Blaize.Security

August 29th, 2023 / V. 1.0

LOCK ON

SMART CONTRACT AUDIT

TABLE OF CONTENTS

AUDIT RATING

SCORE **9.6**/10

The scope of the project includes Lockon Finance set of contracts:

ExchangeIssuanceZeroEx.sol
Operator.sol
Pausable.sol
SetTokenCreator.sol
ExtendModuleBase.sol
TradeModule.sol
BasicIssuanceModule.sol

Repository:

https://gitlab.com/lockon-finance/core-contracts

Branch: main

Initial commit:

- afd8a8aa3b9a728fc1c4b407a493b3f0fc20bbed

Final commit:

- ce49613b76bde57c56ee508388fb8bedac1bea67

TECHNICAL SUMMARY

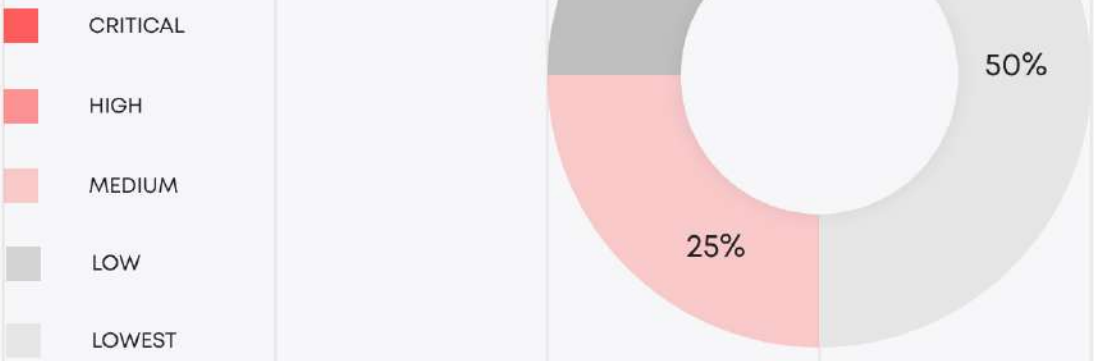
During the audit, we examined the security of smart contracts for the Lockon Finance protocol. Our task was to find and describe any security issues in the smart contracts of the platform. This report presents the findings of the security audit of the **Lockon Finance** smart contracts conducted between **June 13th, 2023** and **June 26th, 2023.** The second audit iteration was conducted between **August 21st, 2023** and **August 22nd, 2023.**

Testable code

The code is 100% testable, which corresponds to the industry standard of 95%.

The scope of the audit includes the unit test coverage, which is based on the smart contract code, documentation and requirements presented by the Lockon Finance team. The coverage is calculated based on the set of Hardhat framework tests and scripts from additional testing strategies. However, to ensure the security of the contract, the Blaize.Security team suggests that the Lockon Finance team launch a bug bounty program to encourage further active analysis of the smart contracts.

THE GRAPH OF VULNERABILITIES DISTRIBUTION:



The table below shows the number of the detected issues and their severity. A total of 8 problems were found. 8 issues were fixed or verified by the Lockon Finance team.

	FOUND	FIXED/VERIFIED
Critical	0	0
High	0	0
Medium	2	2
Low	2	2
Lowest	4	4

SEVERITY DEFINITION

Critical

The system contains several issues ranked as very serious and dangerous for users and the secure work of the system. Requires immediate fixes and a further check.

High

The system contains a couple of serious issues, which lead to unreliable work of the system and migh cause a huge data or financial leak. Requires immediate fixes and a further check.

Medium

The system contains issues that may lead to medium financial loss or users' private information leak. Requires immediate fixes and a further check.

Low

The system contains several risks ranked as relatively small with the low impact on the users' information and financial security. Requires fixes.

Lowest

The system does not contain any issues critical to the secure work of the system, yet is relevant for best practices

AUDITING STRATEGY AND TECHNIQUES APPLIED/PROCEDURE

We have scanned this smart contract for commonly known and more specific vulnerabilities:

- Unsafe type inference;
- Timestamp Dependence;
- Reentrancy;
- Implicit visibility level;
- Gas Limit and Loops;
- Transaction-Ordering Dependence;
- Unchecked external call - Unchecked math;
- DoS with Block Gas Limit;
- DoS with (unexpected) Throw;
- Byte array vulnerabilities;
- Malicious libraries;
- Style guide violation;
- ERC20 API violation;
- Uninitialized state/storage/ local variables;
- Compile version not fixed.

Procedure

We checked the contract for the following parameters:

- Whether the contract is secure;
- Whether the contract corresponds to the documentation;
- Whether the contract meets the best practices in the efficient use of gas, code readability.

Automated analysis:

Scanning contracts by several publicly available automated analysis tools such as Mythril, Solhint, Slither, and Smartdec. Manual verification of all the issues found with tools.

Manual audit:

Manual analysis of smart contracts for security vulnerabilities. We checked smart contract logic and compared it with the one described in the documentation.

EXECUTIVE SUMMARY

Blaize Security team conducted an audit of the Lockon Finance protocol. This protocol is a fork of the Set Protocol with additional contracts. The main added contract, ExchangeIssuanceZeroEx, focuses on swaps between tokens and setToken using the 0x protocol.

The audit aimed to review the additions made to the forked protocol and ensure the new contracts were safe to use. The auditors evaluated the 0x protocol integration and confirmed that it complied with the best practices. They confirmed that the protocol was implemented correctly, allowing seamless and reliable token swaps. Besides, it should be noted that the audit did not cover the Set Protocol itself.

The audit found no critical or high issues. One medium issue was connected to swapping tokens in a loop, which can cause a transaction to revert. The auditors also identified other issues, such as parameters verification, the pragma version, unlimited allowance, unused struct, and usage of the 0x protocol. The Lockon Finance team verified or fixed all issues that were found.

The overall security of the protocol is high enough. The contracts are well-written and contain a sufficient NatSpec. Additionally, the Lockon Finance team updated the unit tests from the Set Protocol to match the changes made. These tests are high-quality and cover every aspect of the contracts. The auditors also prepared their own set of unit tests to verify the security and correctness of the changes and tested the new contracts.

The reaudit was committed to check the correct fee charge on ce49613b76bde57c56ee508388fb8bedac1bea67 commit.

The auditors examined the implemented fee charge during the second audit iteration, finding two issues. The Lockon Finance team successfully verified both of them.

To check the implementation of fees, the auditors tested several extra scenarios, including:

- Normal scenario to check the correctness of fee calculation.
- Scenario with different amounts of tokens including dust and large amounts.
- Scenario with tokens with different decimals.
- Scenario with different fee percentages.

The code has to withstand all the scenarios. Also, the overall security of the fee implementation is high enough.

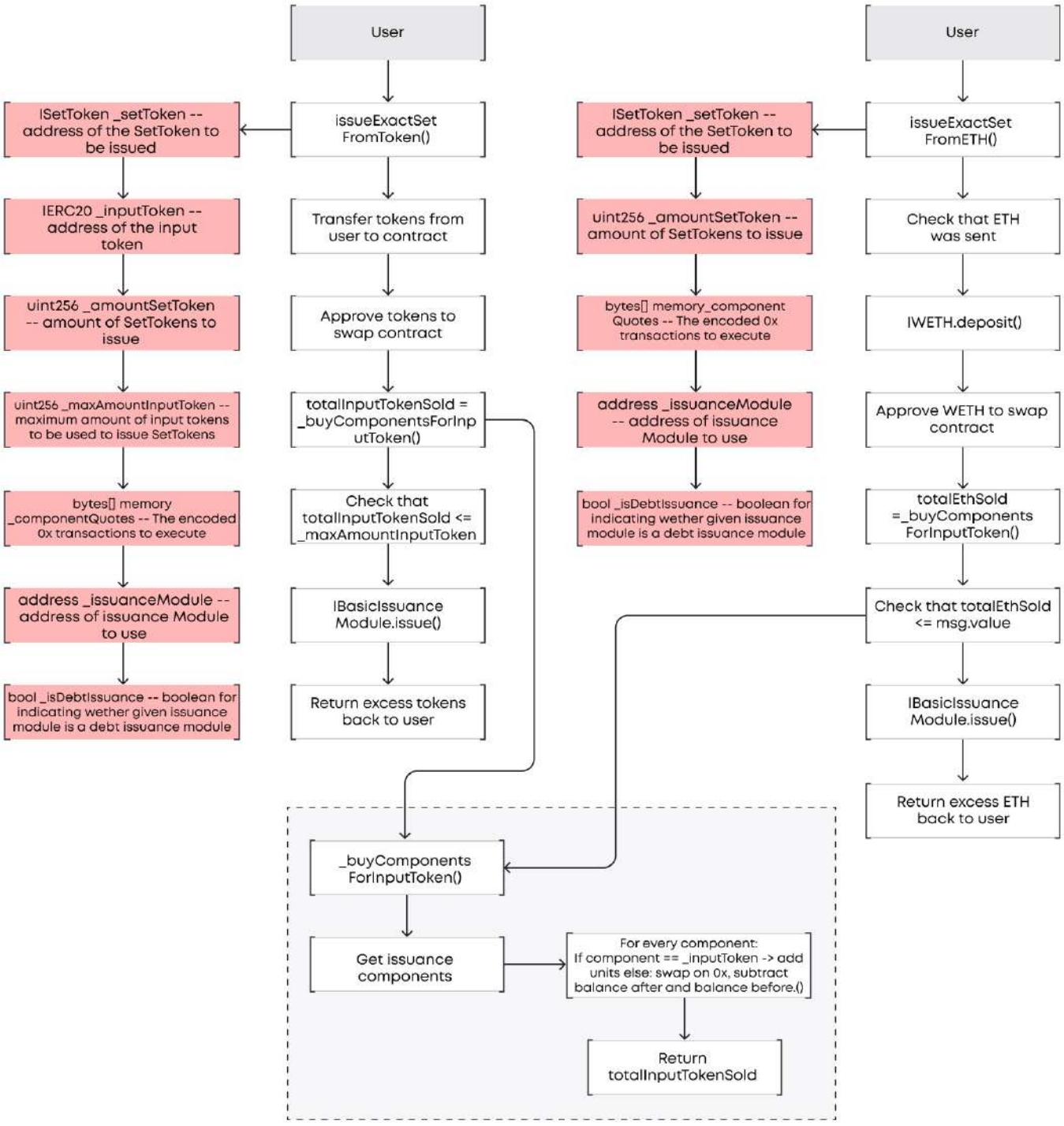
	RATING
Security	9.7
Gas usage and logic optimization	9.5
Code quality	10
Test coverage	9.4
Total	9.6

Note

The scope excludes the original Set Protocol V2 codebase and includes only a review of the changes performed by the Lockon team and integrity checks/tests. The original audit by OpenZeppelin of Set Protocol covers SetToken, SetTokenCreator, and Controller contracts with initial modules only. Therefore, all v2 changes, new modules, and integrations are left unaudited. Blaize Security team recommends providing an audit of it.

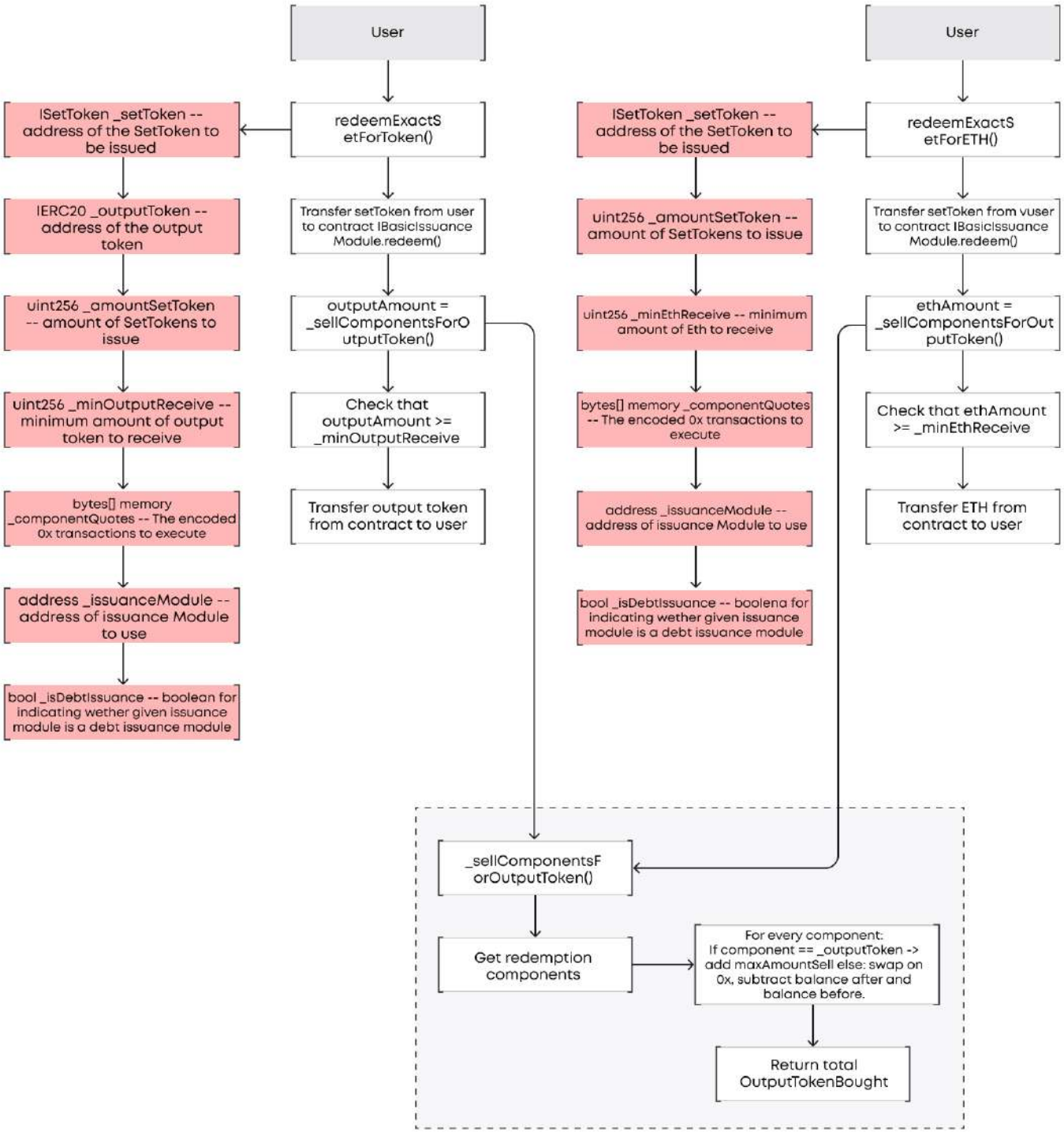
LOCKON FINANCE

ExchangeIssuanceZeroEx.sol



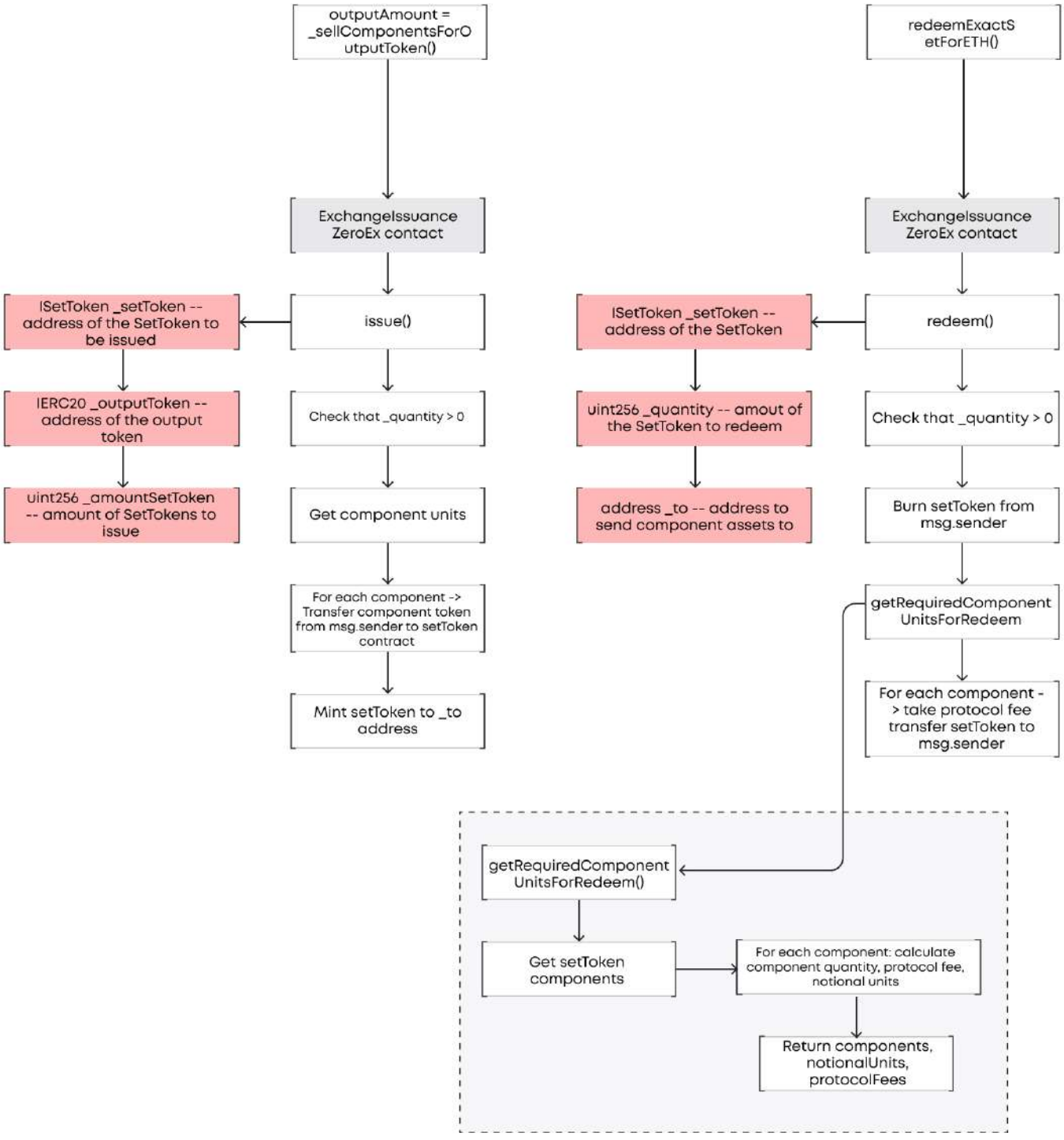
LOCKON FINANCE

ExchangeIssuanceZeroEx.sol



LOCKON FINANCE

BasicIssuanceModule.sol



COMPLETE ANALYSIS (1ST AUDIT ITERATION)

MEDIUM-1	✔ Verified

Transaction could revert.

BasicIssuanceModule.sol: redeem();
ExchangeIssuanceZeroEx.sol: _buyComponentsForInputToken(), _sellComponentsForOutputToken();
Token transfers/swaps are invoked in a loop. If a problem occurs during the loop, the entire transaction will revert. It would be better to implement internal accounting to calculate the amount of tokens to withdraw/swap.

Recommendation:

Implement internal accounting for functions **OR** verify that requirements are met before initiating a transaction.

From client:

The Lockon team verified that swap should be executed in a single transaction, rather than one by one.

LOW-1	✔ Resolved

Parameters are not validated in constructor.

ExchangeIssuanceZeroEx.sol: constructor().
Set the necessary contract addresses for the correct operation of functions within the contract. It is recommended to verify that contracts are not equal to the zero address during setting to prevent missing contract addresses, especially when variables are immutable.

Recommendation:

Add zero address check.

Post audit:
Parameters validation was added in constructor.

LOW-2	✔ Verified

Unlimited allowance.

ExchangeIssuanceZeroEx.sol: approveToken(), line 130, _safeApprove(), line 337.
An unlimited allowance is granted to the spender in the approveToken() function of the ExchangeIssuanceZeroEx contract. Although approval is given to a whitelisted issuance module or immutable swap target, approving a specific amount before transferring is recommended.

Recommendation:

Approve exact transfer amount before transfer.

From client:
ExchangeIssuanceZeroEx does not store ERC20 tokens, so there are no ERC20 tokens to withdraw.

LOWEST-1	✔ Verified

Use the latest version of pragma.

Older versions of Solidity may contain bugs and vulnerabilities and may not be optimized for gas usage. It is recommended to use the latest version of Solidity, which is 0.8.20 at the time of writing.

Recommendation:

Update contracts to latest version.

From client:
Lockon team verified that contract version will be as it is due to forked project.

LOWEST-2	✔ Verified

Convert API response to calldata.

ExchangeIssuanceZeroEx.sol: _fillQuote()
According to the 0x protocol documentation, the response is an object containing information to send to the contract. The _fillQuote() function in the ExchangeIssuanceZeroEx contract takes bytes and sends them to the 0x contract. Therefore, the conversion of responses from the API to bytes should be clarified.

Recommendation:

Verify response conversion to bytes.

From client:
Data value is used from API response.

LOWEST-3	✔ Resolved

Unused struct.

ExchangeIssuanceZeroEx.sol: IssuanceModuleData
Struct has no use in contract. Verify that it will be used in future versions or remove it.

Recommendation:

Verify the usage of a Struct **OR** remove it from contract.

Post audit:

Struct was removed.

COMPLETE ANALYSIS (2ND AUDIT ITERATION)

MEDIUM-1	✔ Verified

Admin is able to set >100% fee on setToken.

Controller.sol: addFee()
The Admin is able to set the fee on the module to 100% to receive all tokens from the user. The same applies when setting the value to more than 100%. However, this can cause the protocol to stop working because of an overflow problem.

Recommendation:

Add max fee value to the function.

Post audit:

Lockon team acknowledged this issue:
- Lockon team verified that fee changing would be controlled by multisig. This control minimizes the risk of misuse.
- There will be no changes to the code to preserve the integrity of the legacy code of SetToken

COMPLETE ANALYSIS (2ND AUDIT ITERATION)

LOWEST-1	✔ Verified

Revert if swap quotes is not in order like components

ExchangeIssuanceZeroEx.sol: _sellComponentsForOutputToken()
The error occurs if, for example, set token components [comp1, comp2] and swap quotes [sell comp1, sell comp2]. The user invokes redeemExactSetForToken with IERC20 _outputToken = component2. In this case, if quote [sell comp1, sell comp2] is used, there will be a subtraction overflow error (line 423) as we get more component2. It is not a security problem, but it could be a problem for a user if they get the exact invalid quote. As a result, they cannot redeem the tokens.

Recommendation:

Verify usage of quotes.

Post audit:

Lockon team verified that the frontend service uses values from the getRequiredRedemptionComponents function to ensure the correct arguments for calling redeemExactSetForToken. Additionally, the code on line 423 is a post-verification to ensure the swap has been executed correctly.

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

CODE COVERAGE AND TEST RESULTS FOR ALL FILES, PREPARED BY LOCKON FINANCE TEAM

SushiSwap TradeModule Integration [@forked-mainnet]

#trade

when trading a Default component on Sushiswap (version 2 adapter)

- ✓ should transfer the correct components to the SetToken (8138ms)
- ✓ should transfer the correct components from the SetToken (2524ms)

UniswapExchangeV2 TradeModule Integration [@forked-mainnet]

#trade

when trading a Default component on Uniswap version 2 adapter

- ✓ should transfer the correct components to the SetToken (9145ms)
- ✓ should transfer the correct components from the SetToken (2776ms)

ExchangeIssuanceZeroEx [@forked-mainnet]

#constructor

- ✓ verify state set properly via constructor (491ms)

when exchange issuance is deployed

#withdrawTokens()

- ✓ should succeed (266ms)
- ✓ should send erc20 amounts to receiver (324ms)
- ✓ should send ether to receiver (338ms)

when the caller is not the owner

- ✓ should revert

when issuance module is basicIssuanceModule

#approveSetToken

- ✓ should update the approvals correctly (83ms)

when the input token is not a set

- ✓ should revert

#receive

- ✓ should revert when receiving ether not from the WETH contract

#approveTokens

- ✓ should update the approvals correctly (69ms)

when the tokens are approved twice

- ✓ should update the approvals correctly (112ms)

when the spender address is not a whitelisted issuance module

- ✓ should revert

#issueExactSetFromToken
- ✓ should issue correct amount of set tokens (2565ms)
- ✓ should use correct amount of input tokens (2949ms)

when the input swap does not use all of the input token amount
- ✓ should return surplus input token to user (3340ms)

when the input token is also a component
- ✓ should issue correct amount of set tokens (2418ms)
- ✓ should use correct amount of input tokens (2753ms)

when an invalid issuance module address is provided
- ✓ should revert

when a position quote is missing
- ✓ should revert (108ms)

when invalid set token amount is requested
- ✓ should revert

when a component swap spends too much input token
- ✓ should revert (86ms)

When the zero ex router call fails with normal revert errror
- ✓ should forward revert reason correctly (63ms)

when a component swap yields insufficient component token
- ✓ should revert (81ms)

when a swap call fails
- ✓ should revert (79ms)

#issueExactSetFromETH
- ✓ emits an ExchangeIssue log (1891ms)
- ✓ should issue the correct amount of Set to the caller (3024ms)
- ✓ should use the correct amount of ether from the caller (3027ms)

when not all eth is used up in the transaction
- ✓ should return excess eth to the caller (3296ms)

when too much eth is used
- ✓ should revert (81ms)

when wrong number of component quotes are used
- ✓ should revert (66ms)

when input ether amount is 0
- ✓ should revert

when amount Set is 0
- ✓ should revert (97ms)

when input ether amount is insufficient
- ✓ should revert (74ms)

#redeemExactSetForToken
- ✓ should redeem the correct number of set tokens (3474ms)
- ✓ should give the correct number of output tokens (2890ms)

when invalid set token amount is requested
- ✓ should revert

when an invalid issuance module address is provided
- ✓ should revert

when a position quote is missing
- ✓ should revert (185ms)

when the output swap yields insufficient DAI
- ✓ should revert (147ms)

when a swap call fails
- ✓ should revert (128ms)

when the output token is also a component
- ✓ should succeed (2824ms)
- ✓ should redeem the correct number of set tokens (1839ms)
- ✓ should give the correct number of output tokens (1824ms)

#redeemExactSetForEth
- ✓ should redeem the correct number of set tokens (3513ms)
- ✓ should disperse the correct amount of eth (2555ms)

when the swaps yield insufficient weth
- ✓ should revert (171ms)

when the swaps yields excess weth
- ✓ should disperse the correct amount of eth (2848ms)

when the swap consumes an excessive amount of component token
- ✓ should revert (148ms)

when issuance module is debtIssuanceModule

#approveSetToken
- ✓ should update the approvals correctly (83ms)

when the input token is not a set
- ✓ should revert

#receive
- ✓ should revert when receiving ether not from the WETH contract (67ms)

#approveTokens
- ✓ should update the approvals correctly (430ms)

when the tokens are approved twice
✓ should update the approvals correctly (142ms)
when the spender address is not a whitelisted issuance module
✓ should revert
#issueExactSetFromToken
✓ should issue correct amount of set tokens (2270ms)
✓ should use correct amount of input tokens (3163ms)
when the input swap does not use all of the input token amount
✓ should return surplus input token to user (3105ms)
when the input token is also a component
✓ should issue correct amount of set tokens (3100ms)
✓ should use correct amount of input tokens (3006ms)
when an invalid issuance module address is provided
✓ should revert
when a position quote is missing
✓ should revert (75ms)
when invalid set token amount is requested
✓ should revert (103ms)
when a component swap spends too much input token
✓ should revert (83ms)
When the zero ex router call fails with normal revert errror
✓ should forward revert reason correctly (69ms)
when a component swap yields insufficient component token
✓ should revert (77ms)
when a swap call fails
✓ should revert (76ms)
#issueExactSetFromETH
✓ emits an ExchangeIssue log (3934ms)
✓ should issue the correct amount of Set to the caller (3130ms)
✓ should use the correct amount of ether from the caller (4032ms)
when not all eth is used up in the transaction
✓ should return excess eth to the caller (3681ms)
when too much eth is used
✓ should revert (99ms)
when wrong number of component quotes are used
✓ should revert (69ms)

when input ether amount is 0
✓ should revert
when amount Set is 0
✓ should revert (95ms)
when input ether amount is insufficient
✓ should revert (73ms)
#redeemExactSetForToken
✓ should redeem the correct number of set tokens (4338ms)
✓ should give the correct number of output tokens (3452ms)
when invalid set token amount is requested
✓ should revert
when an invalid issuance module address is provided
✓ should revert
when a position quote is missing
✓ should revert (189ms)
when the output swap yields insufficient DAI
✓ should revert (198ms)
when a swap call fails
✓ should revert (174ms)
when the output token is also a component
✓ should succeed (3148ms)
✓ should redeem the correct number of set tokens (3694ms)
✓ should give the correct number of output tokens (3799ms)
#redeemExactSetForEth
✓ should redeem the correct number of set tokens (5406ms)
✓ should disperse the correct amount of eth (5015ms)
when the swaps yield insufficient weth
✓ should revert (188ms)
when the swaps yields excess weth
✓ should disperse the correct amount of eth (3956ms)
when the swap consumes an excessive amount of component token
✓ should revert (191ms)

BasicIssuanceModule [@forked-mainnet]

#initialize
✓ should enable the Module on the SetToken (219ms)
✓ should properly set the issuance hooks (219ms)

when the caller is not the SetToken manager
✓ should revert
when SetToken is not in pending state
✓ should revert (66ms)
when the SetToken is not enabled on the controller
✓ should revert
#removeModule
✓ should revert
#issue
when the components are WBTC and WETH
when there are no hooks
when not paused
✓ should issue the Set to the recipient (819ms)
✓ should have deposited the components into the SetToken (817ms)
✓ should emit the SetTokenIssued event (833ms)
when the issue quantity is extremely small
✓ should transfer the minimal units of components to the SetToken (2039ms)
when a SetToken position is not in default state
✓ should revert
when one of the components has a recipient-related fee
✓ should revert (76ms)
when the issue quantity is 0
✓ should revert
when the SetToken is not enabled on the controller
✓ should revert
when paused
✓ should revert
When paused once and unpaused
✓ should issue the Set to the recipient (2304ms)
when a preIssueHook has been set
when not paused
✓ should properly call the pre-issue hooks (988ms)
✓ should emit the SetTokenIssued event (1384ms)
when paused
✓ should revert
When paused once and unpaused
✓ should properly call the pre-issue hooks (981ms)

#redeem
when the components are WBTC and WETH
✓ should redeem the Set (1011ms)
✓ should have deposited the components to the recipients account (1420ms)
✓ should have subtracted from the components from the SetToken (1002ms)
✓ should emit the SetTokenRedeemed event (1617ms)
when the issue quantity is extremely small
✓ should transfer the minimal units of components to the SetToken (1231ms)
when the issue quantity is greater than the callers balance
✓ should revert
when one of the components has a recipient-related fee
✓ should revert (124ms)
when a SetToken position is not in default state
✓ should revert
when the issue quantity is 0
✓ should revert (556ms)
when the SetToken is not enabled on the controller
✓ should revert
when set the redeem fee
✓ should send fee for all tokens (2113ms)
when not set the redeem fee
✓ should not send fees (1104ms)
When zero is set after setting the fees once
✓ should not send fees (1869ms)
#pause
When the owner
When not paused
✓ should paused status be true (40ms)
When paused
✓ should revert
When the wrong owner
When not paused
✓ should revert
When paused
✓ should revert

#unpause
When the owner
When not paused
✓ should revert
When paused
✓ should paused status be false
When the wrong owner
When not paused
✓ should revert
When paused
✓ should revert
TradeModule [@forked-mainnet]
#constructor
✓ should have the correct controller (467ms)
when there is a deployed SetToken with enabled TradeModule
#initialize
✓ should enable the Module on the SetToken (213ms)
when the caller is not the SetToken manager
✓ should revert
when the module is not pending
✓ should revert
when the SetToken is not enabled on the controller
✓ should revert
#trade
when trading a Default component on Kyber
when the module is initialized
when there is a protocol fee charged
When the caller is not the manager or operator
✓ should revert
When the caller is the operator
When the operator is already registered
✓ should transfer the correct components to the SetToken (2431ms)
When an operator is registered once and deleted.
✓ should revert (61ms)
When the caller is the manager
✓ should transfer the correct components to the SetToken (3200ms)
✓ should transfer the correct components from the SetToken (2553ms)

- ✓ should transfer the correct components to the exchange (2300ms)
- ✓ should transfer the correct components from the exchange (1993ms)
- ✓ should update the positions on the SetToken correctly (3583ms)

  when there is a protocol fee charged
- ✓ should transfer the correct components minus fee to the SetToken (2178ms)
- ✓ should transfer the correct components from the SetToken to the exchange (2214ms)
- ✓ should update the positions on the SetToken correctly (3391ms)
- ✓ should emit the correct ComponentExchanged event (3087ms)

  when receive token is more than total position units tracked on SetToken
- ✓ should transfer the correct components minus fee to the SetToken (2815ms)
- ✓ should update the positions on the SetToken correctly (2777ms)

  when send token is more than total position units tracked on SetToken
- ✓ should transfer the correct components from the SetToken (2786ms)
- ✓ should update the positions on the SetToken correctly (4135ms)

  when SetToken is locked
- ✓ should revert (61ms)

  when the exchange is not valid
- ✓ should revert

  when quantity of token to sell is 0
- ✓ should revert

  when quantity sold is more than total units available
- ✓ should revert (38ms)

  when slippage is greater than allowed
- ✓ should revert (78ms)

  when the caller is not the SetToken manager
- ✓ should revert

  when SetToken is not valid
- ✓ should revert

  when there is a protocol not fee charged
- ✓ should not transfer the correct components minus fee to the SetToken (2517ms)

  when module is not initialized
- ✓ should revert (47ms)

  when trading a Default component on Uniswap
- ✓ should transfer the correct components to the SetToken (3413ms)
- ✓ should transfer the correct components from the SetToken (2733ms)

- ✓ should update the positions on the SetToken correctly (2983ms)
  when path is through multiple trading pairs
- ✓ should transfer the correct components to the SetToken (2199ms)
  when trading a Default component with a transfer fee on Uniswap
- ✓ should transfer the correct components to the SetToken (2409ms)
- ✓ should transfer the correct components from the SetToken (2719ms)
- ✓ should update the positions on the SetToken correctly (3139ms)
  when path is through multiple trading pairs
- ✓ should transfer the correct components to the SetToken (2738ms)
  when trading a Default component on Uniswap version 2 adapter
  when path is through one pair and swaps exact tokens for tokens
- ✓ should transfer the correct components to the SetToken (3052ms)
- ✓ should transfer the correct components from the SetToken (2554ms)
- ✓ should update the positions on the SetToken correctly (2906ms)
  when path is through one pair and swaps for exact tokens
- ✓ should transfer the correct components to the SetToken (2446ms)
  when path is through multiple trading pairs and swaps exact tokens for tokens
- ✓ should transfer the correct components to the SetToken (2965ms)
  when path is through multiple trading pairs and swaps for exact tokens
- ✓ should transfer the correct components to the SetToken (3475ms)
- ✓ should update the positions on the SetToken correctly (2785ms)
  when trading a Default component on One Inch
- ✓ should transfer the correct components to the SetToken (2985ms)
- ✓ should transfer the correct components from the SetToken (2527ms)
- ✓ should transfer the correct components to the exchange (2474ms)
- ✓ should transfer the correct components from the exchange (2592ms)
- ✓ should update the positions on the SetToken correctly (2732ms)
  when function signature does not match 1inch
- ✓ should revert (89ms)
  when send token does not match calldata
- ✓ should revert (50ms)
  when receive token does not match calldata
- ✓ should revert (51ms)
  when send token quantity does not match calldata
- ✓ should revert (50ms)
  when min receive token quantity does not match calldata
- ✓ should revert (55ms)

when trading a Default component on 0xAPI
- ✓ should transfer the correct components to the SetToken (2746ms)
- ✓ should transfer the correct components from the SetToken (3395ms)
- ✓ should transfer the correct components to the exchange (2770ms)
- ✓ should transfer the correct components from the exchange (3679ms)
- ✓ should update the positions on the SetToken correctly (3283ms)

when function signature is not supported
- ✓ should revert (60ms)

when send token does not match calldata
- ✓ should revert (58ms)

when receive token does not match calldata
- ✓ should revert (54ms)

when send token quantity does not match calldata
- ✓ should revert (60ms)

when min receive token quantity does not match calldata
- ✓ should revert (55ms)

when trading a Default component on Uniswap V3 using UniswapV3ExchangeAdapte
- ✓ should transfer the correct components to the SetToken (3021ms)
- ✓ should transfer the correct components from the SetToken (3266ms)
- ✓ should update the positions on the SetToken correctly (3058ms)

when path is through multiple trading pairs
- ✓ should transfer the correct components to the SetToken (3828ms)

when trading a Default component on Uniswap V3 using UniswapV3ExchangeAdapterV2
- ✓ should transfer the correct components to the SetToken (3505ms)
- ✓ should transfer the correct components from the SetToken (3188ms)
- ✓ should update the positions on the SetToken correctly (4089ms)

when fixedIn is false
- ✓ should transfer the correct components to the SetToken (2873ms)
- ✓ should transfer the correct components from the SetToken (3227ms)
- ✓ should update the positions on the SetToken correctly (2808ms)

when path is through multiple trading pairs
- ✓ should transfer the correct components from the SetToken (4108ms)
- ✓ should transfer the correct components to the SetToken (3218ms)

when fixIn is false
- ✓ should transfer the correct components from the SetToken (3390ms)
- ✓ should transfer the correct components to the SetToken (2992ms)

#removeModule
✓ should remove the module (293ms)
Operator [@forked-mainnet]
#addOperator
When the caller is the owner
When adding one operator
✓ should be added to the operators (49ms)
When adding two operator
✓ should be added to the operators (71ms)
When the caller is not the owner
✓ should revert
#removeOperator
When the caller is the owner
When removing a registered operator.
✓ should be removed from the operator register (39ms)
When deleting an unregistered operator
✓ should revert
When there are two operator registrations
✓ Should only remove one operator (72ms)
When the caller is not the owner
✓ should revert
#getOperator
When get a registered operator address.
✓ should return the operator address
When get an unregistered operator
✓ should revert
#getAllOperatorsLength
✓ should return the number of operators
#isOperator
When get a registered operator address.
✓ should return true
When get an unregistered operator
✓ should return false
SetTokenCreator [@forked-mainnet]
constructor
✓ should have the correct controller (864ms)

when there is a SetTokenCreator
#create
When the caller is the owner
✓ should properly create the Set (950ms)
✓ should enable the Set on the controller (441ms)
✓ should emit the correct SetTokenCreated event (423ms)
when no components are passed in
✓ should revert
when no components have a duplicate
✓ should revert
when the component and units arrays are not the same length
✓ should revert
when a module is not approved by the Controller
✓ should revert
when no modules are passed in
✓ should revert
when the manager is a null address
✓ should revert
when a component is a null address
✓ should revert
when a unit is 0
✓ should revert
When the caller is not the owner
✓ should revert

239 passing (10m)

TEST COVERAGE RESULTS

FILE	% STMTS	% BRANCH	% FUNCS
ExchangeIssuanceZeroEx.sol	100	100	100
Operator.sol	100	100	100
Pausable.sol	100	100	100
SetTokenCreator.sol	100	100	100
ExtendModuleBase.sol	100	100	100
TradeModule.sol	100	100	100
BasicIssuanceModule.sol	100	100	100
All files	**100**	**100**	**100**

CODE COVERAGE AND TEST RESULTS FOR ALL FILES, PREPARED BY BLAIZE SECURITY TEAM (1ST AUDIT ITERATION)

BasicIssuanceModule changes
Pausable contract integration in BasicIssuanceModule
✓ Unable to issue when contract is on pause
ExchangeIssuanceZeroEx
Testing ExchangeIssuanceZeroEx with tow ERC20 tokens
✓ Check the state variables of the contract after deployment (260ms)
✓ Approval for one component token
✓ Approval for multiple components tokens
✓ Approval for multiple components tokens via SetToken
✓ SetToken issuance relative to the number of ERC20 tokens provided (144ms)
✓ Redeems an exact amount of SetTokens for an ERC20 token (181ms)
✓ Withdraw slippage to selected address: not contract owner
✓ Withdraw slippage to selected address (48ms)
Testing ExchangeIssuanceZeroEx with one ERC20 token and ETH
✓ SetToken issuance relative to the number of ETH provided (333ms)
✓ SetToken issuance relative to the number of ETH zero amount
✓ Redeems an exact amount of SetTokens for ETH (172ms)
Checking the validity of modules
✓ Checking the module when buying for ERC20
✓ Checking the module when selling for ERC20
✓ Checking the module when buying for ETH
✓ Checking the module when selling for ETH
✓ Checking the module when approve tokens
Checking the validity of modules
✓ Receiving an error during ETH transfer

Operator

Operator management
✓ Should add a new operator
✓ Should remove an operator
✓ Should fail to add an existing operator
✓ Should fail to remove a non-existing operator
✓ Should get an operator by index
✓ Should get the number of operators
✓ Should only allow the owner to add or remove operators

Pausable

Pausable operations

- ✓ Should return `paused`
- ✓ Should pause and unpause
- ✓ Should pause and unpause only by owner
- ✓ Shouldn't pause and unpause when already done

SetTokenCreator

SetToken creation

- ✓ Should create a new SetToken
- ✓ Should create a new SetToken only by owner
- ✓ Should fail if components length is 0
- ✓ Should fail if components and units lengths are not the same
- ✓ Should fail if components have duplicates
- ✓ Should fail if modules length is 0
- ✓ Should fail if manager address is zero
- ✓ Should fail if component address is zero
- ✓ Should fail if unit is less than or equal to 0
- ✓ Should fail if module is not enabled by the controller

TradeModule changes

TradeModule ExtendModuleBase integration

- ✓ Should revert with OnlyManagerOrOperator (70ms)
- ✓ Trade function should invoke (74ms)

41 passing (2s)

TEST COVERAGE RESULTS

FILE	% STMTS	% BRANCH	% FUNCS
ExchangeIssuanceZeroEx.sol	97.62	68.97	100
Operator.sol	100	100	100
Pausable.sol	100	100	100
SetTokenCreator.sol	100	100	100
ExtendModuleBase.sol	100	66.67	100
TradeModule.sol *	16	31.25	40
BasicIssuanceModule.sol *	90.91	53.85	87.5
All files	**86.36**	**74.4**	**89.64**

* Only changes/additions to contract were tested including the integration of changes in the protocol.

Note

As Lockon Finance team has good coverage, that was reviewed by audtiors. Therefore, Blaize team decided to focus on testing changes from Set Protocol, main functionality of new contracts and integration with 0x protocol.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES, PREPARED BY BLAIZE SECURITY TEAM (2ND AUDIT ITERATION)

BasicIssuanceModule changes
  Pausable contract integration in BasicIssuanceModule
✓ Unable to issue when contract is on pause
ExchangeIssuanceZeroEx
  # Testing ExchangeIssuanceZeroEx with tow ERC20 tokens
✓ Check the state variables of the contract after deployment (229ms)
✓ Approval for one component token
✓ Approval for multiple components tokens
✓ Approval for multiple components tokens via SetToken
✓ SetToken issuance relative to the number of ERC20 tokens provided (183ms)
✓ Redeems an exact amount of SetTokens for an ERC20 token (190ms)
✓ Withdraw slippage to selected address: not contract owner
✓ Withdraw slippage to selected address (44ms)
Testing ExchangeIssuanceZeroEx with one ERC20 token and ETH
✓ SetToken issuance relative to the number of ETH provided (299ms)
✓ SetToken issuance relative to the number of ETH zero amount
✓ Redeems an exact amount of SetTokens for ETH (185ms)
Checking the validity of modules
✓ Checking the module when buying for ERC20
✓ Checking the module when selling for ERC20
✓ Checking the module when buying for ETH
✓ Checking the module when selling for ETH
✓ Checking the module when approve tokens
Checking the validity of modules
✓ Receiving an error during ETH transfer
Fee changes
  # BasicIssuanceModule -- Check that fee works
✓ Set fee and check result (1% -> 100% -> 10k%) (283ms)
✓ Set fee to 0% and check flow (113ms)
✓ Set fee to 1% and check flow (134ms)
✓ Set fee to 100% and check flow (116ms)
✓ Check 1% fee with different tokens set (314ms)
✓ Check 2% fee with different token decimals in set (329ms)
✓ Check 1% fee with redeem minimal set amount (131ms)

ExchangeIssuanceZeroEx -- Check that fee works
✓ Issue tokens and redeem (257ms)

Operator

Operator management
✓ Should add a new operator
✓ Should remove an operator
✓ Should fail to add an existing operator
✓ Should fail to remove a non-existing operator
✓ Should get an operator by index
✓ Should get the number of operators
✓ Should only allow the owner to add or remove operators

Pausable

Pausable operations
✓ Should return `paused`
✓ Should pause and unpause
✓ Should pause and unpause only by owner
✓ Shouldn't pause and unpause when already done

SetTokenCreator

SetToken creation
✓ Should create a new SetToken
✓ Should create a new SetToken only by owner
✓ Should fail if components length is 0
✓ Should fail if components and units lengths are not the same
✓ Should fail if components have duplicates
✓ Should fail if modules length is 0
✓ Should fail if manager address is zero
✓ Should fail if component address is zero
✓ Should fail if unit is less than or equal to 0
✓ Should fail if module is not enabled by the controller

TradeModule changes

TradeModule ExtendModuleBase integration
✓ Should revert with OnlyManagerOrOperator (64ms)
✓ Trade function should invoke (67ms)

49 passing (4s)

TEST COVERAGE RESULTS

FILE	% STMTS	% BRANCH	% FUNCS
ExchangeIssuanceZeroEx.sol	97.67	66.67	100
Operator.sol	100	100	100
Pausable.sol	100	100	100
SetTokenCreator.sol	100	100	100
ExtendModuleBase.sol	100	66.67	100
TradeModule.sol	16	31.25	40
BasicIssuanceModule.sol	91.67	53.57	87.5
All files	**86.36**	**74.4**	**89.64**

DISCLAIMER

The information presented in this report is an intellectual property of the customer, including all the presented documentation, code databases, labels, titles, ways of usage, as well as the information about potential vulnerabilities and methods of their exploitation. This audit report does not give any warranties on the absolute security of the code. Blaize.Security is not responsible for how you use this product and does not constitute any investment advice.

Blaize.Security does not provide any warranty that the working product will be compatible with any software, system, protocol or service and operate without interruption. We do not claim the investigated product is able to meet your or anyone else's requirements and be fully secure, complete, accurate, and free of any errors and code inconsistency.

We are not responsible for all subsequent changes, deletions, and relocations of the code within the contracts that are the subjects of this report.

You should perceive Blaize.Security as a tool, which helps to investigate and detect the weaknesses and vulnerable parts that may accelerate the technology improvements and faster error elimination.