter.sol: function multicall().
Addresses from array “_receiveSides” are not validated like in MetaRouter.sol (function metaRoute, lines 53, 83, 110), so that users are able to perform calls to any addresses.
Line 301,There is a call to factory to extract real token address for provided stoken. In case, there is no real token for stoken, system might me flooded with invalid requests (For each invalid request an event will be emitted. Events are more likely listened by backend, which will have to process these invalid transactions).
Atack vector: An attacker can impersonate stoken to pass any calls to this token and start sending transactions with this malicious token, which would generate invalid requests.

Recommendation:

Either verify, that addresses should not be validated, or add validate them like in MetaRouter.sol

MEDIUM | RESOLVED

Address of token is not verified when burning.
Synthesis.sol: functions burnSyntheticToken(), metaBurnSyntheticToken().
Variable “rtoken”, which is retrieved from SynFactory is not verified in order not be zero address. This means that any stoken can be passed, which can lead to unpredictable behavior of the protocol. One of the issues is that protocol might be flooded with invalid requests. Thus, in order to disallow users from passing invalid stokens, the retrieved rtoken should be validated.

Recommendation:
Validate, that “rtoken” is not zero address.

INFORMATIONAL | VERIFIED

Possible reentrancy
MetaRouter.sol: functions metaRoute(), externalCall(), metaMintSwap().
Functions perform external calls to provided addresses. It is possible to pass malicious address with reentrancy call or even perform a call to the address of MetaRouter with calldata of one the MetaRouter’s functions. Issue is marked as **info**, since it is unclear how exactly reentrancy can affect the operation of the contract(No storage variables are changed during functions call and contract is not supposed to store any tokens).

Recommendation:
Add a nonReentrant modifier to functions.

Post-audit:
It was verified, that such behavior of the contract is not risky, since the contract doesn’t store any tokens. Adding additional verifications will only increase gas spendings. Though the issue still needs to be represented in the report due to the unpredictable nature of external calls.

	MetaRouteStructs.sol	MetaRouter.sol	Portal.sol
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

	MetaRouterGateway.sol	SyntERC20.sol	SyntFabric.sol
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

	RelayRecipientUpgradeable.sol	Synthesis.sol	Timelock.sol
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

	AdminableUpgradeable.sol	BridgeV2.sol	Wrapper.sol
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Secured team

As part of our work assisting Symbiosis team in verifying the correctness of their contract code, our team was responsible for writing integration tests using Truffle testing framework.

Tests were based on the functionality of the code, as well as review of the Symbiosis contract requirements for details about issuance amounts and how the system handles these.

```
Should check bridgeV2 functions
  ✓ Should check withdraw fee (122ms)
  ✓ Should change mpc
BridgeV2 negative tests
  ✓ Should not allow set transmitter by anyone
  ✓ Should fail on change mpc with zero addres
  ✓ Should fail on receive request with untrusted transmitter
MetaRouter burn V2 tests
  ✓ Should check metaRoute burn V2 with 1inch swap
MetaRouterUSU native tests
  ✓ Should check native metaRoute USU
MetaRouter synth V2 tests
  ✓ Should check metaRoute synth V2
MetaRouterV2 native burn tests (first token is native, 2 tokens in first and final paths)
  ✓ Should check native burn metaRouteV2 (swapExactETHForTokens, swapExactTokensForTokens)
    (131ms)
    Should check native burn
  ✓ metaRouteV2(swapExactETHForTokensSupportingFeeOnTransferTokens,swapExactTokensForTokens
    SupportingFeeOnTransferTokens) (124ms)
MetaRouterV2 burn tests (2 tokens in first and final paths)
  ✓ Should check burn metaRouteV2 (swapExactTokensForTokens) (125ms)
  ✓ Should check burn metaRouteV2 (swapExactTokensForTokensSupportingFeeOnTransferTokens)
    (132ms)
  ✓ Should check burn metaRouteV2 (swapExactTokensForTokens) (147ms)
  ✓ Should check burn metaRouteV2 (swapExactTokensForTokensSupportingFeeOnTransferTokens)
    (285ms)
MetaRouterV2 burn native tests (3 tokens in the first and final paths)
  ✓ Should check burn native metaRouteV2 (swapExactTokensForTokens) (137ms)
  ✓ Should check burn metaRouteV2 (swapExactETHForTokensSupportingFeeOnTransferTokens) (145ms)
MetaRouterV2 burn final slippage triggering test
  ✓ Should check final slippage triggering in metaRoute burn V2 (144ms)
```

MetaRouter burn V2 first slippage triggering test
- ✓ Should check first slippage triggering in metaRoute burn V2

MetaRouter burn V2 second slippage triggering test
- ✓ Should check second slippage triggering in metaRoute burn V2 (97ms)

MetaRouter burn V2 skip final swap test
- ✓ Should check metaRoute burn V2 final swap skipping (121ms)

MetaRouter burn V2 skip first swap test
- ✓ Should check metaRoute burn V2 without first swap (131ms)

MetaRouterV2 incorrect first/second dex router address test

Get factories for contracts.
- ✓ Should check tx revert in metaRoute swap with gateway as dex

MetaRouterV2 incorrect first/second dex router address test
- ✓ Should check tx revert in case of gateway address as first router (38ms)
- ✓ Should check tx revert in case of gateway address as second router (58ms)
- ✓ Should check tx revert in case of gateway address as recipient (90ms)

MetaRouterV2 incorrect first/second dex router address test
- ✓ Should check tx revert in case of incorrect first dex router address
- ✓ Should check tx revert in case of incorrect second dex router address (56ms)

MetaRouterV2 incorrect first/second calldata test
- ✓ Should check tx revert in case of incorrect first calldata
- ✓ Should check tx revert in case of incorrect second calldata (54ms)

MetaRouter burn V2 (revert tx after fail on the source chain)
- ✓ Should check revert tx after fail on the source chain (87ms)

MetaRouter burn V2 (revert burn after fail on the dest chain)
- ✓ Should check revert burn after fail on the dest chain (158ms)

MetaRouter synthesize V2 (revert synthesize after fail on the dest chain)
- ✓ Should check revert synthesize after fail on the dest chain (134ms)

MetaRouterV2 burn tests (2 tokens in first and final paths)
- ✓ Should check burn metaRouteV2 (swapExactTokensForTokens) (87ms)

MetaRouter synth V2 tests (2 tokens in first, final paths)
- ✓ Should check metaRoute synth V2 (swapExactTokensForTokens) (141ms)
- ✓ Should check metaRoute synth V2 (swapExactTokensForTokensSupportingFeeOnTransferTokens) (140ms)

MetaRouterV2 synth native tests
- ✓ Should check metaRouteV2 native synth (swapExactETHForTokens) (145ms)
- ✓ Should check metaRouteV2 native synth (swapExactETHForTokensSupportingFeeOnTransferTokens) (153ms)

MetaRouter synth V2 tests (3 tokens in the first and final paths)
- ✓ Should check synth metaRouteV2 (swapExactTokensForTokens) (244ms)
- ✓ Should check burn metaRouteV2 (swapExactTokensForTokensSupportingFeeOnTransferTokens) (278ms)

MetaRouter synth native V2 tests (3 tokens in the first and final paths)
- ✓ Should check metaRoute synth native V2 (swapExactETHForTokens) (189ms)
- ✓ Should check metaRoute synth native V2 (swapExactETHForTokensSupportingFeeOnTransferTokens) (196ms)

MetaRouter synth V2 final slippage triggering test
- ✓ Should check final slippage triggering in metaRoute synth V2 (147ms)

MetaRouter synth V2 first slippage triggering test

Add liquidity.
- ✓ Should check first slippage triggering in metaRoute synth V2

MetaRouter synth V2 second slippage triggering test
- ✓ Should check second slippage triggering in metaRoute synth V2 (126ms)

MetaRouter synth V2 skip final swap test
- ✓ Should check metaRoute synth V2 final swap skipping (133ms)

MetaRouter synth V2 first swap skipping
- ✓ Should check metaRoute synth V2 without first swap (148ms)

MetaRouter synth V2 second swap skipping test
- ✓ Should check metaRoute synth V2 without second swap (110ms)

Test Upgradable
- ✓ BridgeV2 upgrade works (84ms)

Test invoking BridgeV2 via signature

Should check receiveRequestV2
- ✓ receiveRequestV2 succeed with valid signature (46ms)
- ✓ receiveRequestV2 failed with invalid signature
- ✓ receiveRequestV2 with signature should not allow double spend (49ms)

Should check snts
- ✓ Should fail on burnSyntheticToken when synthesis paused
- ✓ Should fail on burnSyntheticToken with amount under threshold
- ✓ Should check burnSyntheticToken (43ms)

MetaBurn tests
- ✓ Should check metaBurn (76ms)
- ✓ Should check metaBurn without swap (53ms)

metaBurn negative tests
- ✓ Should fail on metaBurn when amount under threshold
- ✓ Should fail on metaUnsynthesize with synthetic tokens emergencyUnburn (83ms)

MetaSynthesize tests
- ✓ Should check metaSynthesize (130ms)
- ✓ Should check fail on metaSynthesize with token that has no synt representation (92ms)
- ✓ Should check metaSynthesize without final swap (105ms)
- ✓ Should check metaSynthesize without swaps (60ms)

metaSynthesize negative tests
- ✓ Should check metaMint call when tokens have been already synthesized (86ms)

```
    ✓ Should fail on amount under threshold during metaSynthesize
Upgradable
    ✓ Portal upgrade works (119ms)
Should check snts
    ✓ Should check that portal returns correct version
Should check pause/unpause in portal
    ✓ Shouldn't pause by anyone in portal
    ✓ Shouldn't unpause by anyone in portal
    ✓ Should check pause/unpause in portal
setMetaRouter()
    ✓ shouldn't allow to set by anyone
    ✓ shouldn't allow to set zero address
    ✓ Should check metaRouter setting in portal
setWhitelistToken()
    ✓ shouldn't allow to set by anyone
    ✓ should set by owner
    ✓ should set by owner
Should check syntezation
    ✓ Should check syntesize req (55ms)
    ✓ Should fail on syntesize when paused
    ✓ Should fail on amount under threshold
    ✓ Should check fail on synthesizing token that is not in whitelist
    ✓ Should fail on synthesation token that has no synt representation (86ms)
Should check unsynthesize
    ✓ Should fail on unsynthesize by anyone
    ✓ Should fail on unsynthesize when paused (45ms)
    ✓ Should fail on double unsynthesize (59ms)
    ✓ Should check hard unsynthesize (45ms)
Revert burn request
sTestTokenBridging attached to 0x3A504b21A66022b58991DC8da9954ED188FFCc23
    ✓ Should check revertBurn (70ms)
sTestTokenBridging attached to 0x729c189387AEe5D732e13b6f889CDEcA624C44fe
    ✓ Should check revertBurn with zero revertable address (68ms)
revertBurn() negative tests
    ✓ should fail on revert burn when portal paused
    ✓ Should fail on revert burn when tokens are already transfered
    ✓ Should fail on revert burn when tx does not exist
    ✓ Should fail on revert burn when state does not open (61ms)
    ✓ Should fail on revert burn with not revertable address
Revert synthesize request
    ✓ Should check revert synthesize (64ms)
```

✓ Should check revert synthesize with zero address (66ms)
revertSynthesize() negative tests
✓ should fail on revert synthesize when synthesis paused
✓ Should fail on revert synthesize when tokens are already minted
✓ should fail on revert synthesize by not revertable address
Should check snts
✓ Should check revertSynthesize vulnerability (2 synthesis contracts on one chain) (75ms)
Test Upgradable
✓ BridgeV2 upgrade works (80ms)
Test invoking BridgeV2 via signature
Should check receiveRequestV2
✓ receiveRequestV2 succeed with valid signature (43ms)
✓ receiveRequestV2 failed with invalid signature
✓ receiveRequestV2 with signature should not allow double spend (57ms)
Should check snts
✓ Should fail on burnSyntheticToken when synthesis paused
✓ Should fail on burnSyntheticToken with amount under threshold
✓ Should check burnSyntheticToken
MetaBurn tests
✓ Should check metaBurn
metaBurn negative tests
✓ Should fail on metaBurn when amount under threshold
Should check snts
Should check metaMintSyntheticToken
✓ Should fail on meta mint by anyone
✓ Should fail on metaMint when paused
✓ Should fail on double mint (98ms)
supply of sTT for adr1 is 4989
✓ Should synt some sTT (126ms)
Should check snts
✓ Should check revert burn (43ms)
✓ Should fail on revert burn with not existed tx
✓ Should fail on revert burn call by anyone
Revert synthesize request
✓ Should check revert synthesize
revertSynthesize() negative tests
✓ should fail on revert synthesize when synthesis paused
✓ Should fail on revert synthesize when tokens are already minted
Upgradable
✓ Synthesis upgrade works (107ms)
Should check snts

✓ Should check version in synthesis
Should check mintSyntheticToken
✓ Should fail on mint by anyone
✓ Should fail on mint when paused
✓ Should fail on double mint
✓ Should synt some sTT
Should check pause/unpause in synthesis
✓ Shouldn't pause by anyone in synthesis
✓ Shouldn't unpause by anyone in synthesis
✓ Should check pause/unpause in synthesis
setMetaRouter()
✓ shouldn't allow to set by anyone
✓ shouldn't allow to set zero address
✓ Should check metaRouter setting in synthesis
setTokenThreshold()
✓ shouldn't allow to set by anyone
✓ Should check token threshold setting in synthesis
setFabric() negative tests
✓ shouldn't allow to set by anyone
✓ Shouldn't allow to set fabric second time
Upgradable
✓ Synthesis upgrade works (106ms)
Should check snts
✓ Should check version in synthesis
Should check mintSyntheticToken
✓ Should fail on mint by anyone
✓ Should fail on mint when paused
✓ Should fail on double mint
✓ Should synt some sTT
Should check pause/unpause in synthesis
✓ Shouldn't pause by anyone in synthesis
✓ Shouldn't unpause by anyone in synthesis
✓ Should check pause/unpause in synthesis
setMetaRouter()
✓ shouldn't allow to set by anyone
✓ shouldn't allow to set zero address
✓ Should check metaRouter setting in synthesis
setTokenThreshold()
✓ shouldn't allow to set by anyone
✓ Should check token threshold setting in synthesis
setFabric() negative tests

```
    ✓ shouldn't allow to set by anyone
    ✓ Shouldn't allow to set fabric second time
Native token syntesation test
    ✓ Should check syntesize req (49ms)
syntWithPermit() negative tests
    ✓ Should fail on amount under threshold during synthesizeWithPermit
    ✓ Should fail on synthesizeWithPermit when portal paused
    ✓ Should fail on synthesizeWithPermit with wrapper not in whitelist
Should check synthesation with permit
    ✓ Should check syntesize req (58ms)
syntWithPermit() negative tests
    ✓ Should fail on amount under threshold during synthesizeWithPermit
    ✓ Should fail on synthesizeWithPermit when portal paused
    ✓ Should fail on synthesizeWithPermit when portal paused
Timelock test
    ✓ Should check tx executing
    ✓ Should check setDelay
    ✓ Should check tx cancelling
    ✓ Should check setPendingAdmin
    ✓ Should check acceptAdmin


157 passing (1m)
```

Tests were verified to be consistent after the regression check. Also, the native coverage is sufficient for the security acceptance.

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	Uncovered Lines
MetaRouteStructs.sol	100	100	100	100	
MetaRouteStructsSolana.sol	100	100	100	100	
MetaRouterGateway.sol	100	50	100	100	
MetaRouterV2.sol	100	96.88	100	100	
MetaRouterV2Solana.sol	90	46.88	83.33	90.7	107,108,128,133
Portal.sol	98.8	94.74	100	98.9	581
RelayRecipientUpgradeable.sol	62.5	25	75	55.56	36,51,52,54
SyntERC20.sol	75	100	75	75	21
SyntFabric.sol	100	50	100	100	
SyntFabricSolana.sol	100	50	100	100	
Synthesis.sol	98.51	96.67	100	98.65	358
SynthesisSolana.sol	94.03	83.33	100	94.59	228,233,324,409
Timelock.sol	97.83	63.89	100	97.83	193
BridgeV2.sol	96.3	75	100	96.77	83
BridgeV2Solana.sol	90	62.5	93.33	91.43	86,181,182
AdminableUpgradeable.sol	60	25	50	57.14	3,14,23
Wrapper.sol	42.86	16.67	66.67	40	... 46,55,56,58
All files	**93.75**	**74.41**	**94.2**	**93.7**	

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Secured team

As part of our work assisting Symbiosis team in verifying the correctness of their contract code, our team was responsible for writing integration tests using Truffle testing framework.

Tests were based on the functionality of the code, as well as review of the Symbiosis contract requirements for details about issuance amounts and how the system handles these.

Contract: **RecipientUpgradeable**
- ✓ check msg.sender as trusted forwared
- ✓ check msg.data for trusted forwared (213ms)
- ✓ check msg.data for non-trusted forwared

Contract: **AdminUpgradeable**
- ✓ should check set admin permission
- ✓ should fail set admin permission
- ✓ should fail only owner or admin
- ✓ should fail only owner or admin

Contract: **BridgeV2**
- ✓ get current chainID
- ✓ check receiveRequestV2 (357ms)
- ✓ check set transmitter status
- ✓ ransmit request v2
- ✓ check change mpc
- ✓ check mpc
- ✓ check withdraw (266ms)
- ✓ check receive request v2 signed (395ms)

Contract: **MetaRouterV2**
- ✓ Should be able to swap
- ✓ Should meta mint swap (153ms)
- ✓ Should be able to meta route (103ms)
- ✓ Should be able to meta route 12 (183ms)
- ✓ metaRouteV2, empty swap call data, fail on relayer call with other side call data (324ms)
- ✓ metaRouterV2 fail first swapp (334ms)
- ✓ metaRouterV2 fail second swapp (338ms)
- ✓ Should meta mint swap, internal swap failed (324ms)
- ✓ Should meta mint swap, external swap failed (358ms)
- ✓ should call swap, external swap failed (318ms)
- ✓ should call swap, succeed (311ms)

Contract: **MetaRouterGateway**

```
    ✓ Should be setup correctly
    ✓ Should be able to claim tokens (55ms)
Contract: Portal
  Initialization
    ✓  Portal upgrade works (259ms)
  Should check unsynthesize
    ✓ Should fail on unsynthesize by anyone
    ✓ Should fail on unsynthesize when paused (45ms)
    ✓ Should fail on double unsynthesize (58ms)
    ✓ Should check hard unsynthesize (46ms)
  pause() and unpause()
    ✓ Shouldn't pause by anyone in portal
    ✓ Shouldn't unpause by anyone in portal
    ✓ Should check pause/unpause in portal
  should emit SynthesizeRequest
    ✓ Should fail with unauthorized token (312ms)
    ✓ Should fail with amount under threshold (545ms)
    ✓ Should check syntesize req (584ms)
    ✓ Should check syntesize req reverteble address is 0x0 (577ms)
    ✓ Should revert with unauthorized token address
    ✓ Should revert with lower token treshold
    ✓ Should revert when paused
  setMetaRouter()
    ✓ Should not allow non-owner account to change metarouter
    ✓ Should not allow zero account to be set as metarouter
    ✓ Should check metaRouter setting in portal
  setWhitelistToken()
    ✓ shouldn't allow to set by anyone
    ✓ should set by owner
    ✓ Should return correct portal version
  synthesizeNative
    ✓ Should revert with uninitialized token
  synthesizeNative
    ✓ Should revert when paused
  synthesizeWithPermit
  sTestToken attached to 0x2877c5d83Ac103177bA375f71AE5371d3808B175
    ✓ revert on  unauthorized token
  sTestToken attached to 0xBA90c4D3F1299E9fe744A9394CAB1E44DA001F0b
    ✓ Should reverts with Symb: amount under threshold
  sTestToken attached to 0x57E23B39AB72daD0A342EaBCc9df734518A64a2
    ✓ Should check syntesize req (581ms)
```

syntWithPermit() negative tests
sTestToken attached to 0x5034662844Aa4DE52aC210F9e215b6fa93d66845
✓ Should fail on amount under threshold during synthesizeWithPermit
sTestToken attached to 0x30760501864c379788FD6A303fb7b9Cf391fE5De
✓ Should fail on synthesizeWithPermit when portal paused
sTestToken attached to 0xf1d8ac14dB1C9c1Ef05bdd6D469121cc7a43dC15
✓ Should fail on synthesizeWithPermit when portal paused
revertSynthesize()
✓ Should reverts with Symb: caller is not the bridge
✓ Should reverts if paused
✓ Should successfully revert synthesize (259ms)
metaUnsynthesize
✓ Reverts with Symb: caller is not the bridge (1050ms)
✓ reverts with BridgeV2: call failed
✓ Should revert with Symb: synthetic tokens emergencyUnburn (802ms)
✓ should have final swap calldata equal 0 and return (809ms)
✓ should have final swap calldata bigger then 0 and swap (823ms)
metaSynthesize
✓ Should return correct request count (41ms)
✓ Should fail unauthorized token (344ms)
✓ Should fail amount under threshold (585ms)
✓ Should use chain2Address as revertable address if addresS(0) parsed (39ms)
revertBurnRequest
✓ should revert with Symb: Real tokens already transfered (70ms)
✓ should fail on revert burn when portal paused
✓ Should fail on revert burn when tx does not exist
✓ Should check revertBunRequest sucesfull with event emitting
✓ Should fail on revert burn with not revertable address
synthesizeNative
✓ Should revert when paused
✓ Should revert with authorised token
✓ Symb: amount under threshold
✓ Should have correct balance in wrapper contract (42ms)
✓ Should emit sendSynthesizeRequest with correct args (40ms)
initialize
✓ Should initialize with whitelisted token differnt from 0x0 (137ms)
Contract: SyntERC20
✓ Should be setup correctly
✓ Should be able to mint from only owner account
BigNumber { value: "50" }
✓ Should be able to burn from only owner account
Contract: SyntFabric

```
    ✓ Should be setup correctly
    ✓ Should be able to set representation
    ✓ Should be able to get synt representation
    ✓ Should be able to get real representation
    ✓ Should be able to get syn representation by key
    ✓ Should be able mint synt tokens (49ms)
    ✓ Should be able to burn  synt tokens (56ms)
Contract: Synthesis
    ✓ Should be setup correctly
    ✓ Should be able to get version recipient
    ✓ Should be able to be paused
    ✓ Should be able to be unpause
    ✓ Should be able to be set Meta Router
    ✓ Should be able to be set fabric
    ✓ Should be able to be set token threshold
    ✓ Should be able to be able to mint synt token (114ms)
    ✓ Should be able to be able to revert synt Token (41ms)
    ✓ Should be able to be able to burn synt token (111ms)
    ✓ Should be able to be able to burn synt token with ADDRESS ZERO as revertable
      address (88ms)
    ✓ Should be able to be able to meta mint synt token (98ms)
    ✓ Should be able to be able to meta mint synt token with swap tokens (213ms)
    ✓ Should be able to be able to meta burn (152ms)
    ✓ Should be able to be able to meta burn with ADDRESS ZERO as revertable address
      (150ms)
    ✓ it should be able to revert burn (132ms)
Contract: TimeLock
    ✓ check constructor requires
    ✓ get que tx check
    ✓ should call setDelay (50ms)
    ✓ accept admin
    ✓ call setPendingAdmin
    ✓ queuq transaction
    ✓ cancle transaction
    ✓ execute transaction (84ms)
Contract: Wrapper
    ✓ shout depoit ether, not as trusted forwared
    ✓ shout depoit ether, as trusted forwared
    ✓ hould withdraw ether, fail
    ✓ should withdraw ether, fail
    ✓ check _msgData as trusted forwarder
    ✓ check _msgData, as not trusted forwarder
  120 passing (2m)
```

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	Uncovered Lines
MetaRouteStructs.sol	100	100	100	100	
MetaRouterGateway.sol	100	100	100	100	
MetaRouterV2.sol	100	100	100	100	
Portal.sol	100	100	100	100	
RelayRecipientUpgradeable.sol	100	100	100	100	
SyntERC20.sol	100	100	100	100	
SyntFabric.sol	100	100	100	100	
Synthesis.sol	100	96.67	100	100	
Timelock.sol	97.96	86.11	100	97.96	194
BridgeV2.sol	100	100	100	100	
AdminableUpgradeable.sol	100	100	100	100	
Wrapper.sol	100	100	100	100	
All files	**99.69**	**96.55**	**100**	**99,71**	

Tests were verified and updated according to the latest changes in the contracts repo.

We are grateful to have been given the opportunity to work with the Symbiosis team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the Symbiosis team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.