

Table of Contents

1. Executive Summary
2. Code review
 1. Review of the Specification
 2. Manual Review of Code
 3. Comparison to Specification
3. Testing and automated analysis
 1. Test Coverage Analysis
 2. Symbolic Execution (Automated Code Path Evaluation)
4. Itemized Recommendations & Best Practices Review

Executive Summary

Bramah Systems, LLC. was engaged by Set Protocol in early December of 2018 in order to conduct a security review of the Set Protocol smart contracts. This assessment was conducted over the course sixty person-hours.

The codebase under review (as provided by the Set Protocol team) represents a continuous body of work. This review pertains to the security posture of the Set Protocol as of commit hash [672e1de6f84d57eac9c2d97f9bc181c18ba75e82](#). The scope of Bramah's engagement covered all relevant smart contracts contained within the "contracts" directory. Other elements contained within the [set-protocol-contracts](#) repository were not reviewed.

Both the Set Protocol specification and codebase indicate the usage of multiple third party libraries. Elements of the Kyber, Taker Wallet, and 0x V2 Protocols are utilized within the operations of the Set Protocol, with "exchange wrappers" serving as a conduit between the decentralized exchanges and the Set Protocol Core. As these third party libraries pose unique security impact, each is detailed within the following report and their interactions are noted.

Various methodologies of analysis were used throughout this review. In particular, manual code inspection and static analysis utilizing formal methods were applied. Dynamic analysis and fuzz testing were utilized during the course of analysis to aid discovery of vulnerabilities which would only become apparent during execution. Manual inspection was additionally used in cases of suspected false positives, confirming (or denying) the vulnerability suggested by the tooling.

Throughout the review, these tools are utilized in best-effort to attempt to unearth potential security vulnerabilities. This report is not a formal endorsement of these tools, the organizations that support them, or a testament to their accuracy. These tool sets were used at the discretion of the reviewers where deemed appropriate for use.

Code Review

Review of Specification

The Set Protocol specification (located [within the whitepaper](#)) details various facets of the Set Protocol implementation and how various contracts interact. Most importantly, a number of key aspects of the protocol are defined (within "Smart Contracts"), which operate as pillars for this review.

These smart contract definitions reflect various assumptions that the course of this review aimed to validate. For instance, in the case of *Vault*, a key assumption is that "*Vault's accounting*

interfaces are only available to Core", validation of such is critical to the continued successful operation of the protocol.

Following the smart contract definitions exists "*Variant ERC20 Standard Considerations*", which similarly presents numerous objectives which must be satisfied (namely those pertaining to decimal differences between presently existing quantities of tokens and token pausability). These assumptions similarly present outsized security considerations, and were included within the scope of review. The Set Protocol itself includes functionality to avoid behavior with a potentially undefined impact to these considerations, namely the avoidance of accepting tokens with non-zero transfer fees and the presence of a "*natural unit*".

As Set Protocol notes throughout the document, many considerations discussed in the white paper are part of an evolving body of work, and may not inherently be reflected in the present version of the codebase. As a result of such, not all statements made in the whitepaper have been inherently tested in the scope of this review, and certain aspects of the specification could not be tested (e.g. Interest-Generating Sets).

Manual Code Inspection

Manual code inspection revealed extensive focus upon overall structure and readability of the codebase. All functionality is documented and possesses unit tests, with a near 1:1 ratio of code to comments.

Clear barriers are established to ensure the principle of least privilege is followed and secure by design guidelines are followed throughout.

The contract makes extensive usage of external libraries (created by OpenZeppelin, KyberNetwork, and 0x). This practice enables relatively minimal modifications to be made. Libraries ensure that low-hanging fruit often associated with typographical errors and simplistic oversights are removed.

Comparison to Specification

Multiple checks and constants are presented through the Truffle project to ensure intended actions are successfully performed. With extensive unit testing, each function has proper representation to validate it performs as specified within the whitepaper. Other than noted anomalies contained within itemized recommendations, Bramah found no deviations from the specification that would present security concerns.

Testing and Automated Analysis

Test Coverage Analysis

Symbolic Execution (Automated Code Path Evaluation)

Itemized Recommendations & Best Practices

Implicit Visibility Levels Set

In version 0.4.25 of Solidity, the default function visibility levels are as follows:

1. Contracts: Public
2. Interfaces: External
3. State Variables: Internal

In a contract, the fallback function can be external or public.

In an interface, all the functions should be declared as external.

Each function should have a defined function visibility to prevent confusion. A relevant mitigation strategy involves declaring a visibility level, removing any potential for ambiguity of the overall visibility of the function.

File: contracts/core/lib/auction-price-libraries/IAuctionPriceCurve.sol
Line: 47

File: contracts/core/lib/auction-price-libraries/IAuctionPriceCurve.sol
Line: 60

File: contracts/core/RebalancingSetToken.sol
Line: 54

File: contracts/core/RebalancingSetToken.sol
Line: 56

File: contracts/core/RebalancingSetToken.sol
Line: 55

File: contracts/core/RebalancingSetToken.sol
Line: 53

File: contracts/external/0x/AssetProxy/libs/LibAssetProxyErrors.sol
Line: 34

File: contracts/external/0x/AssetProxy/libs/LibAssetProxyErrors.sol
Line: 26

File: contracts/external/0x/AssetProxy/libs/LibAssetProxyErrors.sol
Line: 33

File: contracts/external/0x/AssetProxy/libs/LibAssetProxyErrors.sol
Line: 27

File: contracts/external/0x/AssetProxy/libs/LibAssetProxyErrors.sol
Line: 30

File: contracts/external/0x/AssetProxy/libs/LibAssetProxyErrors.sol
Line: 35

File: contracts/external/0x/AssetProxy/libs/LibAssetProxyErrors.sol
Line: 28

File: contracts/external/0x/AssetProxy/libs/LibAssetProxyErrors.sol
Line: 29

File: contracts/external/0x/Exchange/libs/LibEIP712.sol
Line: 23

File: contracts/external/0x/Exchange/libs/LibEIP712.sol
Line: 26

File: contracts/external/0x/Exchange/libs/LibEIP712.sol
Line: 29

File: contracts/external/0x/Exchange/libs/LibOrder.sol
Line: 28

File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 26

File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 58

File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 52

File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 28

File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 61

File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 48

File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 38

File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 51
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 62
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 30
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 29
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 68
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 57
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 33
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 54
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 27
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 45
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 39
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 65
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 66
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 37
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 42
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 67
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 34
File: contracts/external/0x/Exchange/libs/LibExchangeErrors.sol
Line: 53
File: contracts/mocks/tokens/NoXferReturnTokenMock.sol
Line: 10
File: contracts/mocks/tokens/InvalidReturnTokenMock.sol
Line: 9

Unsafe Array Length Manipulation

If possible, one should changing the length of the dynamic array directly. Large array lengths can lead to storage collisions and potentially modification of data outside the confines of the array. Multiple mitigation strategies, documented below, are capable of being used to remediate this.

1. Use `uint[] storage arrayName = new uint[](7)` to create a new array of the desired length.
2. Use `delete arrayName` to clear a dynamic array.
3. Use `.push()` (instead of `.length++`) to write to the end of the array.
4. Use `.pop()` (instead of `.length--`) to delete the last element of the dynamic array.

Multiplication After Division

Solidity operates only with integers. Thus, if the division is done before the multiplication, the rounding errors can increase dramatically. Mitigation should take place in the form of multiplication prior to division.

File: contracts/core/RebalancingSetToken.sol

Line: 886

File: contracts/core/RebalancingSetToken.sol

Line: 874

File: contracts/core/RebalancingSetToken.sol

Line: 473

File: contracts/core/RebalancingSetToken.sol

Line: 840

ERC 20 "Approve" Function Usage

As the Set Protocol team makes clear within their comments, one should only use the approve function of the ERC-20 standard to change allowed amount to 0 or from 0, having validated the transaction has been successfully mined. Relevant mitigation includes using approval and allowance steps, and remains largely contested in the overall Ethereum space.

File: contracts/mocks/lib/ERC20WrapperMock.sol

Lines: 20-28

File: contracts/mocks/tokens/NoXferReturnTokenMock.sol

Lines: 93-95

File: contracts/mocks/tokens/StandardTokenWithFeeMock.sol

Lines: 125-131

File: contracts/mocks/tokens/InvalidReturnTokenMock.sol

Lines: 115-125

Excess Gas Consumption

Excess gas consumption may occur when state variables (.balance or .length) are used in the condition of a for or while loop. Every iteration of loop consumes extra gas with these state variables present. In order to mitigate this excess consumption, if .balance, or .length are used several times, holding their value in a local variable is more gas efficient.

File: contracts/lib/AddressArrayUtils.sol

Lines: 304-308

File: contracts/lib/AddressArrayUtils.sol

Lines: 161-166

File: contracts/lib/AddressArrayUtils.sol

Lines: 303-309

File: contracts/lib/AddressArrayUtils.sol

Lines: 75-77

File: contracts/lib/AddressArrayUtils.sol

Lines: 343-345

File: contracts/lib/AddressArrayUtils.sol

Lines: 147-152

File: contracts/lib/AddressArrayUtils.sol

Lines: 143-146

File: contracts/lib/AddressArrayUtils.sol

Lines: 155-160

File: contracts/lib/AddressArrayUtils.sol

Lines: 323-327

File: contracts/core/Vault.sol
Lines: 243-251
File: contracts/core/Vault.sol
Lines: 282-290
File: contracts/core/Vault.sol
Lines: 322-331
File: contracts/core/Vault.sol
Lines: 204-212
File: contracts/core/exchange-wrappers/TakerWalletWrapper.sol
Lines: 95-109
File: contracts/core/extensions/CoreIssuance.sol
Lines: 444-454
File: contracts/core/extensions/CoreIssuance.sol
Lines: 394-412
File: contracts/core/extensions/CoreIssuance.sol
Lines: 482-484
File: contracts/core/lib/OrderLibrary.sol
Lines: 210-221
File: contracts/core/TransferProxy.sol
Lines: 115-124
File: contracts/core/RebalancingSetToken.sol
Lines: 865-878
File: contracts/core/RebalancingSetToken.sol
Lines: 475-522
File: contracts/core/RebalancingSetToken.sol
Lines: 763-777
File: contracts/core/SetToken.sol
Lines: 109-147
File: contracts/core/modules/IssuanceOrderModule.sol
Lines: 641-657
File: contracts/core/modules/IssuanceOrderModule.sol
Lines: 321-382
File: contracts/core/modules/IssuanceOrderModule.sol
Lines: 611-620

Non-initialized Return Values

In the case of a non-initialized return value, the default value will be returned (even in the event of failure). If the return value of the function is not required, mitigation should include not specifying return in the function signature.

File: contracts/mocks/tokens/StandardTokenWithFeeMock.sol
Lines: 102-104

Costly Loops

If array.length is large enough, the function may exceed the block gas limit, and transactions calling it will never be confirmed. This becomes a security issue, if an external entity influences array.length. Relevant mitigation techniques are numerous, but generally involve limitation to a bounded array.

File: contracts/lib/AddressArrayUtils.sol
Lines: 181-187
File: contracts/lib/AddressArrayUtils.sol
Lines: 323-327
File: contracts/lib/AddressArrayUtils.sol
Lines: 143-146

File: contracts/lib/AddressArrayUtils.sol
Lines: 155-160
File: contracts/lib/AddressArrayUtils.sol
Lines: 17-21
File: contracts/lib/AddressArrayUtils.sol
Lines: 61-63
File: contracts/lib/AddressArrayUtils.sol
Lines: 90-92
File: contracts/lib/AddressArrayUtils.sol
Lines: 105-110
File: contracts/lib/AddressArrayUtils.sol
Lines: 113-118
File: contracts/lib/AddressArrayUtils.sol
Lines: 58-60
File: contracts/lib/AddressArrayUtils.sol
Lines: 190-195
File: contracts/lib/AddressArrayUtils.sol
Lines: 343-345
File: contracts/lib/AddressArrayUtils.sol
Lines: 161-166
File: contracts/lib/AddressArrayUtils.sol
Lines: 147-152
File: contracts/lib/AddressArrayUtils.sol
Lines: 75-77
File: contracts/lib/Bytes32.sol
Lines: 27-27
File: contracts/lib/Bytes32.sol
Lines: 32-32
File: contracts/core/Vault.sol
Lines: 204-212
File: contracts/core/Vault.sol
Lines: 243-251
File: contracts/core/Vault.sol
Lines: 322-331
File: contracts/core/Vault.sol
Lines: 282-290
File: contracts/core/exchange-wrappers/TakerWalletWrapper.sol
Lines: 95-95
File: contracts/core/extensions/CoreIssuance.sol
Lines: 394-412
File: contracts/core/extensions/CoreIssuance.sol
Lines: 482-484
File: contracts/core/extensions/CoreIssuance.sol
Lines: 444-454
File: contracts/core/lib/OrderLibrary.sol
Lines: 210-221
File: contracts/core/TransferProxy.sol
Lines: 115-124
File: contracts/core/RebalancingSetToken.sol
Lines: 475-522
File: contracts/core/RebalancingSetToken.sol

Lines: 865-878

File: contracts/core/RebalancingSetToken.sol

Lines: 763-777

File: contracts/core/SetToken.sol

Lines: 109-147

File: contracts/core/modules/IssuanceOrderModule.sol

Lines: 611-620

File: contracts/core/modules/IssuanceOrderModule.sol

Lines: 321-321

File: contracts/core/modules/IssuanceOrderModule.sol

Lines: 641-657

If-Revert Instead of Require

Using the construction require(condition); instead of if (condition) {revert();} promotes general code readability.

File: contracts/lib/AddressArrayUtils.sol

Lines: 242-247

File: contracts/lib/AddressArrayUtils.sol

Lines: 270-272

File: contracts/lib/AddressArrayUtils.sol

Lines: 289-294

Multiple Return Values to Struct

Rather than utilizing multiple return values for internal or private functions, a struct may be used. It can improve code readability.

File: contracts/lib/AddressArrayUtils.sol

Line: 38

File: contracts/lib/AddressArrayUtils.sol

Line: 15

File: contracts/lib/AddressArrayUtils.sol

Line: 220

File: contracts/core/exchange-wrappers/KyberNetworkWrapper.sol

Line: 192

File: contracts/core/exchange-wrappers/KyberNetworkWrapper.sol

Line: 95

File: contracts/core/exchange-wrappers/KyberNetworkWrapper.sol

Line: 128

File: contracts/core/exchange-wrappers/TakerWalletWrapper.sol

Line: 78

File: contracts/core/exchange-wrappers/TakerWalletWrapper.sol

Line: 133

File: contracts/core/exchange-wrappers/ZeroExExchangeWrapper.sol

Line: 181

File: contracts/core/exchange-wrappers/ZeroExExchangeWrapper.sol

Line: 102

File: contracts/core/exchange-wrappers/ZeroExExchangeWrapper.sol

Line: 261

File: contracts/core/extensions/CoreIssuance.sol

Line: 389

File: contracts/core/extensions/CoreIssuance.sol

Lines: 435-438

File: contracts/core/lib/auction-price-libraries/IAuctionPriceCurve.sol

Line: 62

File: contracts/core/lib/auction-price-libraries/LinearAuctionPriceCurve.sol

Line: 91

File: contracts/core/RebalancingSetToken.sol

Line: 405

File: contracts/core/RebalancingSetToken.sol

Line: 455

File: contracts/core/RebalancingSetToken.sol

Line: 855

File: contracts/core/RebalancingSetToken.sol

Line: 944

File: contracts/core/interfaces/IExchangeWrapper.sol

Line: 53

File: contracts/mocks/core/exchange-wrappers/lib/ZeroExOrderDataHandlerMock.sol

Line: 39

File: contracts/mocks/core/exchange-wrappers/lib/ZeroExOrderDataHandlerMock.sol

Line: 21

File: contracts/mocks/core/lib/ConstantAuctionPriceCurve.sol

Line: 94

Assembly Usage

Assembly usage is traditionally cautioned against as it discards several important safety features of Solidity. In each instance found, assembly is only used in order to minimize gas consumption or perform functions otherwise incapable of being performed.

File: contracts/lib/ERC20Wrapper.sol

Lines: 189-209

File: contracts/core/exchange-wrappers/lib/ZeroExOrderDataHandler.sol

Lines: 167-170

File: contracts/core/exchange-wrappers/lib/ZeroExOrderDataHandler.sol

Lines: 85-89

File: contracts/core/exchange-wrappers/lib/ZeroExOrderDataHandler.sol

Lines: 130-141

File: contracts/core/exchange-wrappers/KyberNetworkWrapper.sol

Lines: 264-269

File: contracts/core/exchange-wrappers/TakerWalletWrapper.sol

Lines: 140-143

File: contracts/core/lib/ExchangeHeaderLibrary.sol

Lines: 64-69

File: contracts/core/lib/EIP712Library.sol

Lines: 73-83

File: contracts/core/RebalancingSetTokenFactory.sol

Lines: 180-186

File: contracts/external/0x/LibBytes.sol

Lines: 49-54

File: contracts/external/0x/LibBytes.sol

Lines: 158-186

File: contracts/external/0x/LibBytes.sol

Lines: 28-30

File: contracts/external/0x/LibBytes.sol

Lines: 102-107

File: contracts/external/0x/LibBytes.sol

Lines: 130-156

File: contracts/external/0x/LibBytes.sol

Lines: 80-82

Appendix
File Directory





