Report

v. 1.0

Customer

Index Coop

Smart Contract Audit

Index Protocol

19th June 2023

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	13.06.23	A. Zveryanskaya	Initial Draft
0.2	13.06.23	A. Zveryanskaya	Minor revision
1.0	19.06.23	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

On October 6th, 2020, the Index Coop was launched as an autonomous decentralized organization (DAO) by Set Labs. The community was made up of former TradFi professionals, engineers, and crypto enthusiasts.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

index-protocol/contracts/protocol/integration/lib/
AaveV3.sol
index-protocol/contracts/protocol/modules/v1/
AaveV3LeverageModule.sol
index-coop-smart-contracts/contracts/adapters/
AaveV3LeverageStrategyExtension.sol

4 Methodology

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

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

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

5 Our findings

We found 2 major, and a few less important issues. All Major issues have been fixed.

	Active	Fixed
Major	0	2
Moderate	0	0
Minor	5	4

Fixed 6 out of 11 issues

6 Major Issues

CVF-1. FIXED

- **Category** Procedural
- **Source** AaveV3.sol

Description This comment seems irrelevant to the function.

Recommendation Consider replacing it.

```
460 * Allows SetToken to borrow a specific `_amountNotional` of the
       ↪ reserve underlying `_asset`, provided that
    * the SetToken already deposited enough collateral, or it was given
       ↪ enough allowance by a credit delegator
    * on the corresponding debt token (StableDebtToken or
       ↪ VariableDebtToken)
```

CVF-2. FIXED

- **Category** Procedural
- **Source** AaveV3.sol

Recommendation This should be resolved.

```
465 @param _categoryId TODO: What is this ?
```

7 Minor Issues

CVF-4. FIXED

- **Category** Documentation
- **Source** AaveV3LeverageStrategyExtension.sol

Description “Extension of an extension” sounds weird.

Recommendation Consider rephrasing.

Client Comment *Technically it is an extension of the extension, but I agree it does sound weird. Changed this line to ”* Expanded version of the AaveLeverageStrategyExtension to add an endpoint for setting the eMode categoryId on AaveV3”.*

```
51 * Extension of AaveLeverageStrategyExtension to add endpoint for
     ↪ setting the eMode categoryId
```

CVF-5. FIXED

- **Category** Procedural
- **Source** AaveV3.sol

Description These are no “LendingPool” contracts in Aave V3.

Recommendation Consider rephrasing.

```
45  * @param _lendingPool          Address of the LendingPool contract
87  * @param _lendingPool          Address of the LendingPool contract
116 * @param _lendingPool          Address of the LendingPool contract
156 * @param _lendingPool      Address of the LendingPool contract
190 * @param _lendingPool          Address of the LendingPool contract
236 * @param _lendingPool          Address of the LendingPool contract
268 * @param _lendingPool          Address of the LendingPool contract
310 * @param _lendingPool          Address of the LendingPool contract
344 * @param _lendingPool          Address of the LendingPool contract
376 * @param _lendingPool          Address of the LendingPool contract
402 * @param _lendingPool      Address of the LendingPool contract
434 * @param _lendingPool      Address of the LendingPool contract
464 * @param _lendingPool          Address of the LendingPool contract
```

CVF-6. FIXED

- **Category** Procedural
- **Source** AaveV3.sol

Recommendation Consider renaming these arguments to "_pool" for consistency with the type name.

```
60  IPool _lendingPool,
94  IPool _lendingPool,
130 IPool _lendingPool,
166 IPool _lendingPool,
206 IPool _lendingPool,
244 IPool _lendingPool,
283 IPool _lendingPool,
321 IPool _lendingPool,
354 IPool _lendingPool,
383 IPool _lendingPool,
412 IPool _lendingPool,
441 IPool _lendingPool,
469 IPool _lendingPool,
```

CVF-7. INFO

- **Category** Procedural
- **Source** AaveV3LeverageModule.sol

Recommendation Consider renaming this field to “pool” for consistency with type name.

Client Comment *Agreed, it should be called pool here as well. However since this is a named struct field it would be a breaking change to the contract interface which we have been able to avoid so far. I will review this later again and might change it but for now I’m leaning towards keeping it as is, since the minor inaccuracy in naming might not be worth it. I added a comment though to explain this discrepancy in the naming.*

```
71 IPool lendingPool;                // Lending pool instance, we grab
   ↪ this everytime since it's best practice not to store
```

CVF-8. INFO

- **Category** Procedural
- **Source** AaveV3LeverageModule.sol

Description There are no lending pools in Aave V3.

Recommendation Consider rephrasing.

Client Comment *See CVF-7.*

```
71 IPool lendingPool;                // Lending pool instance, we grab
   ↪ this everytime since it's best practice not to store
```

CVF-9. INFO

- **Category** Bad naming
- **Source** AaveV3LeverageModule.sol

Recommendation Consider renaming this variable to “poolAddressProvider” for consistency with the type name.

Client Comment *Same situation as CVF-7, will review wether this minor inaccuracy in naming is worth changing the interface. (probably not, since it should be relatively clear that ”lendingPool” refers to ”pool” in aaveV3).*

```
208 IPoolAddressesProvider public immutable lendingPoolAddressesProvider
   ↪ ;
```

CVF-10. FIXED

- **Category** Bad naming
- **Source** AaveV3LeverageModule.sol

Recommendation Consider renaming this argument to “poolAddressProvider” for consistency with the type name.

```
235 IPoolAddressesProvider _lendingPoolAddressesProvider
```

CVF-11. INFO

- **Category** Suboptimal
- **Source** AaveV3LeverageModule.sol

Recommendation Consider using named errors are their gas efficiency doesn't depend on the name length.

Client Comment *This had to be done since an initial version of the contract had a compile size larger than the maximum that can be deployed on Ethereum. Shortening the messages was the easiest way to reduce the contract size, so we will have to keep this as is. (we might want to add an explanation of the codes in the documentation somewhere though).*

```
413 require(borrowAssetEnabled[_setToken][_repayAsset], "BNE");

416 require(notionalRepayQuantity > 0, "BBZ");

512     require(allowedSetTokens[_setToken], "NAS");

520 require(_setToken.isInitializedModule(getAndValidateAdapter(
    ↪ DEFAULT_ISSUANCE_MODULE_NAME)), "INI");

548     require(underlyingToReserveTokens[borrowAsset].variableDebtToken
        ↪ .balanceOf(address(setToken)) == 0, "VDR");

579 require(_setToken.isInitializedModule(address(_debtIssuanceModule)),
    ↪  "INI");

605 require(address(underlyingToReserveTokens[_underlying].aToken) ==
    ↪ address(0), "MAE");

610 require(isActive, "IAE");

640     require(collateralAssetEnabled[_setToken][collateralAsset], "CNE
        ↪ ");

673     require(borrowAssetEnabled[_setToken][borrowAsset], "BNE");
        require(underlyingToReserveTokens[borrowAsset].variableDebtToken
        ↪ .balanceOf(address(_setToken)) == 0, "VDR");

689 require(controller.isSet(address(_setToken)) || allowedSetTokens[
    ↪ _setToken], "IST");
```

(734, 755, 875, 1143, 1163, 1167, 1184)

CVF-12. INFO

- **Category** Procedural
- **Source** AaveV3LeverageModule.sol

Recommendation Consider renaming the "_lendingPool" arguments into just "_pool" for consistency with the type name.

783
```
function _deposit(ISetToken _setToken, IPool _lendingPool, IERC20
    ↪ _asset, uint256 _notionalQuantity) internal {
```

793
```
function _withdraw(ISetToken _setToken, IPool _lendingPool, IERC20
    ↪ _asset, uint256 _notionalQuantity) internal {
```

802
```
function _repayBorrow(ISetToken _setToken, IPool _lendingPool,
    ↪ IERC20 _asset, uint256 _notionalQuantity) internal {
```

821
```
function _borrow(ISetToken _setToken, IPool _lendingPool, IERC20
    ↪ _asset, uint256 _notionalQuantity) internal {
```

ABDK
Consulting

About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting