ABDK CONSULTING

SMART CONTRACT AUDIT

Voltz

CompoundRateOracle

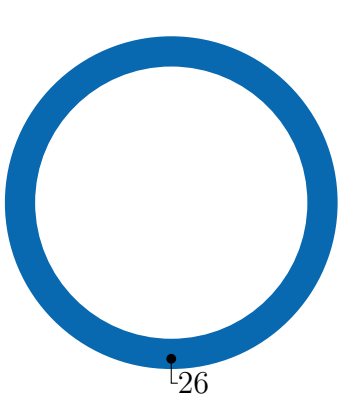
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
25th May 2022

We've been asked to review 4 files in a Github repository. We found 26 minor issues.



Findings

ID	Severity	Category	Status
CVF-1	Minor	Procedural	Info
CVF-2	Minor	Procedural	Info
CVF-3	Minor	Suboptimal	Info
CVF-4	Minor	Procedural	Info
CVF-5	Minor	Suboptimal	Info
CVF-6	Minor	Suboptimal	Info
CVF-7	Minor	Overflow/Underflow	Fixed
CVF-8	Minor	Suboptimal	Fixed
CVF-9	Minor	Overflow/Underflow	Fixed
CVF-10	Minor	Overflow/Underflow	Fixed
CVF-11	Minor	Suboptimal	Info
CVF-12	Minor	Documentation	Fixed
CVF-13	Minor	Procedural	Info
CVF-14	Minor	Bad datatype	Info
CVF-15	Minor	Procedural	Fixed
CVF-16	Minor	Suboptimal	Info
CVF-17	Minor	Suboptimal	Info
CVF-18	Minor	Procedural	Fixed
CVF-19	Minor	Suboptimal	Info
CVF-20	Minor	Unclear behavior	Fixed
CVF-21	Minor	Procedural	Info
CVF-22	Minor	Procedural	Info
CVF-23	Minor	Procedural	Info
CVF-24	Minor	Procedural	Info
CVF-25	Minor	Suboptimal	Fixed
CVF-26	Minor	Bad datatype	Fixed

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	May 24, 2022	D. Khovratovich	Initial Draft
0.2	May 24, 2022	D. Khovratovich	Minor revision
1.0	May 25, 2022	D. Khovratovich	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting 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.
We have reviewed the contracts at repository:

- contracts/CompoundFCM.sol
- contracts/interfaces/fcms/ICompoundFCM.sol
- contracts/interfaces/rate_oracles/ICompoundRateOracle.sol
- contracts/rate_oracles/CompoundRateOracle.sol

The fixes were provided in a new commit.

2.1 About ABDK

ABDK Consulting, 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.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best 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 and that 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 is 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 that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check 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.

3 Detailed Results

3.1 CVF-1

- **Severity** Minor
- **Status**
- **Category** Procedural
- **Source** CompoundFCM.sol

Description Specifying a particular version makes it harder to migrate to newer versions. Consider specifying as "^0.8.0". Also relevat for: CompoundRateOracle.sol, ICompoundRateOracle.sol.
Recommendation Happy to keep specific version and change when required.
Client Comment Info

Listing 1:

```
3 pragma solidity =0.8.9;
```

3.2 CVF-2

- **Severity** Minor
- **Status**
- **Category** Procedural
- **Source** CompoundFCM.sol

Recommendation Understood.
Client Comment Info

Listing 2:

```
7  import "./storage/FCMStorage.sol";

12 import "prb-math/contracts/PRBMathUD60x18.sol";

16 import "./utils/Printer.sol";
```

3.3 CVF-3

- **Severity** Minor
- **Status**
- **Category** Suboptimal
- **Source** CompoundFCM.sol

Description The cast is redundant.
Recommendation Won't fix.
Client Comment Info

Listing 3:

```
36 if (msg.sender != address(_marginEngine)) {
```

3.4 CVF-4

- **Severity** Minor
- **Status**
- **Category** Procedural
- **Source** CompoundFCM.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.
Recommendation In both cases, the exsisting comment above will suffice.
Client Comment Info

Listing 4:

```
44 constructor () initializer {}

82 function _authorizeUpgrade(address) internal override onlyOwner
   ↪ {}
```

3.5 CVF-5

- **Severity** Minor
- **Status**
- **Category** Suboptimal
- **Source** CompoundFCM.sol

Description Should be "__Pausable_init_unchained".
Recommendation Clarity worth the suboptimality.
Client Comment Info

Listing 5:

```
57 __Pausable_init();
```

3.6 CVF-6

- **Severity** Minor
- **Status**
- **Category** Suboptimal
- **Source** CompoundFCM.sol

Description Should be "__UUPSUpgradeable_init_unchained".
Recommendation Clarity worth the suboptimality.
But out of interest, why? Who will call ERC1967UpgradeUpgradeable's init? I appreciate they may all be empty but I'd rather not rely on implementation detail.
Client Comment Info

Listing 6:

```
58 __UUPSUpgradeable_init();
```

3.7 CVF-7

- **Severity** Minor
- **Status**
- **Category** Overflow/Underflow
- **Source** CompoundFCM.sol

Description Consider adding a range check for tickSpacing at initialization.
Recommendation Already require tickSpacing > 0 at initialisation.
Client Comment Info

Listing 7:

```
113 tickLower: -tickSpacing,

196 tickLower: -tickSpacing,
```

3.8 CVF-8

- **Severity** Minor
- **Status**
- **Category** Suboptimal
- **Source** CompoundFCM.sol

Description Consider handling this case explicitly.
Client Comment Fixed

Listing 8:

```
127 uint256 yieldBearingTokenDelta = uint256(-variableTokenDelta).
      ↪ wadDiv(currentExchangeRate);

187 require(uint256(-trader.variableTokenBalance) >=
      ↪ notionalToUnwind, "notional to unwind > notional");

194       amountSpecified: -notionalToUnwind.toInt256(),

255 uint256 marginToCoverVariableLegFromNowToMaturity = uint256(-
      ↪ traderVariableTokenBalance);

262     if (-remainingSettlementCashflow >
          ↪ marginToCoverRemainingSettlementCashflow) {

325 trader.updateBalancesViaDeltas(-trader.fixedTokenBalance, -
      ↪ trader.variableTokenBalance);
```

3.9 CVF-9

- **Severity** Minor
- **Status**
- **Category** Overflow/Underflow
- **Source** CompoundFCM.sol

Description Consider adding an explicit check that it is not positive.
Client Comment Fixed

Listing 9:

```
255 uint256 marginToCoverVariableLegFromNowToMaturity = uint256(-
    ↪ traderVariableTokenBalance);
```

3.10 CVF-10

- **Severity** Minor
- **Status**
- **Category** Overflow/Underflow
- **Source** CompoundFCM.sol

Description Consider using safe conversion.
Client Comment Fixed

Listing 10:

```
256 int256 marginToCoverRemainingSettlementCashflow = int256(
    ↪ getTraderMarginInUnderlyingTokens(
    ↪ traderMarginInScaledYieldBearingTokens)) - int256(
    ↪ marginToCoverVariableLegFromNowToMaturity);
```

3.11 CVF-11

- **Severity** Minor
- **Status**
- **Category** Suboptimal
- **Source** CompoundFCM.sol

Description Consider renaming the error.
Recommendation Agreed but won't change.
Client Comment Info

Listing 11:

```
309 revert CannotSettleBeforeMaturity();
```

3.12 CVF-12

- **Severity** Minor
- **Status**
- **Category** Documentation
- **Source** CompoundFCM.sol

Description Consider documenting.
Client Comment Fixed

Listing 12:

```
348 /// @dev in case of Compound this is done by redeeming the
    ↪ underlying token directly from the cToken: https://
    ↪ compound.finance/docs/ctokens#redeem-underlying
```

3.13 CVF-13

- **Severity** Minor
- **Status**
- **Category** Procedural
- **Source** CompoundRateOracle.sol

Recommendation OK
Client Comment Info

Listing 13:

```
6 import "../interfaces/compound/ICToken.sol";
```

3.14 CVF-14

- **Severity** Minor
- **Status**
- **Category** Bad datatype
- **Source** CompoundRateOracle.sol

Description The array size should be a named constant.
Recommendation Agreed but matches Uniswap so won't change.
Client Comment Info

Listing 14:

```
12 using OracleBuffer for OracleBuffer.Observation[65535];
```

3.15 CVF-15

- **Severity** Minor
- **Status**
- **Category** Procedural
- **Source** CompoundRateOracle.sol

Description These variables should be declared as immutable.
Client Comment Fixed

Listing 15:

```
15 ICToken public override ctoken;

18 uint256 public override decimals;
```

3.16 CVF-16

- **Severity** Minor
- **Status**
- **Category** Suboptimal
- **Source** CompoundRateOracle.sol

Description No need to pass separately.
Recommendation Decaimals not mandatory part of ERC20 standard. Won't change.
Client Comment Info

Listing 16:

```
25 uint8 _decimals
```

3.17 CVF-17

- **Severity** Minor
- **Status**
- **Category** Suboptimal
- **Source** CompoundRateOracle.sol

Description Consider removing the argument.
Recommendation Agreed but acts as sanity check. Won't change.
Client Comment Info

Listing 17:

```
29 ctoken.underlying() == address(underlying),
```

3.18 CVF-18

- **Severity** Minor
- **Status**
- **Category** Procedural
- **Source** CompoundRateOracle.sol

Description These values and the condition can be precomputed and stored in immutable variables for performance
Client Comment Fixed

Listing 18:

```
53 if (decimals >= 17) {
       uint256 scalingFactor = 10**(decimals - 17);

57     uint256 scalingFactor = 10**(17 - decimals);
```

3.19 CVF-19

- **Severity** Minor
- **Status**
- **Category** Suboptimal
- **Source** CompoundRateOracle.sol

Description Consider handling it appropriately for efficiency.
Recommendation Decimals = 17 so uncommon that extra branching may cost more in aggregate than savings. Won't fix.
Client Comment Info

Listing 19:

```
53 if (decimals >= 17) {
```

3.20 CVF-20

- **Severity** Minor
- **Status**
- **Category** Unclear behavior
- **Source** CompoundRateOracle.sol

Description Consider clarifying.
Client Comment Fixed

Listing 20:

```
65 /// @param index The index of the Observation that was most
   ↪ recently written to the observations buffer
```

3.21 CVF-21

- **Severity** Minor
- **Status**
- **Category** Procedural
- **Source** CompoundRateOracle.sol

Description No problem - it was covered in earlier audit.
Client Comment Info

Listing 21:

```
216 rateValueRay = interpolateRateValue(
```

3.22 CVF-22

- **Severity** Minor
- **Status**
- **Category** Procedural
- **Source** ICompoundFCM.sol

Description ok
Client Comment Info

Listing 22:

```
5 import "../compound/ICToken.sol";
```

3.23 CVF-23

- **Severity** Minor
- **Status**
- **Category** Procedural
- **Source** ICompoundFCM.sol

Description ok
Client Comment Info

Listing 23:

```
6 import "../IERC20Minimal.sol";
```

3.24 CVF-24

- **Severity** Minor
- **Status**
- **Category** Procedural
- **Source** ICompoundRateOracle.sol

Description ok
Client Comment Info

Listing 24:

```
4 import "../compound/ICToken.sol";
```

3.25 CVF-25

- **Severity** Minor
- **Status**
- **Category** Suboptimal
- **Source** ICompoundRateOracle.sol

Description As this file is located in a directory named "rate_oracles", the path could be reduced to "./IRateOracle.sol".
Client Comment Fixed

Listing 25:

```
5 import "../rate_oracles/IRateOracle.sol";
```

3.26 CVF-26

- **Severity** Minor
- **Status**
- **Category** Bad datatype
- **Source** ICompoundRateOracle.sol

Description Consider changing the return type to "uint8" for consistency.
Client Comment Fixed

Listing 26:

```
15 function decimals() external view returns (uint);
```